KR
2018

International Conference on Principles of
Knowledge Representation and Reasoning

Tempe, October 27th - 29th 2018

17th INTERNATIONAL WORKSHOP ON NON-MONOTONIC REASONING

NMR 2018

Eduardo Fermé
University of Madeira
Serena Villata
Université Côte d'Azur

Preface

NMR is the premier forum for results in the area of Non-Monotonic Reasoning. Its aim is to bring together active researchers in this broad field within knowledge representation and reasoning (KR), including belief revision, uncertain reasoning, reasoning about actions, planning, logic programming, preferences, argumentation, causality, and many other related topics including systems and applications.

The NMR workshop series is the premier specialized forum for researchers in non-monotonic reasoning and related areas. This will be the 16th workshop in these series. Its aim is to bring together active researchers in the broad area of non-monotonic reasoning, including belief revision, reasoning about actions, argumentation, declarative programming, preferences, non-monotonic reasoning for ontologies, uncertainty, and other related topics.

This volume contains the papers presented at NMR-2018: 17th International Workshop on Non-Monotonic Reasoning held on October 20-29, 2018 in Tempe. There were 22 submissions. Each submission was reviewed by at least 2, and on the average 3 program committee members. The committee decided to accept 20 papers. The program also includes two invited talks by João Leite by the New University of Lisbon and Renata Wassermann by the University of São Paulo.

October 21, 2018
Tempe, Arizona

Eduardo Fermé
Serena Villata

Table of Contents

Program Committee

Christoph Beierle	University of Hagen
Alexander Bochman	Computer Science Dept., Holon Institute of Technology
Marina De Vos	University of Bath
Juergen Dix	Clausthal University of Technology
Wolfgang Faber	Alpen-Adria-Universität Klagenfurt
Eduardo Fermé	Universidade da Madeira
Martin Gebser	University of Potsdam
Michael Gelfond	Texas Tech University
Sven Ove Hansson	Royal Institute of Technology, Stockholm
Andreas Herzig	CNRS, IRIT, Univ. Toulouse
Anthony Hunter	University College London
Katsumi Inoue	NII
Tomi Janhunen	Aalto University
Gabriele Kern-Isberner	Technische Universitaet Dortmund
Sébastien Konieczny	CRIL - CNRS
Thomas Lukasiewicz	University of Oxford
Maria Vanina Martinez	Instituto de Ciencias e Ingeniería de la Computación (CONICET - Universidad Nacional del Sur in Bahia Blanca)
Thomas Meyer	University of Cape Town and CAIR
Nir Oren	University of Aberdeen
Odile Papini	LSIS UMR CNRS 7296
Pavlos Peppas	University of Patras
Laurent Perrussel	IRIT - Universite de Toulouse
Ramon Pino Perez	Universidad de Los Andes
Ken Satoh	National Institute of Informatics and Sokendai
Steven Schockaert	Cardiff University
Gerardo Simari	Universidad Nacional del Sur and CONICET
Guillermo R. Simari	Universidad del Sur in Bahia Blanca
Matthias Thimm	Universität Koblenz-Landau
Ivan Varzinczak	Univ. Artois and CNRS
Joost Vennekens	Katholieke Universiteit Leuven
Serena Villata	CNRS - Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis
Renata Wassermann	University of São Paulo
Emil Weydert	CSC, University of Luxembourg
Stefan Woltran	Vienna University of Technology

Additional Reviewers

B

Budan, Maximiliano
F

Fandinno, Jorge
N

Nitta, Katsumi
R

Romero, Javier
S

Saribatur, Zeynep G.
Sauerwald, Kai
T

Tojo, Satoshi

A Tutorial for Weighted Bipolar Argumentation with Continuous Dynamical Systems and the Java Library Attractor

Nico Potyka
Institute of Cognitive Science, University of Osnabrück, Germany

Abstract

Weighted bipolar argumentation frameworks allow modeling decision problems and online discussions by defining arguments and their relationships. The strength of arguments can be computed based on an initial weight and the strength of attacking and supporting arguments. While previous approaches assumed an acyclic argumentation graph and successively set arguments' strength based on the strength of their parents, recently continuous dynamical systems have been proposed as an alternative. Continuous models update arguments' strength simultaneously and continuously. While there are currently no analytical guarantees for convergence in general graphs, experiments show that continuous models can converge quickly in large cyclic graphs with thousands of arguments. Here, we focus on the high-level ideas of this approach and explain key results and applications. We also introduce *Attractor*, a Java library that can be used to solve weighted bipolar argumentation problems. *Attractor* contains implementations of several discrete and continuous models and numerical algorithms to compute solutions. It also provides base classes that can be used to implement, to evaluate and to compare continuous models easily.

1 Introduction

Abstract argumentation (Dung 1995) studies the acceptability of arguments based purely on their relationships and abstracted from their content. The basic framework has a two-valued semantics and allows only defining arguments and an attack relation between them. This basic setting has been extended in different directions. For example, *bipolar argumentation frameworks* (Amgoud, Cayrol, and Lagasquie-Schiex 2004; Oren and Norman 2008; Cayrol and Lagasquie-Schiex 2013; Polberg and Oren 2014) take account of the fact that arguments cannot only attack each other and add a support relation. A survey of different approaches can be found in (Cohen et al. 2014). The classical two-valued semantics that distinguishes only between acceptance and rejection of arguments has been extended in various ways. Examples include probabilistic semantics (Thimm 2012; Hunter 2013; Hunter and Potyka 2017) and ranking semantics that can be based on fixed point equations (Besnard and Hunter 2001; Leite and Martins 2011; Correia, Cruz, and Leite 2014; Barringer, Gabbay, and Woods 2012) or the graph structure (Cayrol and Lagasquie-Schiex 2005; Amgoud and Ben-Naim 2013). Other recent extensions include recursive attacks on attacks (Baroni et al. 2011) and the temporal availability of arguments (Budán et al. 2015).

Our focus here is on *weighted bipolar argumentation frameworks* that allow defining attack and support relationships and an initial weight for arguments (Baroni et al. 2015; Rago et al. 2016; Amgoud and Ben-Naim 2017; Mossakowski and Neuhaus 2018). A strength value is computed for every argument based on its initial weight and the strength of its attackers and supporters. Examples for computational models include the *QuAD algorithm* from (Baroni et al. 2015) that was designed to evaluate the strength of answers in decision-support systems. Soon after, the *DF-QuAD algorithm* (Rago et al. 2016) was proposed as an alternative, which avoids discontinuous behaviour of the QuAD algorithm that can be undesirable in some applications. Some additional interesting guarantees are given by the *Euler-based semantics* introduced in (Amgoud and Ben-Naim 2017). The QuAD algorithms mainly lack these properties due to the fact that their aggregated strength values *saturate*. That is, as soon, as an attacker (supporter) with strength 1 exists, the other attackers (supporters) become irrelevant for the aggregated value. However, while the Euler-based semantics avoids these problems, it has some other drawbacks that may be undesirable. Arguments initialized with strength 0 or 1 remain necessarily unchanged under Euler-based semantics and the impact of attacks and supports is non-symmetrical. The *quadratic energy model* introduced in (Potyka 2018a) avoids these problems. In particular, while the previous approaches are discrete in nature, the quadratic energy model is a continuous model. Discrete models often assume that the argumentation graph is acyclic, so that the strength of arguments can be computed successively according to a topological ordering. Continuous models change arguments' strength continuously and simultaneously. They can be naturally applied to cyclic graphs, but convergence in general remains an open question.

More formally, continuous models correspond to n-dimensional functions $f(t)$ mapping continuous points in time to n-dimensional state vectors whose i-th component represents the strength of the i-th argument at time t. The initial state $f(0)$ is given by the initial weights and a sys-

tem of differential equations describes how the strength values evolve as time progresses. This approach, in particular, allows plotting the evolution of strength values in order to better understand the final strength values of the limit $s = \lim_{t\to\infty} f(t)$ or to inspect the convergence behaviour visually. Even though convergence in general graphs is an open question, so far no diverging example has been found and tests with large randomly generated bipolar graphs show that the strength values converge in many cases.

In this tutorial paper, we will review some ideas and results from (Potyka 2018a) with a stronger focus on the high-level ideas. The goal of this paper is, in particular, to demonstrate how results can be applied to

1. solve weighted bipolar argumentation problems with continuous models,
2. transform existing discrete models to well-defined continuous models.

We also introduce *Attractor*, a Java library that provides basic implementations of the ideas discussed here and in (Potyka 2018a). The main goals of *Attractor* are to

1. simplify applying continuous models to weighted bipolar argumentation problems,
2. to improve reproducibility of the results in (Potyka 2018a),
3. to simplify implementing new continuous models and
4. to simplify comparing different models.

These goals are achieved by providing

1. implementations of some continuous models and the random generator introduced in (Potyka 2018a),
2. implementations of base classes that solve initial value problems with basic and advanced methods,
3. implementations of utility classes for benchmarking, plotting and working with benchmark files.

We will start with an introduction to dynamical systems and the quadratic energy model from (Potyka 2018a) in Section 2. In Section 3, we will discuss the problem of computing solutions and explain some important algorithms. Section 4 contains some additional information on convergence guarantees and open questions and discusses the computational complexity of the continuous approach. In Section 5, we explain how discrete models can be transformed to continuous models. Finally, Section 6 explains how the previously discussed ideas can be put into practice using *Attractor*.

2 Dynamical Systems and The Quadratic Energy Model

Roughly speaking, a dynamical system describes the evolution of a natural or technical system over time. If time is discretized, the system is called discrete, otherwise it is called continuous. Formally, we describe the state of the system at time t by a function $s(t)$. The state is usually given as a real vector and a system of differential equations describes how the system evolves dependent on the current state.

In the context of weighted argumentation, a state vector contains one component for every argument that can take a strength value between 0 and 1. The strength of arguments should evolve based on the initial weight, and the current strength of attackers and supporters. Before describing this approach in more detail, we define *weighted bipolar argumentation graphs (BAGs)* as introduced in (Amgoud and Ben-Naim 2017).

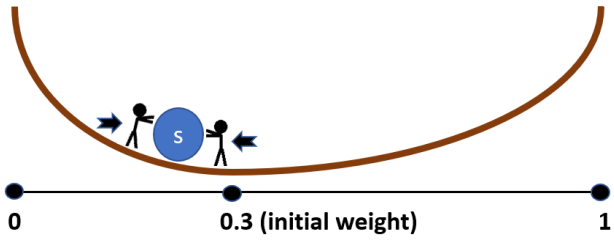


Figure 1: Illustration of dynamical system.

Definition 1 (BAG)**.** A *BAG* is a quadruple $\mathbf{A} = (\mathcal{A}, \mathrm{w}, \mathcal{R}, \mathcal{S})$, where $\mathcal{A}$ is a finite set of arguments, $\mathrm{w} : \mathcal{A} \to [0, 1]$ is a weight function and $\mathcal{R}$ and $\mathcal{S}$ are binary relations on $\mathcal{A}$ called *attack* and *support*.

In order to simplify notation, we assume that the i-th argument is called i, that is, $\mathcal{A} = \{1, \dots, n\}$. The weight function w defines an initial strength value between 0 and 1 for each argument. If $a\mathcal{R}b$ ($a\mathcal{S}b$), we say that a attacks (supports) b. We let $\mathrm{Att}_i = \{h \in \mathcal{A} \mid h\mathcal{R}i\}$ denote i's attackers and let $\mathrm{Sup}_i = \{h \in \mathcal{A} \mid h\mathcal{S}i\}$ denote i's supporters.

We can now describe our dynamical system more precisely. A state is a vector $s \in \mathbb{R}^n$ whose i-th component s_i is the strength of argument i. Our state model is a function $s : \mathbb{R}_0^+ \to \mathbb{R}^n$ that maps non-negative time points to strength vectors. That is, $s_i(t)$ is the strength of argument i at time t. Initially, the strength of an argument should correspond to its initial weight, that is, we let $s_i(0) = \mathrm{w}(i)$. The evolution of the system should be based on three considerations:

1. Strength values are attracted by their initial weight.
2. Attackers force the strength value towards 0 proportionally to their strength.
3. Supporters force the strength value towards 1 proportionally to their strength.

Intuitively, there are three forces acting on the strength of each argument as illustrated in Figure 1. In this physical metaphor, attackers push the strength towards 0, while the supporters push the strength towards 1. Gravity pulls the strength back to its initial weight.

This intuition can be modeled by a system of differential equations. If this system is designed carefully, it uniquely defines a model $s_{\mathbf{A}} : \mathbb{R}_0^+ \to \mathbb{R}^n$ for every BAG $\mathbf{A}$. We are then interested in the long-term behaviour of the model. Intuitively, we expect the forces to counteract until the strength reaches an equilibrium state where all forces are in balance. More formally, if the model converges to a state $s^* = \lim_{t\to\infty} s(t)$ as time progresses, we call s^* the *equilibrium state* reached by the model.

The quadratic energy model introduced in (Potyka 2018a) is defined as follows.

Definition 2 (The Quadratic Energy Model)**.** Let $\mathbf{A}$ be a BAG. For all $j \in \mathcal{A}$, the *energy at* j is defined as

$$E_j = \sum_{i \in \mathrm{Sup}_j} s_i - \sum_{i \in \mathrm{Att}_j} s_i$$

and for all $x \in \mathbb{R}$, the *impact of* x is defined as

$$h(x) = \frac{\max\{x,0\}^2}{1+\max\{x,0\}^2}.$$

The *quadratic energy model* $\sigma^{\mathbf{A}} : \mathbb{R}_0^+ \to \mathbb{R}^n$ for $\mathbf{A}$ is the unique solution of the system of differential equations

$$\frac{\mathrm{d}s_j}{\mathrm{d}t} = \mathrm{w}(j) - s_j + (1-\mathrm{w}(j)) \cdot h(E_j) - \mathrm{w}(j) \cdot h(-E_j), \qquad j \in \mathcal{A} \quad (1)$$

with initial conditions $s_j(0) = \mathrm{w}(j)$.

Intuitively, $\frac{\mathrm{d}s_j}{\mathrm{d}t}(t)$ describes the momentary change of s_j at time t. If $\frac{\mathrm{d}s_j}{\mathrm{d}t}(t) > 0$, the strength will increase, if $\frac{\mathrm{d}s_j}{\mathrm{d}t}(t) < 0$, the strength will decrease and if $\frac{\mathrm{d}s_j}{\mathrm{d}t}(t) = 0$ the strength will remain in its current state. The definition uses two auxiliary functions. The *energy* E_j at argument j aggregates the strength of attackers and supporters in a linear fashion. This notion of energy has been first defined for the Euler-based restricted semantics in (Amgoud and Ben-Naim 2017). The energy is then fed into the *impact* function h that is 0 for all negative arguments and then strictly increases, but is bounded from above by 1. The definition of $\frac{\mathrm{d}s_j}{\mathrm{d}t}$ can be divided into three parts that correspond to our three considerations above.

1. The difference $\mathrm{w}(j) - s_j$ draws the strength of an argument to its initial weight. Notice that if $\mathrm{w}(j) > s_j$ ($\mathrm{w}(j) < s_j$), this term is positive (negative) and tends to increase (decrease) j's strength.
2. The term $-\mathrm{w}(j) \cdot h(-E_j)$ moves the strength towards 0 if the negative force of attackers is stronger than the positive force of supporters.
3. Dually, the term $(1-\mathrm{w}(j)) \cdot h(E_j)$ moves the strength towards 1 if the positive force of supporters is stronger than the negative force of attackers.

As shown in (Potyka 2018a), the quadratic energy model is well-defined. That is, the system has a unique solution $\sigma^{\mathbf{A}}$ by means of which we can simulate the evolution of strength values over time. If $\sigma^{\mathbf{A}}$ reaches an equilibrium state, the final strength values at every argument are completely determined by the energy at this state as explained in the following proposition.

Proposition 1 (Strength in Equilibrium (Potyka 2018a))**.** *If the limit* $s^* = \lim_{t\to\infty} \sigma^{\mathbf{A}}(t)$ *exists, then we have*

$$s_j^* = \begin{cases} \mathrm{w}(j) & \text{if } E_j = 0 \\ \mathrm{w}(j) + (1-\mathrm{w}(j)) \cdot h(E_j) & \text{if } E_j > 0 \\ \mathrm{w}(j) - \mathrm{w}(j) \cdot h(-E_j). & \text{if } E_j < 0 \end{cases} \quad (2)$$

Equation 2 shows, in particular, that the strength will be the initial weight if the energy is 0 and otherwise will go to 1 (0) as the energy goes to ∞ ($-\infty$). The quadratic energy model satifies a collection of postulates proposed in (Amgoud and Ben-Naim 2017). These postulates range from very general properties like *Anonymity* (strength values do not depend on the identity of the argument) and *Independence* (arguments are independent of disconnected arguments) to properties tailor-made for weighted bipolar argumentation frameworks that guarantee that attacks, supports and initial weights have the intended meaning. An interesting property that distinguishes weighted argumentation frameworks from some other numerical argumentation frameworks is *Directionality*, which guarantees that arguments influence other arguments only in direction of the edges. This property distinguishes weighted argumentation approaches from probabilistic approaches that usually cause influence in both directions due to the nature of probability theory. Please see (Amgoud and Ben-Naim 2017) and (Potyka 2018a) for a more thorough discussion of the properties.

In the following example, we illustrate the quadratic energy model by means of a small decision problem.

Example 1. *Suppose we want to decide whether to buy or to sell stocks of an electronics company. We base our decision on information given by different experts:*

1: *The development of the new phone was too expensive. Therefore, the company has to cut down research and development and will not stay competitive in the future.*

2: *The company's new phone is innovative and will increase the company's profit considerably.*

3: *The price of the new phone is too high and there will not be too many sales.*

4: *There is a large number of preorders of the new phone already.*

5: *The company's investment in research and development is far beyond competitors' investment and the company is likely to become the market leader in the future.*

Initially, we do not have a preference for buying or selling stocks and set both initial weights to 0.5*. In order to weigh the expert opinions, we could use historical information about how frequently the expert's assessment was true or false. If there were* t *true and* f *false assessments, we could set the initial weight to* $\frac{t}{t+f}$*. In order to incorporate arguments of new experts and to set an initial bias for the weight, we could add pseudocounts to* t *and* f*. That is, we set the initial weight to* $\frac{t+t'}{t+t'+f+f'}$*, where* $t', f' \in \mathbb{N}$ *are pseudocounts that encode an initial bias. If* $t' = f'$*, our initial weight is* 0.5 *when no historical information is available. Setting* $t' > f'$ *(*$t' < f'$*), the weight will initially be greater (smaller) than* 0.5*. The larger* $t' + f'$ *is, the more data is needed to deviate from the bias. For instance, if* $t = 5$*,* $f = 2$*, then we have the weight* $\frac{5+1}{5+1+2+1} \approx 0.66$ *for pseudocounts* $t' = f' = 1$*, whereas we have* $\frac{5+10}{5+10+2+10} \approx 0.55$ *for pseudocounts* $t' = f' = 10$*. Figure 2 shows a BAG for our problem along with initial weights and the final strength values under the quadratic energy model. One advantage of a continuous model is that we can illustrate the evolution of*

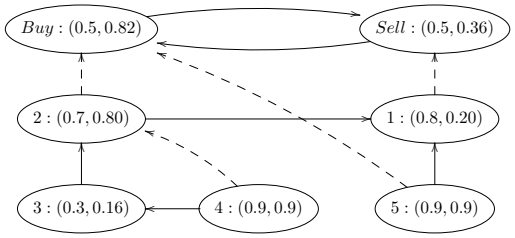


Figure 2: BAG for stock examples. Nodes show (initial weight, final strength) under quadratic energy model.

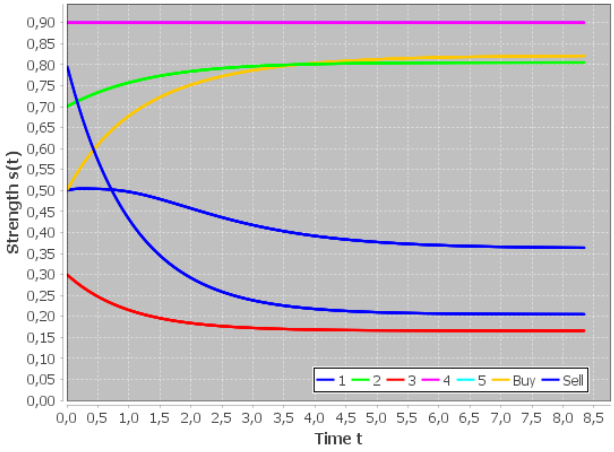


Figure 3: Long-term behaviour of $\sigma^{\mathbf{A}}$ for BAG in Figure 2.

the strength values by drawing function graphs for selected arguments over time. This makes it easier to explain the final strength values. Recall that a state in our dynamical system contains a strength value for every argument. Our example is sufficiently small to draw all function graphs simultaneously without making the picture too messy. The graphs are shown in Figure 3. For example, the blue graph starting from 0.5 *shows the evolution of the strength of the sell-argument. Initially, its strength slightly increases due to the support by argument* 1 *(the blue graph starting from* 0.8*). However, argument* 1 *becomes gradually weaker due to its attackers. At about time* 0.5*, the attacking buy-argument becomes as strong as argument* 1 *(the blue and yellow graphs intersect) and the strength of the selling-argument starts decreasing. It actually starts decreasing slightly before that because it is also drawn to its initial weight* 0.5*.*

3 Numerical Computation of Solutions

Even though we know that the quadratic energy model $\sigma^{\mathbf{A}}$ is well-defined for every BAG, we usually cannot solve for $\sigma^{\mathbf{A}}$ analytically. This is the case for most nonlinear dynamical systems, but is not a heavy drawback in practice since the solution can be approximated numerically.

The intuitive idea is best illustrated by *Euler's method*, which is a simple algorithm to perform this approximation. Recall that $\frac{\mathrm{d}s_j}{\mathrm{d}t}(t)$ describes the momentary change of s_j at time t. Hence, if we know $s_j(t)$, we can approximate $s_j(t+\delta)$ by letting $\hat{s}_j(t+\delta) = s_j(t) + \delta \cdot \frac{\mathrm{d}s_j}{\mathrm{d}t}(t)$. Formally, this approach is justified by the fact that differentiable functions can be approximated locally by the derivative. In particular, as we let the step size δ go to 0, the approximation error $|\hat{s}_j(t+\delta) - s_j(t+\delta)|$ goes to 0 as well.

EulerApproximation($\mathbf{A}$, δ, ϵ):
  $t \leftarrow 0$
  for $i \in \mathcal{A}$:
    $s_i \leftarrow \mathrm{w}(i)$
  while $\|\frac{\mathrm{d}s}{\mathrm{d}t}\|_\infty > \epsilon$:
    for $i \in \mathcal{A}$:
      $s'_i \leftarrow s_i + \delta \cdot \frac{\mathrm{d}s_i}{\mathrm{d}t}$
    $s \leftarrow s'$
  return s

Figure 4: Euler's method for approximating the energy model $\sigma^{\mathbf{A}}$ given a BAG $\mathbf{A} = (\mathcal{A}, \mathrm{w}, \mathcal{R}, \mathcal{S})$, step size δ and convergence threshold ϵ.

In the context of weighted bipolar argumentation, we can initialize all strength values with the initial weights. This gives us $s(0)$ and we can compute $\frac{\mathrm{d}s_j}{\mathrm{d}t}(0)$ according to Definition 2. We can then approximate $\sigma^{\mathbf{A}}(\delta)$ by letting $\hat{s}(\delta) = s(0) + \delta \cdot \frac{\mathrm{d}s}{\mathrm{d}t}(0)$. Given our approximation $\hat{s}(\delta)$, we can compute $\frac{\mathrm{d}\hat{s}}{\mathrm{d}t}(\delta)$ and $\hat{s}(2\cdot\delta) \approx \hat{s}(\delta) + \delta \cdot \frac{\mathrm{d}\hat{s}}{\mathrm{d}t}(\delta)$. Continuing in this way, we can compute $\hat{s}(n \cdot \delta)$ for arbitrary $n \in \mathbb{N}$. Hopefully, the strength values will eventually converge. This is the case if and only if the derivative $\frac{\mathrm{d}\hat{s}}{\mathrm{d}t}(t)$ goes to 0. Therefore, a simple termination condition is to demand that $\frac{\mathrm{d}\hat{s}_i}{\mathrm{d}t}(t) \leq \epsilon$ for all $i \in \mathcal{A}$ and some small $\epsilon > 0$. Formally, this corresponds to demanding that the maximum norm of $\frac{\mathrm{d}\hat{s}}{\mathrm{d}t}(t)$ is smaller than ϵ, denoted as $\|\frac{\mathrm{d}\hat{s}}{\mathrm{d}t}\|_\infty \leq \epsilon$. The complete algorithm is shown in Figure 4.

While Euler's method is easy to understand and to implement, it does not give very strong approximation guarantees. A better alternative that is frequently used is the family of Runge-Kutta methods. The most prominent member is the classical Runge-Kutta method RK4 that guarantees an approximation error in the order of $O(\delta^4)$. In practice, this means that if we halve the step size δ (double the number of iterations), we usually decrease the approximation error by a factor of 16 (Polyanin and Zaitsev 2017). Since the derivatives of the quadratic energy model can never become larger than 1, using RK4 with constant step size 0.01 should be safe. If we want to make sure that the step size is sufficiently small, we can run the algorithm until termination and then repeat with a smaller step size like 0.005 and check that the final values remain unchanged up to the desired accuracy.

4 Convergence and Complexity

The quadratic energy model $\sigma^{\mathbf{A}}(t)$ uniquely defines the evolution of strength values for every BAG $\mathbf{A}$ over time. Our hope is that $\sigma^{\mathbf{A}}(t)$ converges to an equilibrium state as $t \to \infty$. This allows us to define strength values by means of

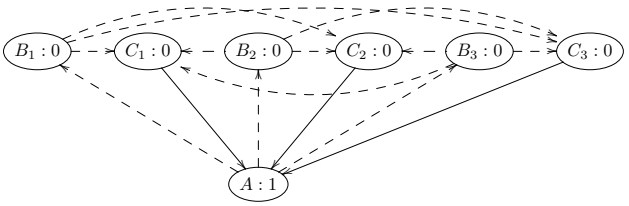


Figure 5: Cycle(3) graph.

the equilibrium $s^* = \lim_{t\to\infty} \sigma^{\mathbf{A}}(t)$. Unfortunately, convergence of $\sigma^{\mathbf{A}}(t)$ for general BAGs with cycles is currently an open question. One can show convergence for some special cases. For example, if cycles contain only support relations, the strength values must be monotonically increasing. They are also bounded from above by 1 due to the nature of the differential equations, and so they must eventually converge. For cycles with only attacks, things are already less straightforward because the attraction force of the initial weight may become stronger than the attacking force as the attackers become weaker. This may let the strength oscillate between the initial weight and the first lower peak reached. However, since arguments' strength can never exceed the initial weight if there is no support, the amplitude of the oscillations must eventually go to 0 and the strength values will converge.

If cycles contain both support and attack relations, the strength values may oscillate more radically. In order to illustrate this, Figure 5 shows a BAG from a family that we call Cycle(k). Each member contains one argument A with initial weight 1 that supports k arguments B_i with weight 0. Each B_i in turn supports the same k arguments C_i that have initial weight 0 as well. Finally, each C_i attacks A. Figure 5 shows Cycle(3) and figure 6 shows the long-term behaviour of the quadratic energy model for Cycle(3) at the top and for Cycle(10) at the bottom. As we may expect, the oscillations for Cycle(k) take more time as we increase k. However, the amplitude decreases and the strength values eventually converge. It is currently unclear if there exist BAGs where the strength values oscillate for all time. However, experiments in (Potyka 2018a) with 3,000 randomly generated BAGs with thousands of nodes and ten thousands of edges demonstrate that the quadratic energy model converges for many cyclic BAGs.

How can we deal with potential divergence in practice? Since equilibrium states can usually be computed in seconds, it is pragmatic to set a time limit for the quadratic energy model. Say, if the model did not converge after 30 seconds, the algorithm stops. Arguments whose strength value has not converged yet can then be detected automatically because we must have $|\frac{\mathrm{d}s_j}{\mathrm{d}t}| > \epsilon$ at such an argument. The evolution of the strength value can then be plotted like in Figure 3 in order to see whether the strength value diverges, oscillates with decreasing amplitude or just converges slowly for other reasons. In particular, even if the strength diverges, it may oscillate between meaningful bounds. For example, if the strength oscillates between 0.8 and 0.9, we could still infer that the argument is rather strong. Of course, this analysis can also be performed automatically by just storing lower and upper bounds and monitoring the derivatives (oscillations occur when the derivative changes signs repeatedly). However, since no non-convergent example has been found so far, we do not discuss these issues further here.

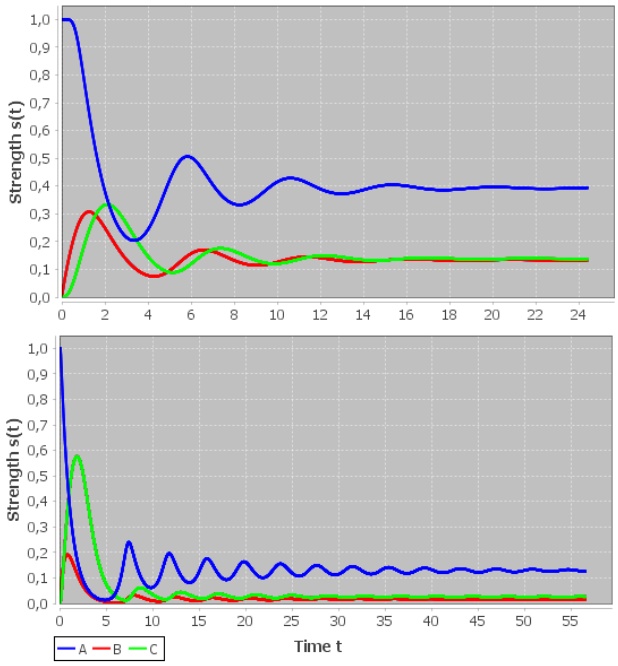


Figure 6: Long-term behaviour of quadratic energy model for Cycle(3) (top) and Cycle(10) (bottom).

Currently, my feeling is that the quadratic energy model always converges. This assumption is based on the idea that the strength of every argument will reach a peak at some point in time. After all arguments have reached their peak, I assume that the amplitude of oscillations will necessarily decrease similar to the observation in Figure 6. Intuitively, the overall energy in the system increases up to one point, but will necessarily decrease after having reached its peak.

On the bright side, while convergence for cyclic BAGs is a difficult question, the quadratic energy model is guaranteed to converge for arbitrary acyclic BAGs. The equilibrium can be computed by numerical methods as discussed before, but can also be computed by a discrete iteration scheme in linear time. The key observation is that arguments' strength depends only on the initial weight and the strength of their parents (attackers and supporters). By evaluating the arguments according to a topological ordering, we can make sure that the final strength of all parents is known in advance and we can compute the final strength values for every argument in a single pass through the graph. This is basically the same mechanism that is used to compute the strength values for discrete models for weighted bipolar argumentation like in (Baroni et al. 2015; Rago et al. 2016; Amgoud and Ben-Naim 2017). We only present the main result here and refer to (Potyka 2018a) for more details and the proof.

Proposition 2 (Equilibria in Acyclic BAGs (Potyka 2018a))**.** *Let* $\mathbf{A}$ *be an acyclic BAG. Then* $\sigma^{\mathbf{A}}$ *converges and the equi-*

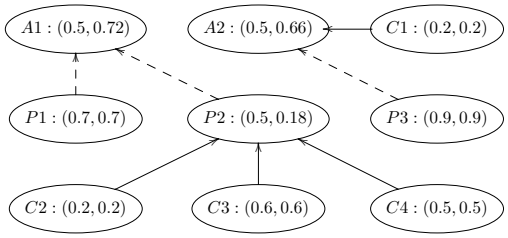


Figure 7: BAG for e-democracy examples. Nodes show (initial weight, final strength) under quadratic energy model.

librium $s^* = \lim_{t\to\infty} \sigma^{\mathcal{A}}(t)$ *can be computed in linear time by the following procedure:*

1. *Compute a topological ordering of the arguments.*
2. *Pick the next argument* i *in the order and set*

$$s_i = \mathrm{w}(i) + (1 - \mathrm{w}(i)) \cdot h(E_i) - \mathrm{w}(i) \cdot h(-E_i),$$

   where E_i *is the energy at* i*.*
3. *Repeat step 2 until all strength values have been computed.*

Example 2. *We illustrate Proposition 2 with an e-democracy problem from (Rago et al. 2016). The question is how to spend a portion of a council's budget. The arguments are divided into decision arguments (prefix A), pro arguments (prefix P) and contra arguments..*

A1: *Build a new cycle path.*

A2: *Repair current infrastructure.*

P1: *Cyclists complain of dangerous roads.*

P2: *A path would enhance the councils green image.*

P3: *Potholes have caused several accidents recently.*

C1: *Significant disruptions to traffic would occur.*

C2: *Environmentalists are a fraction of the population.*

C3: *Recent policies already enhance this green image.*

C4: *Donors do not see the environment as a priority.*

Figure 7 shows the initial weights and final strength values under the quadratic energy model. Since the authors in (Rago et al. 2016) considered only subgraphs of this BAG, I defined additional initial weights for C2, C3, C4. One topological ordering of the arguments is P1, P3, C1, C2, C3, C4, P2, A1, A2. Since only P2, A1, A2 have parents, the energy at all other arguments is 0 *for all time and their final strength is just the initial weight. For P2, the energy is then* $-0.2 - 0.6 - 0.5 = -1.3$ *and the final strength is* $0.5 - 0.5 \cdot h(-(-1.3)) \approx 0.186$*. The energy at A1 is approximately* $0.7 + 0.186 \approx 0.886$ *and the final strength is* $0.5 + 0.5 \cdot h(0.886) \approx 0.719$*. Finally, the energy at A2 is* $0.9 - 0.2 = 0.7$ *and the final strength is* $0.5 - 0.5 \cdot h(0.7) \approx 0.664$*. The continuous evolution of the quadratic energy model is shown in Figure 8. Note that it does indeed converge to the values that we computed.*

Even though we currently cannot give convergence guarantees for cyclic BAGs, the quadratic energy model uniquely defines a strength value for every time point t. In particular,

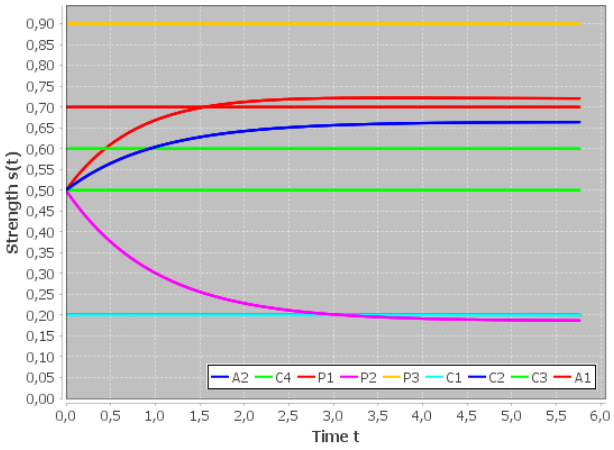


Figure 8: Long-term behaviour of $\sigma^{\mathbf{A}}$ for BAG in Figure 7.

we can analyze the runtime for evaluating the quadratic energy model from time 0 to time T. Using step size δ, this can be done in time $O(\frac{(|\mathcal{A}|+|\mathcal{R}|+|\mathcal{S}|)\cdot T}{\delta})$. The cost is basically composed of the factor $O((\mid \mathcal{A} \mid + \mid \mathcal{R} \mid + \mid \mathcal{S} \mid))$ for the cost of evaluating the differential equations and the factor $O(\frac{T}{\delta})$ for the number of evaluations. While Euler's method needs only a single computation of the differential equations at each time step, more sophisticated methods like RK4 evaluate the differential equations at several points in order to improve the approximation. While this increases the runtime for fixed δ, these methods can usually work with significantly larger step sizes than Euler's method and are therefore more efficient. In our implementation, we let T grow until $\|\frac{\mathrm{d}s}{\mathrm{d}t}\|_\infty < 10^{-4}$. It is reasonable to assume that the point of convergence T^* depends on the size and complexity of cycles. Experiments in (Potyka 2018a) indicate that the overall runtime is bounded from above quadratically by the size of the BAG.

5 From Discrete to Continuous Models

Proposition 2 basically tells us that if the BAG is acyclic, we can transform the continuous quadratic energy model to a discrete model similar to the ones considered in (Baroni et al. 2015; Rago et al. 2016; Amgoud and Ben-Naim 2017). This is interesting from a computational perspective because it gives us a linear runtime guarantee for acyclic BAGs.

On the other hand, continuous models are computationally interesting because they can improve stability in cyclic BAGs as we explain at the end of this section. Therefore, it is natural to ask under what conditions we can transform discrete models like in (Baroni et al. 2015; Rago et al. 2016; Amgoud and Ben-Naim 2017) to well-defined continuous models. A simple sufficient criterion along with some guarantees is given in the following result from (Potyka 2018a).

Proposition 3 (Continuizing Iterative Schemes)**.** *Consider an iterative scheme* $\mathcal{I}$ *that defines the strength values for acyclic BAGs by letting*

$$s_i = f_{\mathcal{I}}(\mathrm{w}(i), \{s_j \mid j \in \mathrm{Att}_i\}, \{s_j \mid j \in \mathrm{Sup}_i\}),$$

where $f_{\mathcal{I}}$ is a function that depends on the initial weight and the strength of attackers and supporters and the strength values s_i are computed in topological order.

1. *If $f_{\mathcal{I}}$ is continuously differentiable with respect to all involved strength values, then for all BAGs $\mathbf{A}$, the system*

$$\frac{\mathrm{d}s_i}{\mathrm{d}t} = f_{\mathcal{I}}(\mathrm{w}(i), \{s_j \mid j \in \mathrm{Att}_i\}, \{s_j \mid j \in \mathrm{Sup}_i\}) - s_i$$

   with initial conditions $s_i(0) = \mathrm{w}(i)$ for $i = 1, \ldots, n$ has a unique solution $\sigma_{\mathcal{I}}^{\mathbf{A}} : \mathbb{R}_0^+ \to \mathbb{R}^n$.
2. *If $\sigma_{\mathcal{I}}^{\mathbf{A}}$ reaches an equilibrium state $s^* = \lim_{t\to\infty} \sigma_{\mathcal{I}}^{\mathbf{A}}(t)$, then $s_i^* = f_{\mathcal{I}}(\mathrm{w}(i), \{s_j^* \mid j \in \mathrm{Att}_i\}, \{s_j^* \mid j \in \mathrm{Sup}_i\})$.*
3. *$\sigma_{\mathcal{I}}^{\mathbf{A}}$ reaches an equilibrium state whenever $\mathbf{A}$ is acyclic.*

Item 1 explains how to transform the definition of a discrete iteration scheme to a system of differential equations and gives a sufficient condition under which the system has a unique solution. We can then use this solution similar to the quadratic energy model as we will illustrate soon.

Item 2 guarantees that if the model reaches an equilibrium state, this state is a fixed point of the discrete update function $f_{\mathcal{I}}$. This implies, in particular, that if the BAG is acyclic, then the continuized model agrees with the discrete model.

Item 3 states that the continuized model is again guaranteed to convergence for acyclic graphs.

Continuizing Discrete Models

Let us now illustrate Proposition 3 by means of the Euler-based restricted semantics that was introduced in (Amgoud and Ben-Naim 2017). As explained before, the Euler-based restricted semantics used the energy $E_i = \sum_{i\in\mathrm{Sup}} s_i - \sum_{i\in\mathrm{Att}} s_i$ before. Given an acyclic BAG, the weights for every argument are then set in topological order by letting

$$s_i = 1 - \frac{1 - \mathrm{w}(i)^2}{1 + \mathrm{w}(i) \cdot \exp(E_i)}. \tag{3}$$

That is, $f_{\mathcal{I}}(\mathrm{w}(i), \{s_j \mid j \in \mathrm{Att}_i\}, \{s_j \mid j \in \mathrm{Sup}_i\}) = 1 - \frac{1-\mathrm{w}(i)^2}{1+\mathrm{w}(i)\cdot\exp(E_i)}$. In order to apply item 1, we have to check that $f_{\mathcal{I}}$ is continuously differentiable with respect to all s_i. Note that E_i is a linear function of the strength values and therefore continuously differentiable. The exponential function $\exp$ is continuously differentiable as well. Therefore, $f_{\mathcal{I}}$ is defined by combining constant and continuously differentiable functions and is therefore itself continuously differentiable (notice, in particular, that the denominator in the fraction in $f_{\mathcal{I}}$ is always greater than 1 because $\exp$ is a positive function). Hence, we can apply Proposition 3.

Item 1 tells us that we obtain the differential equations for s_i by subtracting s_i from $f_{\mathcal{I}}$. Hence, the system that defines the continuous Euler-based semantics is

$$\frac{\mathrm{d}s_i}{\mathrm{d}t} = 1 - \frac{1 - \mathrm{w}(i)^2}{1 + \mathrm{w}(i) \cdot \exp(E_i)} - s_i, \qquad i \in \mathcal{A}. \tag{4}$$

We can now approximate the solution with Euler's method as described in Algorithm 4 or with faster methods like RK4.

Proposition 3 is not always applicable. For example, the update formula for the DF-QuAD algorithm from (Rago et al. 2016) is not continuously differentiable. The formula is also based on some auxiliary functions. We slightly change the notation in order to make the presentation more homogeneous. We define the *geometric energy* at argument j as

$$\mathit{GE}_j = \prod_{i\in\mathrm{Att}_j} (1 - s_i) - \prod_{i\in\mathrm{Sup}_j} (1 - s_i),$$

where we use the convention that the empty product equals 1. Given an acyclic BAG, the DF-QuAD algorithm sets the weights for every argument in topological order by letting

$$s_i = \mathrm{w}(i) + \mathrm{w}(i) \cdot \min\{\mathit{GE}_i, 0\} + (1 - \mathrm{w}(i)) \cdot \max\{\mathit{GE}_i, 0\}.$$

We have $f_{\mathcal{I}}(\mathrm{w}(i), \{s_j \mid j \in \mathrm{Att}_i\}, \{s_j \mid j \in \mathrm{Sup}_i\}) = \mathrm{w}(i) + \mathrm{w}(i) \cdot \min\{\mathit{GE}_i, 0\} + (1 - \mathrm{w}(i)) \cdot \max\{\mathit{GE}_i, 0\}$. The derivative of $f_{\mathcal{I}}$ is discontinuous at 0-energy states. Hence, Proposition 3 is not applicable. However, the conditions in Proposition 3 are sufficient and not necessary. Indeed, one can show in another way that the system for the continuous DF-Quad algorithm has a unique solution (Potyka 2018b).

If Proposition 3 is not applicable, we may also modify the update formula in order to guarantee continuous differentiability. For the DF-Quad algorithm, we could square the strength values in the geometric energy. When replacing the geometric energy with the *squared geometric energy*

$$\mathit{SGE}_j = \prod_{i\in\mathrm{Att}_j} (1 - s_i^2) - \prod_{i\in\mathrm{Sup}_j} (1 - s_i^2),$$

Proposition 3 is applicable. When using the squared geometric energy, an argument with strength 1 will have the same influence as before, but as the strength gets closer to 0 the influence will get gradually weaker. Implementations of all continuizations can be found in *Attractor* that we describe in the final section of this article.

Continuization and Convergence

To get an intuition for why continuizing a discrete model may improve the convergence behaviour in cyclic graphs, it is instructive to look at Euler's method again. Suppose we apply Euler's method with (rather large) step size $\delta = 1$ to the system given in Proposition 3. Then we update each strength value s_i with $s_i \leftarrow s_i + 1 \cdot \frac{\mathrm{d}s_i}{\mathrm{d}t}$ in every iteration. Hence, the update is just

$$\begin{aligned} &s_i + 1 \cdot \big(f_{\mathcal{I}}(\mathrm{w}(i), \{s_j \mid j \in \mathrm{Att}_i\}, \{s_j \mid j \in \mathrm{Sup}_i\}) - s_i\big) \\ =&f_{\mathcal{I}}(\mathrm{w}(i), \{s_j \mid j \in \mathrm{Att}_i\}, \{s_j \mid j \in \mathrm{Sup}_i\}). \end{aligned}$$

That is, we just update all strength values simultaneously with respect to the iterative update formula $f_{\mathcal{I}}$. Hence, applying the discrete iteration scheme can be seen as a very coarse approximation of a continuous system. In the presence of cycles, these coarse steps may lead to divergence even when the continuous model $\sigma_{\mathcal{I}}^{\mathbf{A}}$ converges. Intuitively, this is because a large step size like $\delta = 1$ can let us jump from the graph of the true solution $\sigma_{\mathcal{I}}^{\mathbf{A}}$ to the graph of another solution for different initial conditions. By choosing a smaller step size, we can avoid these jumps and make the procedure more stable. This is basically what we are doing when continuizing $\mathcal{I}$.

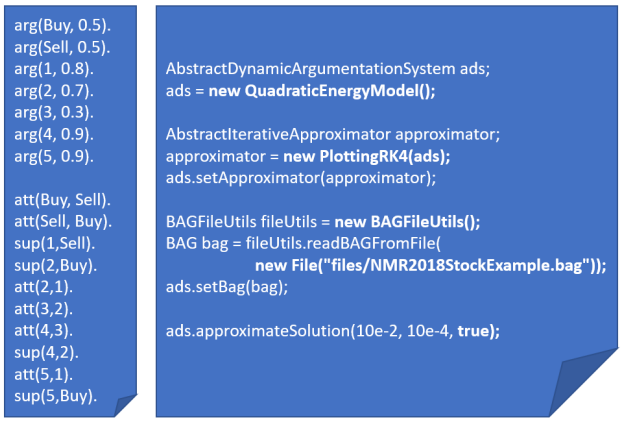


Figure 9: Reading a BAG from a file in *Attractor*.

6 The Java Library Attractor

Finally, we will discuss how the previous ideas can be put into practice using the Java library *Attractor*. A download link for the current version is given in the footnote[1]. The latest code is available at sourceforge[2]. *Attractor* is work in progress and currently provides only a programming interface. However, in the future, a graphical user interface will be added. *Attractor* can be used to

1. compute solutions with existing models,
2. use base classes to implement new models and
3. evaluate continuous models on benchmarks and randomly generated BAGs.

We will discuss each use case in turn.

Computing Solutions

BAGs can be created either programmatically or, more conveniently, by using a file reader. The programming approach is useful when considering families of BAGs like Cycle(k) (c.f. Figure 5 and 6). A code example can be found in */Attractor/src/examples/NMR2018CycleK.java*. In this tutorial, we will focus on the file approach. Figure 9 shows an example file on the left and the code to compute the final strength values with RK4 and to create a plot similar to Figure 3 on the right. Files consist of definitions of arguments, attacks and supports. Argument definitions start with the keyword *arg* and are followed by a name and an optional weight. If no weight is provided in the definition, it is initialized with 0.5 by default. Attack and support definitions start with the keywords *att* and *sup* and are followed by the source and the target of the edge as usual. The file format is inspired by the format used in ConArg[3] (Bistarelli, Rossi, and Santini 2016), but adds optional weights and support relations. The file reader in *Attractor* can also read the current files from the ConArg benchmarks.

In order to compute a solution, we first have to initialize a model that is a subclass of the abstract class *AbstractDynamicArgumentationSystem* that we will explain in the next section. The *AbstractDynamicArgumentationSystem* reference *ads* can also be initialized with implementations of the continuized Euler-based semantics and DF-Quad algorithm that we described in Section 5. By default, RK4 is used to compute solutions. Different algorithms can be selected by using the method *setApproximator*. In the example, we select *PlottingRK4*, which still uses RK4, but simultaneously creates a plot for the evolution of the strength values using JFreeChart[4]. The utility class *BAGFileUtils* is used to read the file and the BAG object is passed to the model. Afterwards, the call of the method *approximateSolution* starts the approximation and will plot graphs like in Figure 3. The first two parameters determine the step size δ and the termination accuracy ϵ. The third parameter is optional and can be used to print the final strength values to the console. However, arguments and their strength values can also be accessed programmatically from the BAG object.

More file and programming examples can be found in *Attractor/files* and *Attractor/examples*. In particular, the code example *NMR2018StockExampleComparison.java* shows how to compute and plot solutions for all models in *Attractor* in order to compare the different semantics.

Implementing new Models and Algorithms

New implementations of continuous models should be derived from the abstract class *AbstractDynamicArgumentationSystem* that can be found in the package *edu.cs.ai.weightedArgumentation.dynamicalSystems*. The package already contains implementations of the quadratic energy model and the continuized models that we discussed before. *AbstractDynamicArgumentationSystem* already provides most of the functionality, the programmer just has to implement the abstract methods *computeDerivativeAt* that basically implements the differential equations and the method *getName* that just returns the name of the model (this is used, for example, when creating plots). Figure 10 shows the implementation of the continuous Euler-based model. The code in Figure 10 is a straightforward translation of the derivatives given in Equation 4 into Java code. As the code demonstrates, preinitialized arrays can be used to access arguments and their supporters and attackers efficiently. All implementations of *AbstractDynamicArgumentationSystem* can be used exactly as demonstrated in Figure 9. In particular, different approximators can be selected. Currently, there are implementations of RK4, Euler's method and a ploting variant of RK4. New algorithms should be derived from the abstract class *AbstractIterativeApproximator* and can then be selected analogously.

Evaluating Models and Algorithms

Utility classes for evaluating models and algorithms can be found in the package *edu.cs.ai.weightedArgumentation.util*. The class *RandomBagGenerator* contains the random generator used for creating the BAGs for the benchmark from

[1] https://www.researchgate.net/publication/326677792_Attractor_v01

[2] https://sourceforge.net/projects/attractorproject/

[3] http://www.dmi.unipg.it/conarg/

[4] http://www.jfree.org/jfreechart/

```
@Override
protected double[] computeDerivativeAt(double[] state) {

  double[] derivatives = new double[state.length];

  for(int i=0; i<arguments.length; i++) {

    double energy = 0;
    for(int s: supporter[i]) {
      energy += state[s];
    }
    for(int a: attacker[i]) {
      energy -= state[a];
    }

    double weight = arguments[i].getInitialWeight();
    double derivative = 1 - (1 - weight*weight)/(1 + weight * Math.exp(energy));
    derivative -= state[i];

    derivatives[i] = derivative;
  }

  return derivatives;
}

@Override
public String getName() {
  return "Continuous Euler-based Model";
}
```

Figure 10: Implementation of Continuous Euler-based Model.

```
RandomBagGenerator generator = new RandomBagGenerator();
BAG bag = generator.createRandomBAG(100);
```

```
RandomBagGenerator generator = new RandomBagGenerator();
generator.createRandomBagFiles(100, 30, 100, "files/Benchmark", "bag");
```

Figure 11: Generating a single random BAG (top) or a batch of random BAGs (bottom) in *Attractor*.

(Potyka 2018a). The original benchmark can be downloaded from the link given in the footnote[5]. Figure 11 shows how to create a single BAG of size 100 at the top and how to create a batch of BAGs of different sizes at the bottom. The arguments for the method *createRandomBagFiles* allow configuring the basic size (100), the number of increments of the size (30) and the number of trials for each size (100). In the example in Figure 11, 100 graphs of size $100, 200, 300, \ldots, 3000$ each will be created and stored in the local directory 'files/Benchmark', each file starting with the prefix 'bag'.

The class *BenchmarkUtils* can be used to run benchmarks and to plot statistics for the evaluation similar to the evaluation in (Potyka 2018a). Figure 12 shows how to run the benchmark files in a directory. The method *runBenchmark* assumes that the given directory contains subdirectories. Each of these subdirectories contains BAGs of a fixed size and the name of the directory is supposed to be the size. The passed model will then be evaluated on all benchmark

[5] https://www.researchgate.net/publication/326557254_Weighted_Bipolar_Argumentation_Benchmark_KR2018

```
BenchmarkUtils benchmark = new BenchmarkUtils();
File benchmarkDirectory = new File("files/networks/barabasi");

QuadraticEnergyModel qas = new QuadraticEnergyModel();
benchmark.runBenchmark(benchmarkDirectory, qas);
```

Figure 12: Running benchmarks in *Attractor*.

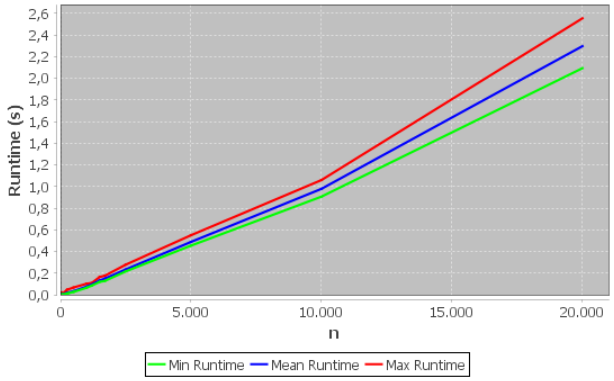


Figure 13: Runtime results for quadratic energy model model on ConArg Barabasi benchmark.

files and the method stores minimum, mean and maximum runtime for all sizes. Runtime results for individual files are printed to the console. The statistics are plotted simultaneously as shown in Figure 13. In this case, we evaluated the quadratic energy model on the Barabasi files from the ConArg benchmark. Other implementations of the base class *AbstractDynamicArgumentationSystem* can be evaluated analogously. Let us note that the ConArg benchmark does not contain weights and supports. By default, all weights will be set to 0.5. In order to evaluate new models on BAGs with supports, the benchmark from (Potyka 2018a) can be used or new benchmarks can be generated using the class *RandomBagGenerator*.

7 Conclusions and Future Work

Continuous dynamical systems are an alternative to discrete models that may give stronger convergence guarantees for cyclic BAGs in the future. Analyzing the general convergence behaviour is difficult and there are no general convergence guarantees currently. However, experiments show that continuous models converge in many cyclic BAGs and do so quickly. There are also interesting relationships between continuous and discrete models. For acyclic BAGs, equilibrium states are guaranteed to exist and can be computed by a discrete iteration scheme. This is computationally advantageous because it gives us a linear runtime guarantee. While continuous models converge superlinearly, they converged subquadratically in all previous experiments and the ability to plot the continuous evolution of strength values may be interesting to improve the explainability of the final strength values even for acyclic graphs. Existing discrete models can

be transformed to continuous iteration schemes and continuous differentiability of the update formula is a sufficient condition for some basic guarantees.

In order to simplify the use of continuous models, *Attractor* provides basic implementations of the ideas discussed here and in (Potyka 2018a). For applications, it allows computing solutions for weighted argumentation problems. For further development, it allows deriving new models and algorithms from base classes that already provide basic functionality. In particular, utility functions can be used to evaluate new models and to compare them to existing models.

One main goal of future work is to advance the understanding of convergence conditions in cyclic BAGs. This involves trying to prove convergence in general cyclic BAGs or finding a counterexample. Furthermore, some empirical studies on the applicability in decision support and the analysis of Twitter discussions similar to the work in (Baroni et al. 2015; Rago et al. 2016; Alsinet et al. 2017) shall be conducted. Developing a graphical user interface for *Attractor* to simplify experiments will also be part of future work.

References

Alsinet, T.; Argelich, J.; Béjar, R.; Fernández, C.; Mateu, C.; and Planes, J. 2017. Weighted argumentation for analysis of discussions in twitter. *International Journal of Approximate Reasoning* 85:21–35.

Amgoud, L., and Ben-Naim, J. 2013. Ranking-based semantics for argumentation frameworks. In *Scalable Uncertainty Management (SUM)*, 134–147. Springer.

Amgoud, L., and Ben-Naim, J. 2017. Evaluation of arguments in weighted bipolar graphs. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, 25–35. Springer.

Amgoud, L.; Cayrol, C.; and Lagasquie-Schiex, M.-C. 2004. On the bipolarity in argumentation frameworks. In *International Workshop on Non-Monotonic Reasoning (NMR)*, volume 4, 1–9.

Baroni, P.; Cerutti, F.; Giacomin, M.; and Guida, G. 2011. Afra: Argumentation framework with recursive attacks. *International Journal of Approximate Reasoning* 52(1):19–37.

Baroni, P.; Romano, M.; Toni, F.; Aurisicchio, M.; and Bertanza, G. 2015. Automatic evaluation of design alternatives with quantitative argumentation. *Argument & Computation* 6(1):24–49.

Barringer, H.; Gabbay, D. M.; and Woods, J. 2012. Temporal, numerical and meta-level dynamics in argumentation networks. *Argument & Computation* 3(2-3):143–202.

Besnard, P., and Hunter, A. 2001. A logic-based theory of deductive arguments. *Artificial Intelligence* 128(1-2):203–235.

Bistarelli, S.; Rossi, F.; and Santini, F. 2016. Conarg: A tool for classical and weighted argumentation. In *COMMA*, 463–464.

Budán, M. C.; Lucero, M. G.; Chesñevar, C.; and Simari, G. R. 2015. Modeling time and valuation in structured argumentation frameworks. *Information Sciences* 290:22–44.

Cayrol, C., and Lagasquie-Schiex, M.-C. 2005. Graduality in argumentation. *Journal of Artificial Intelligence Research (JAIR)* 23:245–297.

Cayrol, C., and Lagasquie-Schiex, M.-C. 2013. Bipolarity in argumentation graphs: Towards a better understanding. *International Journal of Approximate Reasoning* 54(7):876–899.

Cohen, A.; Gottifredi, S.; García, A. J.; and Simari, G. R. 2014. A survey of different approaches to support in argumentation systems. *Knowledge Eng. Review* 29(5):513–550.

Correia, M.; Cruz, J.; and Leite, J. 2014. On the efficient implementation of social abstract argumentation. In *ECAI*, 225–230.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence* 77(2):321–357.

Hunter, A., and Potyka, N. 2017. Updating probabilistic epistemic states in persuasion dialogues. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, 46–56. Springer.

Hunter, A. 2013. A probabilistic approach to modelling uncertain logical arguments. *International Journal of Approximate Reasoning* 54(1):47–81.

Leite, J., and Martins, J. 2011. Social abstract argumentation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 11, 2287–2292.

Mossakowski, T., and Neuhaus, F. 2018. Modular semantics and characteristics for bipolar weighted argumentation graphs. *arXiv preprint arXiv:1807.06685*.

Oren, N., and Norman, T. J. 2008. Semantics for evidence-based argumentation. In *Computational Models of Argument (COMMA)*, volume 172, 276–284. IOS Press.

Polberg, S., and Oren, N. 2014. Revisiting support in abstract argumentation systems. In *Computational Models of Argument (COMMA)*, 369–376.

Polyanin, A. D., and Zaitsev, V. F. 2017. *Handbook of ordinary differential equations*. Chapman and Hall/CRC.

Potyka, N. 2018a. Continuous dynamical systems for weighted bipolar argumentation. In *16th International Conference on Principles of Knowledge Representation and Reasoning, KR (to appear)*.

Potyka, N. 2018b. Convergence and open-mindedness of discrete and continuous semantics for bipolar weighted argumentation (technical report). *arXiv preprint arXiv:1809.07133*.

Rago, A.; Toni, F.; Aurisicchio, M.; and Baroni, P. 2016. Discontinuity-free decision support with quantitative argumentation debates. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 63–73.

Thimm, M. 2012. A probabilistic semantics for abstract argumentation. In *European Conference on Artificial Intelligence (ECAI)*, volume 12, 750–755.

Measuring Disagreement among Knowledge Bases (Extended Version)

Nico Potyka
Institute of Cognitive Science, University of Osnabrück, Germany

Abstract

When combining beliefs from different sources, often not only new knowledge but also conflicts arise. In this paper, we investigate how we can measure the disagreement among sources. We start our investigation with disagreement measures that can be induced from inconsistency measures in an automated way. After discussing some problems with this approach, we propose a new measure that is inspired by the η-inconsistency measure. Roughly speaking, it measures how well we can satisfy all sources simultaneously. We show that the new measure satisfies desirable properties, scales well with respect to the number of sources and illustrate its applicability in inconsistency-tolerant reasoning.

1 Introduction

One challenge in logical reasoning are conflicts between given pieces of information. Therefore, a considerable amount of work has been devoted to repairing inconsistent knowledge bases (Kalyanpur et al. 2006; Lehmann and Bühmann 2010) or performing paraconsistent reasoning (Benferhat, Dubois, and Prade 1997; Arieli, Avron, and Zamansky 2011; Priest 2002). Inconsistency measures (Knight 2002; Grant and Hunter 2006) quantify the degree of inconsistency and help analyzing and resolving conflicts. While work on measuring inconsistency was initially inspired by ideas from repairing knowledge bases and paraconsistent reasoning (Hunter and Konieczny 2005), inconsistency measures also inspired new repair (Thimm 2009; Muiño 2011) and paraconsistent reasoning mechanisms (Potyka 2014; Potyka and Thimm 2015).

Here, we are interested in belief profiles $(\kappa_1, \dots, \kappa_n)$ rather than single knowledge bases κ. Intuitively, we can think of each κ_i as the set of beliefs of an agent. Our goal is then to measure the disagreement among the agents. A natural idea is to reduce measuring disagreement to measuring inconsistency by transforming multiple knowledge bases to a single base using multiset union or conjunction. However, both approaches have some flaws as we will discuss in the following. This observation is similar to the insight that merging belief profiles should be guided by other principles than repairing single knowledge bases (Konieczny 2000). We will therefore propose some new principles for measuring disagreement and introduce a new measure that complies with them.

After explaining the necessary basics in Section 2, we will discuss the relationship between inconsistency measures and disagreement measures in Section 3. To begin with, we will define disagreement measures as functions with two basic properties that seem quite indisputable. We will then show that disagreement measures induced from inconsistency measures by taking the multiset union or conjunction satisfy these basic desiderata and give us some additional guarantees. In Section 4, we will propose some stronger principles for measuring disagreement. One key idea is to allow resolving conflicts by majority decisions. We will show that many measures that are induced from inconsistency measures must necessarily violate some of these principles. In Section 5, we will then introduce a new disagreement measure that is inspired by the η-inconsistency measure from (Knight 2002). Intuitively, it attempts to satisfy all agents' beliefs as well as possible and then measures the average dissatisfaction. We will show that the measure satisfies the principles proposed in Section 4 and some other properties that correspond to principles for measuring inconsistency. To give additional motivation for this work, we will sketch how the measure can be used for belief merging and inconsistency-tolerant reasoning at the end of Section 5. This paper is a slightly extended version of (Potyka 2018) adding proofs and some additional comments on the content and related work.

2 Basics

We consider a propositional logical language $\mathcal{L}$ built up over a finite set $\mathcal{A}$ of propositional atoms using the usual connectives. Satisfaction of formulas $F \in \mathcal{L}$ by valuations $v : \mathcal{A} \to \{0, 1\}$ is defined as usual. A *knowledge base* κ is a non-empty finite multiset over $\mathcal{L}$. $\mathcal{K}$ denotes the set of all knowledge bases. An n-tuple $\mathcal{B} = (\kappa_1, \dots, \kappa_n) \in \mathcal{K}^n$ is called a *belief profile*. We let $\bigsqcup \mathcal{B} = \bigsqcup_{i=1}^n \kappa_i$, where $\sqcup$ denotes multiset union. Note that using multisets is crucial to avoid information loss when several sources contain syntactically equal beliefs. For instance, $\{\neg a\} \sqcup \{a\} \sqcup \{a\} = \{\neg a, a, a\}$. We let $\mathcal{B} \circ \kappa = (\kappa_1, \dots, \kappa_n, \kappa)$, that is, $\mathcal{B} \circ \kappa$ is obtained from $\mathcal{B}$ by adding κ at the end of the profile. Furthermore, we let $\mathcal{B} \circ^1 \kappa = \mathcal{B} \circ \kappa$ and $\mathcal{B} \circ^k \kappa = \left(\mathcal{B} \circ^{k-1} \kappa\right) \circ \kappa$ for

$k > 1$. That is, $\mathcal{B} \circ^k \kappa$ is obtained from $\mathcal{B}$ by adding k copies of κ. We call a non-contradictory formula f *safe in* κ iff f and κ are built up over distinct variables from $\mathcal{A}$. Intuitively, adding a safe formula to κ cannot introduce any conflicts.

A model of κ is a valuation v that satisfies all $f \in \kappa$. We denote the set of all *models* of κ by $\mathrm{Mod}(\kappa)$. If $\mathrm{Mod}(\kappa) \neq \emptyset$, we call κ consistent and inconsistent otherwise. A *minimal inconsistent (maximal consistent) subset* of κ is a subset of κ that is inconsistent (consistent) and minimal (maximal) with this property. If $\mathrm{Mod}(\kappa) \subseteq \mathrm{Mod}(\kappa')$, we say that κ entails κ' and write $\kappa \models \kappa'$. If $\kappa \models \kappa'$ and $\kappa' \models \kappa$, we call κ and κ' equivalent and write $\kappa \equiv \kappa'$. If $\kappa = \{f\}$ and $\kappa' = \{g\}$ are singletons, we just write $f \models g$ or $f \equiv g$.

An inconsistency measure $\mathcal{I} : \mathcal{K}^n \to \mathbb{R}_0^+$ maps knowledge bases to non-negative degrees of inconsistency. The most basic example is the *drastic measure* that yields 0 if the knowledge base is consistent and 1 otherwise (Hunter and Konieczny 2008). Hence, it basically performs a satisfiability test. There exist various other measures, see (Thimm 2016) for a recent overview. While there is an ongoing debate about what properties an inconsistency measure should satisfy, there is general agreement that it should be *consistent* in the sense that $\mathcal{I}(\kappa) = 0$ if and only if κ is consistent. Hence, the inconsistency value is greater than zero if and only if κ is inconsistent. Various other properties of inconsistency measures have been discussed (Hunter and Konieczny 2008; Besnard 2014; Thimm 2016). We will present some of these later, when talking about corresponding properties of disagreement measures.

3 Induced Disagreement Measures

To begin with, we define disagreement measures as functions over the set of all belief profiles $\bigcup_{n=1}^{\infty} \mathcal{K}^n$ that satisfy two basic desiderata.

Definition 1 (Disagreement Measure)**.** A *disagreement measure* is a function $\mathcal{D} : \bigcup_{n=1}^{\infty} \mathcal{K}^n \to \mathbb{R}_0^+$ such that for all belief profiles $\mathcal{B} = (\kappa_1, \ldots, \kappa_n)$, we have

1. *Consistency:* $\mathcal{D}(\mathcal{B}) = 0$ iff $\bigsqcup_{i=1}^{n} \kappa_i$ is consistent.
2. *Symmetry:* $\mathcal{D}(\mathcal{B}) = \mathcal{D}(\kappa_{\sigma(1)}, \ldots, \kappa_{\sigma(n)})$ for each permutation σ of $\{1, \ldots, n\}$.

Consistency generalizes the corresponding property for inconsistency measures. Symmetry assures that the disagreement value is independent of the order in which the knowledge bases are presented. It is similar to *Anonymity* in social choice theory (Zwicker 2016) and guarantees equal treatment of different sources.

Note that each disagreement measure $\mathcal{D}$ induces a *corresponding inconsistency measure* $\mathcal{I}_{\mathcal{D}} : \mathcal{K} \to \mathbb{R}_0^+$ defined by $\mathcal{I}_{\mathcal{D}}(\kappa) = \mathcal{D}(\kappa)$. Conversely, we can induce disagreement measures from inconsistency measures as we discuss next.

$\sqcup$-induced Disagreement Measures

It is easy to see that each inconsistency measure induces a corresponding disagreement measure by taking the multiset union of knowledge bases in the profile.

Proposition 1 ($\sqcup$-induced Measure)**.** *If $\mathcal{I}$ is an inconsistency measure, then the function $\mathcal{D}_{\mathcal{I}}^{\sqcup} : \bigcup_{n=1}^{\infty} \mathcal{K}^n \to \mathbb{R}_0^+$ defined by $\mathcal{D}_{\mathcal{I}}^{\sqcup}(\mathcal{B}) = \mathcal{I}(\bigsqcup \mathcal{B})$ for all $\mathcal{B} \in \mathcal{K}^n$ is a disagreement measure. We call $\mathcal{D}_{\mathcal{I}}^{\sqcup}$ the* measure $\sqcup$-induced by $\mathcal{I}$.

Proof. Non-negativity and Consistency follow immediately from the corresponding properties of inconsistency measures. Symmetry follows from commutativity of $\sqcup$. □

Let us note that each inconsistency measure $\sqcup$-induces a different family of disagreement measures.

Observation 1. *Let $\mathcal{I}_1, \mathcal{I}_2$ be inconsistency measures. If $\mathcal{D}_{\mathcal{I}_1}^{\sqcup} = \mathcal{D}_{\mathcal{I}_2}^{\sqcup}$, then $\mathcal{I}_1 = \mathcal{I}_2$.*

Proof. If $\mathcal{D}_{\mathcal{I}_1}^{\sqcup} = \mathcal{D}_{\mathcal{I}_2}^{\sqcup}$, then we have $\mathcal{I}_1(\kappa) = \mathcal{D}_{\mathcal{I}_1}^{\sqcup}(\kappa) = \mathcal{D}_{\mathcal{I}_2}^{\sqcup}(\kappa) = \mathcal{I}_2(\kappa)$ for all $\kappa \in \mathcal{K}$. Hence, $\mathcal{I}_1 = \mathcal{I}_2$. □

What can we say about the properties of $\sqcup$-induced measures? As we explain first, many properties for inconsistency measures have a natural generalization to disagreement measures that is compatible with $\sqcup$-induced measures in the following sense.

Definition 2 (Corresponding Properties)**.** Let P be a property for inconsistency measures and let P' be a property for disagreement measures. We call (P, P') a *pair of corresponding properties* iff

1. if an inconsistency measure $\mathcal{I}$ satisfies P, then the $\sqcup$-induced measure $\mathcal{D}_{\mathcal{I}}^{\sqcup}$ satisfies P',
2. if a disagreement measure $\mathcal{D}$ satisfies P', then the corresponding inconsistency measure $\mathcal{I}_{\mathcal{D}}$ satisfies P.

One big class of properties for inconsistency measures gives guarantees about the relationship between inconsistency values when we extend the knowledge bases by particular formulas. We start with a general lemma and give some examples in the subsequent proposition.

Lemma 1 (Transfer Lemma)**.** *Let $\sim$ be a binary relation on $\mathbb{R}$ and let $C \subseteq \mathcal{K}^3$ be a ternary constraint on knowledge bases. Given a property for inconsistency measures*

$$\forall \kappa, S, T \in \mathcal{K}, \text{ if } C(\kappa, S, T) \text{ then } \mathcal{I}(\kappa \sqcup S) \sim \mathcal{I}(\kappa \sqcup T), \tag{1}$$

define a property for disagreement measures as follows:

$$\text{For all } \kappa_1, \ldots, \kappa_n, S, T \in \mathcal{K}, \text{ if } C(\bigsqcup_{i=1}^{n} \kappa_i, S, T) \text{ then}$$
$$\mathcal{D}(\kappa_1 \sqcup S, \kappa_2, \ldots, \kappa_n) \sim \mathcal{D}(\kappa_1 \sqcup T, \kappa_2, \ldots, \kappa_n). \tag{2}$$

Then $((1), (2))$ is a pair of corresponding properties.

Proof. First assume that $\mathcal{I}$ satisfies (1). Then if the constraint $C(\bigsqcup_{i=1}^{n} \kappa_i, S, T)$ is satisfied, the definition of $\mathcal{D}_{\mathcal{I}}^{\sqcup}$ and (1) imply that for all $n \in \mathbb{N}$, $\mathcal{D}_{\mathcal{I}}^{\sqcup}(\kappa_1 \sqcup S, \kappa_2, \ldots, \kappa_n) = \mathcal{I}(\kappa_1 \sqcup S \sqcup \bigsqcup_{i=2}^{n} \kappa_i) \sim \mathcal{I}(\kappa_1 \sqcup T \sqcup \bigsqcup_{i=2}^{n} \kappa_i) = \mathcal{D}_{\mathcal{I}}^{\sqcup}(\kappa_1 \sqcup T, \kappa_2, \ldots, \kappa_n)$, where we used commutativity and associativity of $\sqcup$. Hence, $\mathcal{D}_{\mathcal{I}}^{\sqcup}$ satisfies (2).

The second condition from Definition 2 follows from the fact that (1) is just the special case of (2) for $n = 1$. □

Remark 1. The reader may wonder why the corresponding property looks only at the first argument. Note that by symmetry of disagreement measures, the same is true for all other arguments. For instance, we have $\mathrm{Inc}^*(\kappa_1, \kappa_2 \sqcup S) = \mathrm{Inc}^*(\kappa_2 \sqcup S, \kappa_1) \sim \mathrm{Inc}^*(\kappa_2 \sqcup T, \kappa_1) = \mathrm{Inc}^*(\kappa_1, \kappa_2 \sqcup T)$.

We now apply Lemma 1 to some basic properties for inconsistency measures from (Hunter and Konieczny 2008) and adjunction invariance from (Besnard 2014) that will play an important role later.

Proposition 2. *The following are pairs of corresponding properties for inconsistency and disagreement measures:*

- Monotony:
  $\mathcal{I}(\kappa) \leq \mathcal{I}(\kappa \sqcup \kappa')$
  $\mathcal{D}(\kappa_1, \kappa_2, \ldots, \kappa_n) \leq \mathcal{D}(\kappa_1 \sqcup \kappa', \kappa_2, \ldots, \kappa_n)$
- Dominance: *For $f, g \in \mathcal{L}$ such that $f \models g$ and $f \not\models \bot$,*
  $\mathcal{I}(\kappa \sqcup \{f\}) \geq \mathcal{I}(\kappa \sqcup \{g\})$
  $\mathcal{D}(\kappa \sqcup \{f\}, \kappa_2, \ldots, \kappa_n) \geq \mathcal{D}(\kappa \sqcup \{g\}, \kappa_2, \ldots, \kappa_n)$
- Safe Formula Independence: *If $f \in \mathcal{L}$ is safe in κ, then*
  $\mathcal{I}(\kappa \sqcup \{f\}) = \mathcal{I}(\kappa)$
  If $f \in \mathcal{L}$ is safe in $\bigsqcup_{i=1}^n \kappa_i$, then
  $\mathcal{D}(\kappa_1 \sqcup \{f\}, \kappa_2, \ldots, \kappa_n) = \mathcal{D}(\kappa_1, \kappa_2, \ldots, \kappa_n)$
- Adjunction Invariance: *For all $f, g \in \mathcal{L}$,*
  $\mathcal{I}(\kappa \cup \{f, g\}) = \mathcal{I}(\kappa \cup \{f \wedge g\})$
  $\mathcal{D}(\kappa_1 \cup \{f, g\}, \kappa_2, \ldots) = \mathcal{D}(\kappa_1 \cup \{f \wedge g\}, \kappa_2, \ldots)$

Proof. All results follow from Lemma 1.

- *Monotony:* R is $\leq$ and $C(\kappa^*, S, T)$ is true if $S = \emptyset$.
- *Dominance:* R is $\geq$ and $C(\kappa^*, S, T)$ is true if $S = \{f\}$ and $T = \{g\}$ are singletons such that $f \models g$ and $f \not\models \bot$.
- *Safe Formula Independence:* R is $=$ and $C(\kappa^*, S, T)$ is true if $S = \{f\}$ is a singleton, $T = \emptyset$ and f is safe for κ^*.
- *Adjunction Invariance:* R is $=$ and $C(\kappa^*, S, T)$ is true if $S = \{f, g\}$ contains two formulas and $T = \{f \wedge g\}$ contains their conjunction.

□

Monotony demands that adding knowledge can never decrease the disagreement value. Dominance says that replacing a claim with a (possibly weaker) implication of the original claim can never increase the disagreement value. Safe Formula Independence demands that a safe formula does not affect the disagreement value. Adjunction invariance says that it makes no difference whether two pieces of information are presented independently or as a single formula.

Example 1. *The inconsistency measure $\mathcal{I}_{LP_m}$ that was discussed in (Hunter and Konieczny 2010) satisfies Monotony, Dominance, Safe Formula Independence and Adjunction Invariance. From Proposition 2, we can conclude that the $\sqcup$-induced disagreement measure $\mathrm{Inc}^{\sqcup}_{\mathrm{LP_m}}$ satisfies the corresponding properties for disagreement measures.*

What we can take from our discussion so far is that each inconsistency measure induces a disagreement measure with similar properties. As it turns out, each $\sqcup$-induced disagreement measure satisfies an additional property and, in fact, only the $\sqcup$-induced measures do. We call this property *partition invariance*. Intuitively, partition invariance means that the disagreement value depends only on the pieces of information in the belief profile and is independent of the distribution of these pieces. In the following proposition, a partition of a multiset M is a sequence of non-empty multisets $M_1, \ldots, M_k$ such that $\bigsqcup_{i=1}^k M_i = M$.

Proposition 3 (Characterizations of Induced Families)**.** *The following statements are equivalent:*

1. *$\mathcal{D}$ is $\sqcup$-induced by an inconsistency measure.*
2. *$\mathcal{D}$ is $\sqcup$-induced by $\mathcal{I}_\mathcal{D}$.*
3. *$\mathcal{D}$ is* partition invariant*, that is, for all $\kappa \in \mathcal{K}$ and for all partitions $\bigsqcup_{i=1}^{n_1} P_i = \bigsqcup_{i=1}^{n_2} P'_i = \kappa$ of κ, we have that $\mathcal{D}(P_1, \ldots, P_{n_1}) = \mathcal{D}(P'_1, \ldots, P'_{n_2})$.*

Proof. 1 implies 2 because if $\mathcal{D}$ is $\sqcup$-induced by an inconsistency measure $\mathcal{I}$, we have $\mathcal{I}_\mathcal{D}(\kappa) = \mathcal{D}(\kappa) = \mathcal{I}(\kappa)$ for all $\kappa \in \mathcal{K}$, i.e., $\mathcal{I} = \mathcal{I}_\mathcal{D}$.

2 implies 3 because if $\mathcal{D}$ is induced by $\mathcal{I}_\mathcal{D}$, we have $\mathcal{D}(P_1, \ldots, P_{n_1}) = \mathcal{I}_\mathcal{D}(\bigsqcup_{i=1}^{n_1} P_i) = \mathcal{I}_\mathcal{D}(\kappa) = \mathcal{I}_\mathcal{D}(\bigsqcup_{i=1}^{n_2} P'_i) = \mathcal{D}(P'_1, \ldots, P'_{n_2})$, so $\mathcal{D}$ is partition invariant.

Finally, 3 implies 1 because if $\mathcal{D}$ is partition invariant, then for all $\kappa_1, \ldots, \kappa_m$, we have $\mathcal{D}(\kappa_1, \ldots, \kappa_m) = \mathcal{D}(\bigsqcup_{i=1}^m \kappa_i) = \mathcal{I}_\mathcal{D}(\bigsqcup_{i=1}^m \kappa_i)$ because $\kappa_1, \ldots, \kappa_m$ and $\bigsqcup_{i=1}^m \kappa_i$ are partitions of $\kappa = \bigsqcup_{i=1}^m \kappa_i$. Hence, $\mathcal{D}$ is $\sqcup$-induced by the inconsistency measure $\mathcal{I}_\mathcal{D}$. □

So the $\sqcup$-induced disagreement measures are exactly the partition invariant measures. However, partition variance can be undesirable in some scenarios.

Example 2. *Consider the political goals 'increase wealth of households' (h), 'increase wealth of firms' (f), 'increase wages' (w). Suppose there are three political parties whose positions we represent in the profile*

$$\mathcal{B} = (\{f, w, f \to w\}, \{w, h, w \to h\}, \{f, \neg w, w \to \neg f\}).$$

In this scenario, the parties only disagree about w. We modify $\mathcal{B}$ by moving $w \to \neg f$ from the third to the second party:

$$\mathcal{B}' = (\{f, w, f \to w\}, \{w, h, w \to h, w \to \neg f\}, \{f, \neg w\}).$$

The conflict with respect to w remains, but party 2's positions now imply $\neg f$. Since we now have an additional conflict with respect to f, we would expect $\mathcal{D}(\mathcal{B}) < \mathcal{D}(\mathcal{B}')$.

Partition invariant measures are unable to detect the difference in Example 2. Since partition invariance is an inherent property of $\sqcup$-induced measures, we should also investigate non-$\sqcup$-induced measures.

$\wedge$-induced disagreement Measures

Instead of taking the multiset union of all knowledge bases in the profile, we can also just replace each knowledge base with the conjunction of the formulas that it contains in order to induce a disagreement measure.

Proposition 4 ($\wedge$-induced Measure)**.** *If $\mathcal{I}$ is an inconsistency measure, then $\mathcal{D}^\wedge_\mathcal{I} : \bigcup_{n=1}^\infty \mathcal{K}^n \to \mathbb{R}^+_0$ defined by $\mathcal{D}^\wedge_\mathcal{I}(\mathcal{B}) = \mathcal{I}(\bigsqcup_{\kappa \in \mathcal{B}} \{\bigwedge_{F \in \kappa} F\})$ for $\mathcal{B} \in \mathcal{K}^n$ is a disagreement measure. We call $\mathcal{D}^\wedge_\mathcal{I}$ the* measure $\wedge$-induced by $\mathcal{I}$.

Proof. Non-negativity follows immediately from Non-negativity of $\mathcal{I}$. Consistency follows from observing that $\bigsqcup \mathcal{B}$ is consistent if and only if $\bigsqcup_{\kappa \in \mathcal{B}} \{\bigwedge_{F \in \kappa} F\}$ is. Symmetry follows from commutativity of $\sqcup$. □

By repeated application of adjunction invariance (c.f. Proposition 2), one can show that each adjunction invariant inconsistency measure satisfies $\mathcal{I}(\kappa) = \mathcal{I}(\{\bigwedge_{f \in \kappa} f\})$, see (Besnard 2014), Proposition 9. We can use this result to show that for adjunction invariant inconsistency measures, the $\wedge$-induced and the $\sqcup$-induced measures are equal.

Corollary 1. *If $\mathcal{I}$ is an adjunction invariant inconsistency measure, then $\mathcal{D}^{\wedge}_{\mathcal{I}} = \mathcal{D}^{\sqcup}_{\mathcal{I}}$.*

Proof. Consider an arbitrary profile $\mathcal{B} = (\kappa_1, \ldots, \kappa_n)$ and assume that $\mathcal{I}$ is adjunction invariant. Applying $\mathcal{I}(\kappa) = \mathcal{I}(\{\bigwedge_{f \in \kappa} f\})$ from (Besnard 2014), Proposition 9, repeatedly yields $\mathcal{D}^{\wedge}_{\mathcal{I}}(\mathcal{B}) = \mathcal{I}(\bigsqcup_{i=1}^{n} \{\bigwedge_{f \in \kappa_i} f\}) = \mathcal{I}(\{\bigwedge_{i=1}^{n} \bigwedge_{f \in \kappa_i} f\}) = \mathcal{I}(\bigsqcup_{i=1}^{n} \bigsqcup_{f \in \kappa_i} \{f\}) = \mathcal{I}(\bigsqcup_{i=1}^{n} \kappa_i) = \mathcal{I}(\bigsqcup \mathcal{B}) = \mathcal{D}^{\sqcup}_{\mathcal{I}}(\mathcal{B})$. □

This is actually the only case in which the $\wedge$-induced measure can be $\bigsqcup$-induced.

Proposition 5. *Let $\mathcal{I}$ be an inconsistency measure. $\mathcal{D}^{\wedge}_{\mathcal{I}}$ is $\sqcup$-induced if and only if $\mathcal{I}$ is adjunction invariant.*

Proof. If $\mathcal{I}$ is adjunction invariant, $\mathcal{D}^{\wedge}_{\mathcal{I}} = \mathcal{D}^{\sqcup}_{\mathcal{I}}$ according to Corollary 1. Hence, $\mathcal{D}^{\wedge}_{\mathcal{I}}$ is $\sqcup$-induced.

Conversely, if $\mathcal{I}$ is not adjunction invariant, then we have $\mathcal{I}(\kappa \cup \{g_1, g_2\}) \neq \mathcal{I}(\kappa \cup \{g_1 \wedge g_2\})$ for some knowledge base κ and formulas g_1, g_2. Let $\kappa = \{f_1, \ldots, f_n\}$. Then $\mathcal{D}^{\wedge}_{\mathcal{I}}(\{f_1\}, \ldots, \{f_n\}, \{g_1\}, \{g_2\}) = \mathcal{I}(\{f_1, \ldots, f_n, g_1, g_2\}) = \mathcal{I}(\kappa \cup \{g_1, g_2\}) \neq \mathcal{I}(\kappa \cup \{g_1 \wedge g_2\}) = \mathcal{I}(\{f_1, \ldots, f_n, g_1 \wedge g_2\}) = \mathcal{D}^{\wedge}_{\mathcal{I}}(\{f_1\}, \ldots, \{f_n\}, \{g_1, g_2\})$ (note that the conjunction of a one-elementary knowledge bases is just the single formula that it contains). Hence, $\mathcal{D}^{\wedge}_{\mathcal{I}}$ violates partition invariance. Therefore it cannot be $\sqcup$-induced according to Proposition 3. □

The $\sqcup$-induced disagreement measures are characterized by partition invariance. Adjunction invariance plays a similar role for $\wedge$-induced measures.

Proposition 6. *For each inconsistency measure $\mathcal{I}$, $\mathcal{D}^{\wedge}_{\mathcal{I}}$ satisfies adjunction invariance.*

Proof. We have $\mathcal{D}^{\wedge}_{\mathcal{I}}(\kappa_1 \cup \{g_1, g_2\}, \kappa_2, \ldots, \kappa_n) = \mathcal{I}(\{\bigwedge_{f \in \kappa_1 \cup \{g_1, g_2\}} f, \bigwedge_{f \in \kappa_2} f, \ldots, \bigwedge_{f \in \kappa_n} f\}) = \mathcal{I}(\{\big(\bigwedge_{f \in \kappa_1} f\big) \wedge \big(g_1 \wedge g_2\big), \bigwedge_{f \in \kappa_2} f, \ldots, \bigwedge_{f \in \kappa_n} f\}) = \mathcal{D}^{\wedge}_{\mathcal{I}}(\kappa_1 \cup \{g_1 \wedge g_2\}, \kappa_2, \ldots, \kappa_n)$, where the second equality holds because of associativity of $\wedge$. □

Note that the inconsistency measure $\mathcal{I}_{\mathcal{D}^{\wedge}_{\mathcal{I}}}$ induced by $\mathcal{D}^{\wedge}_{\mathcal{I}}$ will also be adjunction invariant. Therefore, $\mathcal{I}_{\mathcal{D}^{\wedge}_{\mathcal{I}}} \neq \mathcal{I}$ if $\mathcal{I}$ is not adjunction invariant. In particular, $\mathcal{D}^{\wedge}_{\mathcal{I}}$ can be a rather coarse measure if $\mathcal{I}$ is not adjunction invariant.

Example 3. *The inconsistency measure $\mathcal{I}_{MI}$ from (Hunter and Konieczny 2010) counts the number of minimal inconsistent sets of a knowledge base. $\mathcal{I}_{MI}$ is not adjunction invariant. For instance, $\mathcal{I}_{MI}(\{a, \neg a, a \wedge b\}) = 2$ because $\{a, \neg a\}$ and $\{\neg a, a \wedge b\}$ are the only minimal inconsistent sets. However, $\mathcal{I}_{MI}(\{a \wedge \neg a \wedge a \wedge b\}) = 1$ because the only minimal inconsistent set is the knowledge base itself. Furthermore, we will have $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}(\kappa) = 1$ whenever $\bigwedge_{f \in \kappa} f$ is inconsistent and $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}(\kappa) = 0$ otherwise. Hence, the inconsistency measure corresponding to $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}$ is the drastic measure.*

Proposition 6 tells us that $\wedge$-induced measures are necessarily adjunction invariant. Whether or not each adjunction invariant disagreement measure is $\wedge$-induced is currently an open question. However, we have the following result.

Proposition 7. *If $\mathcal{D}$ satisfies adjunction invariance and*

$$\mathcal{D}(\{f_1\}, \ldots, \{f_n\}) = \mathcal{D}(\bigsqcup_{i=1}^{n} \{f_i\}), \tag{3}$$

then $\mathcal{D}$ is $\wedge$-induced by an inconsistency measure.

Proof. Adjunction invariance implies

$$\mathcal{D}(\kappa_1, \ldots, \kappa_n) = \mathcal{D}(\{\bigwedge_{f \in \kappa_1} f\}, \ldots, \{\bigwedge_{f \in \kappa_n} f\}).$$

and singleton union invariance implies

$$\mathcal{D}(\{\bigwedge_{f \in \kappa_1} f\}, \ldots, \{\bigwedge_{f \in \kappa_n} f\}) = \mathcal{D}(\bigsqcup_{i=1}^{n} \{\bigwedge_{f \in \kappa_i} f\}).$$

For the disagreement measure $\mathcal{D}^{\wedge}_{\mathcal{I}_{\mathcal{D}}}$ induced by the inconsistency measure $\mathcal{I}_{\mathcal{D}}$ induced by $\mathcal{D}$, we have $\mathcal{D}^{\wedge}_{\mathcal{I}_{\mathcal{D}}}(\kappa_1, \ldots, \kappa_n) = \mathcal{I}_{\mathcal{D}}(\bigsqcup_{i=1}^{n} \{\bigwedge_{f \in \kappa_i} f\}) = \mathcal{D}(\bigsqcup_{i=1}^{n} \{\bigwedge_{f \in \kappa_i} f\})$. Hence, $\mathcal{D}$ is $\wedge$-induced by the inconsistency measure $\mathcal{I}_{\mathcal{D}}$. □

We call property (3) *singleton union invariance* in the following. While adjunction invariance and singleton union invariance are sufficient for being $\wedge$-induced, they are no longer necessary as the following example illustrates.

Example 4. *Consider again the inconsistency measure $\mathcal{I}_{MI}$ from (Hunter and Konieczny 2010) that was explained in Example 3. We have $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}(\{a \wedge b\}, \{\neg a \wedge b\}, \{a \wedge \neg b\}) = \mathcal{I}_{MI}(\{a \wedge b, \neg a \wedge b, a \wedge \neg b\}) = 3$ by definition of the $\wedge$-induced measure. However, $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}(\{a \wedge b, \neg a \wedge b, a \wedge \neg b\}) = \mathcal{I}_{MI}(\{a \wedge b \wedge \neg a \wedge b \wedge a \wedge \neg b\}) = 1$. Hence, $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}$ is not singleton union invariant.*

We close this section by showing that the set of disagreement measures $\sqcup$-induced and $\wedge$-induced from inconsistency measures are neither equal nor disjoint.

To begin with, the $\mathcal{I}_{LP_m}$ inconsistency measure that was discussed in (Hunter and Konieczny 2010) is adjunction invariant. Therefore, $\mathcal{D}^{\sqcup}_{\mathcal{I}_{LP_m}} = \mathcal{D}^{\wedge}_{\mathcal{I}_{LP_m}}$ according to Corollary 1. Hence, the intersection of $\sqcup$-induced and $\wedge$-induced disagreement measures is non-empty.

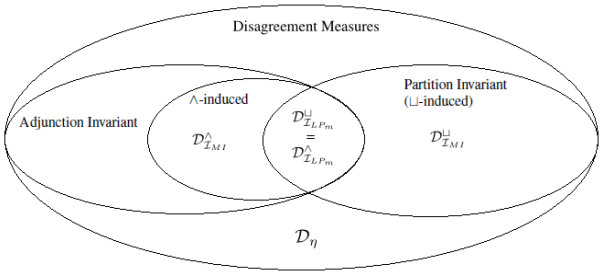


Figure 1: Induced measures and the η-disagreement measure in the space of all disagreement measures.

In order to show that there are partition invariant measures that are not adjunction invariant and vice versa, we use the minimal inconsistent set measure $\mathcal{I}_{MI}$ from (Hunter and Konieczny 2010). As demonstrated in Example 3, $\mathcal{I}_{MI}$ is not adjunction invariant. Therefore, the Transfer Lemma implies that $\mathcal{D}^{\sqcup}_{\mathcal{I}_{MI}}$ is not adjunction invariant either. Hence, $\mathcal{D}^{\sqcup}_{\mathcal{I}_{MI}}$ cannot be $\wedge$-induced according to Proposition 6.

On the other hand, $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}$ is adjunction invariant because each $\wedge$-induced measure is. However, since $\mathcal{I}_{MI}$ is not adjunction invariant, we know from Proposition 5 that $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}$ is not $\sqcup$-induced. Hence, $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}$ is an example of a disagreement measure that is $\wedge$-induced, but not $\sqcup$-induced.

Figure 1 illustrates the relationships between different incompatibility measures. The $\wedge$-induced incompatibility measures are a subset of the adjunction invariant measures (Proposition 6). The fact that all measures in the intersection of partition invariant and adjunction invariant measures are $\wedge$-induced follows from observing that partition invariance implies singleton union invariance (3) and Proposition 7. The η-disagreement measure $\mathcal{D}_\eta$ that is neither partition- nor adjunction-invariant will be discussed in Section 5.

4 Principles for Measuring Disagreement

As illustrated in Figure 1, induced measures correspond to disagreement measures with very specific properties. $\sqcup$-induced measures are necessarily partition invariant. This may be undesirable in certain applications as illustrated in Example 2. If an inconsistency measure is adjunction invariant, the $\wedge$-induced measure will also be partition invariant. If it is not adjunction invariant, the $\wedge$-induced measure will not be partition invariant, but the measure may become rather coarse as illustrated in Example 3. This is some evidence that it is worth investigating non-induced measures. To further distinguish inconsistency from disagreement measures, we will now propose some stronger principles that go beyond our basic desiderata from Definition 1.

To guide our intuition, we think of each knowledge base as the belief set of an agent. We say that κ_i *contradicts* κ_j if $\kappa_i \cup \kappa_j$ is inconsistent. To begin with, let us consider an agent whose beliefs do not contradict any consistent position (its knowledge base is tautological). When adding such an agent to a belief profile, the disagreement value should not increase. Dually, if we add an agent that contradicts every position (its knowledge base is inconsistent), the disagreement value should not decrease. This intuition is captured by the following principles.

Tautology Let $\mathcal{B} \in \mathcal{K}^n$ and let $\kappa_\top \in \mathcal{K}$ be tautological. Then $\mathcal{D}(\mathcal{B} \circ \kappa_\top) \leq \mathcal{D}(\mathcal{B})$.

Contradiction Let $\mathcal{B} \in \mathcal{K}^n$ and let $\kappa_\bot \in \mathcal{K}$ be contradictory. Then $\mathcal{D}(\mathcal{B} \circ \kappa_\bot) \geq \mathcal{D}(\mathcal{B})$.

Inconsistency measures focus mainly on the existence of conflicts. However, in a multiagent setting, conflicts can often be resolved by majority decisions. Given a belief profile $\mathcal{B} = (\kappa_1, \ldots, \kappa_n) \in \mathcal{K}^n$, we call a subset $C \subseteq \{1, \ldots, n\}$ a *consistent coalition* iff $\bigcup_{i \in C} \kappa_i$ is consistent. We say that κ_j is *involved in a conflict in* $\mathcal{B}$ iff there is a consistent coalition C such that $\kappa_j \cup \bigcup_{i \in C} \kappa_i$ is inconsistent. Our next principle demands that conflicts can be eased by majority decisions.

Majority Let $\mathcal{B} = (\kappa_1, \ldots, \kappa_n) \in \mathcal{K}^n$. If κ_j is consistent and involved in a conflict, then there is a $k \in \mathbb{N}$ such that $\mathcal{D}(\mathcal{B} \circ^k \kappa_j) < \mathcal{D}(\mathcal{B})$.

Intuitively, Majority says that we can decrease the severity of a conflict by giving sufficient support for one of the conflicting positions. It does not matter what position we choose as long as this position is consistent. In future work, one may look at alternative principles based on other methods to make group decisions (Zwicker 2016), but Majority seems to be a natural starting point.

Majority implies that we can strictly decrease the disagreement value by adding copies of one consistent position. However, this does not imply that the disagreement value will vanish. If we keep adding copies, the disagreement value will necessarily decrease but it may converge to a value strictly greater than 0. While one may argue that the limit should be 0 if almost all agents agree, one may also argue that the limit should be bounded from below by a positive constant if an unresolved conflict remains. We therefore do not strengthen majority. Instead, we consider an additional principle that demands that the limit is indeed 0 if the majority agrees on all non-contradictory positions. This intuition is captured by the next principle.

Majority Agreement in the Limit Let $\mathcal{B} \in \mathcal{K}^n$. If M is a $\subset$-maximal consistent subset of $\bigsqcup \mathcal{B}$, then $\lim_{k \to \infty} \mathcal{D}(\mathcal{B} \circ^k M) = 0$.

We close this section with an impossibility result: Monotony and Partition Invariance cannot be satisfied jointly with our majority principles. The reason is that such measures can never decrease when receiving new information as explained in the following proposition.

Proposition 8. *If $\mathcal{D}$ satisfies Monotony and Partition Invariance, then $\mathcal{D}(\mathcal{B} \circ^k \kappa) \geq \mathcal{D}(\mathcal{B})$ for all $\mathcal{B} \in \mathcal{K}^n, \kappa \in \mathcal{K}, k \in \mathbb{N}$.*

Proof. Partition invariance implies $\mathcal{D}((\kappa_1, \ldots, \kappa_n) \circ^k \kappa) = \mathcal{D}(\kappa_1 \sqcup \big(\bigsqcup_{i=1}^k \kappa\big), \ldots, \kappa_n)$. Furthermore, Monotony implies $\mathcal{D}(\kappa_1 \sqcup \big(\bigsqcup_{i=1}^k \kappa\big), \ldots, \kappa_n) \geq \mathcal{D}(\kappa_1, \ldots, \kappa_n)$. Hence, $\mathcal{D}((\kappa_1, \ldots, \kappa_n) \circ^k \kappa) \geq \mathcal{D}(\kappa_1, \ldots, \kappa_n)$. □

The conditions of Proposition 8 are in particular met by several induced measures.

Corollary 2. *Every disagreement measure that is*

- *partition invariant and monotone or*
- $\sqcup$*-induced from a monotone inconsistency measure or*
- $\wedge$*-induced from a monotone and adjunction invariant inconsistency measure*

violates Majority *and* Majority Agreement in the Limit.

Proof. Consider the belief profile $\mathcal{B} = (\{a\}, \{\neg a\})$. By Consistency, we have $\mathcal{D}(\mathcal{B}) > 0$ for all disagreement measures $\mathcal{D}$. Hence, Proposition 8 implies that $\mathcal{D}(\mathcal{B} \circ^k \kappa) \geq \mathcal{D}(\mathcal{B}) > 0$ for all $\kappa \in \mathcal{K}$ and $k \in \mathbb{N}$. This implies the first claim. The second claim follows from the first claim with Proposition 3 and the Transfer Lemma. The third claim follows from the first claim with Proposition 5 and the Transfer Lemma. □

5 The η-disagreement Measure

We now consider a novel disagreement measures inspired by the η-inconsistency measure from (Knight 2002). Roughly speaking, the *η-inconsistency measure* attempts to maximize the probability of all formulas within a knowledge base. By subtracting this probability from 1, we get an inconsistency value. In order to assign probabilities to formulas, we consider probability distributions over the set of all valuations $\Omega = \{v \mid v : \mathcal{A} \to \{0,1\}\}$ of our language. Given a probability distribution $\pi : \Omega \to [0,1]$ ($\sum_{v \in \Omega} \pi(v) = 1$) and a formula $F \in \mathcal{L}$, we let

$$P_\pi(F) = \sum_{v \models F} \pi(v).$$

Intuitively, $P_\pi(F)$ is the probability that F is true with respect to π. The η-inconsistency measure from (Knight 2002) is defined by

$$\mathcal{I}_\eta(\kappa) = 1 - \max\{p \mid \exists \pi : \forall F \in \kappa : P_\pi(F) \geq p\}.$$

This formula describes the intuition that we explained in the beginning. $p^* = \max\{p \mid \exists \pi : \forall F \in \kappa : P_\pi(F) \geq \kappa\}$ is the maximum probability that all formulas in κ can simultaneously take. We will have $p^* = 1$ if and only if κ is consistent (Knight 2002).

Let us first look at the disagreement measures induced by $\mathcal{I}_\eta$. $\mathcal{I}_\eta$ satisfies Monotony (Thimm 2016). Therefore, $\mathcal{D}^{\sqcup}_{\mathcal{I}_\eta}$ will violate our majority principles as explained in Corollary 2. However, $\mathcal{I}_\eta$ is not adjunction invariant (Thimm 2016). Therefore, Proposition 5 implies that $\mathcal{D}^{\sqcup}_{\mathcal{I}_\eta} \neq \mathcal{D}^{\wedge}_{\mathcal{I}_\eta}$. Still, $\mathcal{D}^{\wedge}_{\mathcal{I}_\eta}$ does not satisfy our majority principles either.

Example 5. *Let $\mathcal{B} = (\{a\}, \{\neg a\})$. Since $P_\pi(a) = 1 - P_\pi(\neg a)$, we have for all $n \in \mathbb{N}$*

$$\mathcal{D}^{\wedge}_{\mathcal{I}_\eta}(\{a\}, \{\neg a\}) = \mathcal{I}_\eta(\{a, \neg a\}) = 0.5$$
$$= \mathcal{I}_\eta(\{a, \neg a\} \sqcup \bigsqcup_{i=1}^{n} \{a\}) = \mathcal{D}^{\wedge}_{\mathcal{I}_\eta}((\{a\}, \{\neg a\}) \circ^n \{a\}).$$

However, we can modify the definition of the η-inconsistency measure in order to get a disagreement measure that satisfies our desiderata. If we think of $P_\pi(F)$ as the degree of belief in F, then we should try to find a π such that the beliefs of all agents are satisfied as well as possible. To do so, we can first look at how well π satisfies the beliefs of each agent and then look at how well π satisfies the agents' beliefs overall. To measure satisfaction of one agent's beliefs, we take the minimum of all probabilities assigned to the formulas in the agent's knowledge base. Formally, for all probability distributions π and knowledge bases κ over our language, we let

$$s_\pi(\kappa) = \min\{P_\pi(F) \mid F \in \kappa\}.$$

and call $s_\pi(\kappa)$ the *degree of satisfaction* of κ. In order to measure satisfaction of a belief profile, we take the average degree of satisfaction of the knowledge bases in the profile. Formally, we let for all probability distributions π and belief profiles $\mathcal{B}$

$$S_\pi(\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{\kappa \in \mathcal{B}} s_\pi(\kappa)$$

and call $S(\mathcal{B})$ the *degree of satisfaction* of $\mathcal{B}$. We now define a new disagreement measure. Intuitively, it attempts to maximize the degree of satisfaction of the profile. By subtracting the maximum degree of satisfaction from 1, we get a disagreement value.

Definition 3 (η-Disagreement Measure)**.** The η-Disagreement Measure is defined by

$$\mathcal{D}_\eta(\mathcal{B}) = 1 - \max\{p \mid \exists \pi : S_\pi(\mathcal{B}) = p\}.$$

To begin with, we note that $\mathcal{D}_\eta$ is a disagreement measures as defined in Definition 1 and can be computed by linear programming techniques.

Proposition 9. *$\mathcal{D}_\eta$ is a disagreement measures and can be computed by solving a linear optimization problem.*

Proof. Linear Program: In order to give a linear programming formulation, we introduce some notation. First of all, we assume that the valuations in Ω are ordered in a sequence $(v_1, \ldots, v_k)$, where $k = |\Omega|$. The order can be arbitrary (e.g. lexicographic) and is only needed to represent probability distributions π as vectors. We identify each π with the vector $p_\pi \in \mathbb{R}^k$ whose i-th component is $\pi(v_i)$. For a formula F, let $a_F \in \mathbb{R}^k$ be the vector whose i-th component is 1 if $v_i \models F$ and 0 otherwise. Then $a'_F \, p_\pi = \sum_{v \models F} \pi(v) = P_\pi(F)$, where a' denotes the transpose of a.

In order to compute $\mathcal{D}_\eta$, we introduce n auxiliary variables for the knowledge bases in $\mathcal{B} = (\kappa_1, \ldots, \kappa_n)$ that will take on the minimum probability of the formulas in the base. The linear program looks as follows:

$$\begin{aligned} \max_{(x,\eta) \in \mathbb{R}^{k+n}} \quad & \sum_{i=1}^{n} \frac{1}{|\mathcal{B}|} \eta_i \\ \textit{subject to} \quad & a'_F \, x \geq \eta_i \textit{ for all } i = 1, \ldots, n, F \in \kappa_i \\ & a'_\top \, x = 1, \\ & x, \eta \geq 0, \end{aligned} \tag{4}$$

Note that (4) is always feasible because $(p_\pi, 0)$ is a feasible solution for each probability distribution π. (4) corresponds to $\max\{p \mid \exists \pi : S_\pi(\mathcal{B}) = p\}$ as we explain now. Suppose

first that π^* is an optimal solution of the problem $\max\{p \mid \exists \pi : S_\pi(\mathcal{B}) = p\}$. Let $\eta_i^* = s_{\pi^*}(\kappa_i)$. Then $(p_{\pi^*}, \eta^*) \in \mathbb{R}^{k+n}$ is a feasible solution of (4). Since $\sum_{i=1}^n \frac{1}{|\mathcal{B}|}\eta_i^* = S_\pi(\mathcal{B})$, $\max\{p \mid \exists \pi : S_\pi(\mathcal{B}) = p\}$ is less-than or equal to the optimal value of (4).

Consider now an optimal solution (x^*, η^*) of (4). For all $i = 1, \ldots, n$, there must be some formula $F \in \kappa_i$ such that $a'_F \pi = \eta_i$. For if $a'_F \pi > \eta_i$ for all $F \in \kappa_i$, we could improve (x^*, η^*) by increasing η_i, but this contradicts optimality of (x^*, η^*). Hence, $\eta_i = \min\{P_{\pi_{x^*}}(F) \mid F \in \kappa\} = s_{\pi_{x^*}}(\kappa_i)$, where π_{x^*} denotes the probability distribution corresponding to the probability vector x^* ($\pi_{x^*}(v_j) = x_j^*$). Therefore, the optimal value of (4) is $\sum_{i=1}^n \frac{1}{|\mathcal{B}|} s_{\pi_{x^*}}(\kappa_i) = S_{\pi_{x^*}}(\mathcal{B})$ and the optimal value of (4) must also be less-than or equal to the optimal value of $\max\{p \mid \exists \pi : S_\pi(\mathcal{B}) = p\}$. Hence, both values have to be equal.

Well-definedness: The linear optimization problem is guaranteed to be feasible and bounded from above by 1. Therefore, the theory of linear optimization implies that the maximum exists (Matousek and Gärtner 2007). Hence, the disagreement value with respect to $\mathcal{D}_\eta$ is well-defined.

Consistency: If $\mathcal{D}_\eta(\mathcal{B}) = 0$, then $\max\{p \mid \exists \pi : S_\pi(\mathcal{B}) = p\} = 1$. Hence, there is a π such that $S_\pi(\mathcal{B}) = 1$. But this is only possible if $s_\pi(\kappa) = 1$ for all $\kappa \in \mathcal{B}$. Now $s_\pi(\kappa) = 1$ is only possible if $P_\pi(F) = 1$ for all $F \in \kappa$. Hence, $P_\pi(F) = 1$ for all $F \in \bigsqcup_{\kappa\in\mathcal{B}} \kappa$. But then each valuation v with $\pi(v) > 0$ must satisfy all $F \in \bigsqcup_{\kappa\in\mathcal{B}} \kappa$. Hence, $F \in \bigsqcup_{\kappa\in\mathcal{B}} \kappa$ is consistent.

Conversely, if $\bigsqcup_{\kappa\in\mathcal{B}} \kappa$ is inconsistent, then we can argue as before that $S_\pi(\mathcal{B}) < 1$ for all π. Therefore $\mathcal{D}(\mathcal{B}) > 1 - 1 = 0$.

Symmetry: Symmetry follows from observing that all involved operations are commutative. □

As we show next, $\mathcal{D}_\eta$ is neither $\sqcup$- nor $\wedge$-induced from any inconsistency measure. According to Proposition 3 and Proposition 6, it suffices to show that it is neither partition invariant nor adjunction invariant.

Example 6. *Consider again the belief profiles $\mathcal{B}$ and $\mathcal{B}'$ from Example 2. We have $\mathcal{D}_\eta(\mathcal{B}) \approx 0.33$ and $\mathcal{D}_\eta(\mathcal{B}) \approx 0.44$. As desired, $\mathcal{D}_\eta$ recognizes the increased disagreement in the profile. In particular, $\mathcal{D}_\eta$ is not partition invariant.*

Example 7. *To see that $\mathcal{D}_\eta$ is not adjunction invariant, note that $\mathcal{D}_\eta(\{a, \neg a\}) = 0.5$, whereas $\mathcal{D}_\eta(\{a \wedge \neg a\}) = 1$ (contradictory formulas have probability 0 with respect to each π). Hence, $\mathcal{D}_\eta$ is also not adjunction invariant.*

$\mathcal{D}_\eta$ satisfies our four principles for measuring disagreement as we show next. To begin with, we note that the disagreement value necessarily decreases as the proportion of agreeing agents increases.

Proposition 10. *Let $\mathcal{B} \in \mathcal{K}^n$. If $\mathcal{B}$ contains a consistent coalition of size k, then $\mathcal{D}_\eta(\mathcal{B}) \leq 1 - \frac{k}{n}$.*

Proof. Let $C \subseteq \{1, \ldots, n\}$ be a consistent coalition of size k. Since $\kappa = \bigcup_{i\in C} \kappa_i$ is consistent, there is a model v' of κ. Let π be the probability distribution with $\pi(v') = 1$ and $\pi(v) = 0$ for all other valuations in Ω. Then $s_\pi(\kappa_i) = 1$ for all $i \in C$ and $S_\pi(\mathcal{B}) = \frac{1}{n}\big(\sum_{i\in C} s_\pi(\kappa_i) + \sum_{i\in\{1,\ldots,n\}\setminus C} s_\pi(\kappa_i)\big) \geq \frac{1}{n}\big(\sum_{i\in C} 1\big) = \frac{k}{n}$. Hence, $\mathcal{D}_\eta(\mathcal{B}) \leq 1 - \frac{k}{n}$. □

Proposition 10 implies, in particular, that the disagreement value goes to 0 as the proportion of agreeing agents $\frac{k}{n}$ goes to 1. Therefore, $\mathcal{D}_\eta$ satisfies our majority principles.

Corollary 3. *$\mathcal{D}_\eta$ satisfies* Majority *and* Majority Agreement in the Limit.

Proof. Assume $|\mathcal{B}| = n$. We prove the claim for Majority. The proof for Agreement in the Limit is similar and is therefore left out.

$\mathcal{B} \circ^k \kappa_j$ contains the consistent coalition $\{n+1, \ldots, n+k\}$ of size k. Hence, $\mathcal{D}_\eta(\mathcal{B} \circ^k \kappa_j) \leq 1 - \frac{k}{n+k}$ according to Proposition 10. Since $\frac{k}{n+k} \to 1$ as $k \to \infty$, there must be a natural number K such that $1 - \frac{K}{n+K} < \mathcal{D}_\eta(\mathcal{B})$ (recall that $\mathcal{D}_\eta(\mathcal{B}) > 0$ by assumption of Majority) and therefore $\mathcal{D}_\eta(\mathcal{B} \circ^K \kappa_j) < \mathcal{D}_\eta(\mathcal{B})$ as desired. □

Tautology and Contradiction are also satisfied and can be strengthened slightly.

Proposition 11. *$\mathcal{D}_\eta$ satisfies Tautology and Contradiction. Furthermore,*

- *If $\mathcal{D}_\eta(\mathcal{B}) > 0$, then $\mathcal{D}_\eta(\mathcal{B} \circ \kappa_\top) < \mathcal{D}_\eta(\mathcal{B})$.*
- *If $\mathcal{D}_\eta(\mathcal{B}) < 1$, then $\mathcal{D}_\eta(\mathcal{B} \circ \kappa_\bot) > \mathcal{D}_\eta(\mathcal{B})$.*

Proof. We consider only the case for $\kappa_\top$. The proof for the case $\kappa_\bot$ is similar. Since $\kappa_\top$ is tautological, we have $s_\pi(\kappa_\top) = 1$ for arbitrary probability distributions π and $S_\pi(\mathcal{B} \circ \kappa_\top) = \frac{1}{|\mathcal{B}|+1}\big(\sum_{\kappa\in\mathcal{B}} s_\pi(\kappa) + 1\big) = \frac{|\mathcal{B}|}{|\mathcal{B}|+1} S_\pi(\mathcal{B}) + \frac{1}{|\mathcal{B}|+1} \geq \frac{|\mathcal{B}|}{|\mathcal{B}|+1} S_\pi(\mathcal{B}) + \frac{1}{|\mathcal{B}|+1} S_\pi(\mathcal{B}) = S_\pi(\mathcal{B})$, where we used $\sum_{\kappa\in\mathcal{B}} s_\pi(\kappa) = |\mathcal{B}| \cdot S_\pi(\mathcal{B})$ for the second equality and $1 \geq S_\pi(\mathcal{B})$ for the inequality. Therefore,

$$\begin{aligned}\mathcal{D}_\eta(\mathcal{B}) &= 1 - \max\{p \mid \exists \pi : S_\pi(\mathcal{B}) = p\} \\ &\geq 1 - \max\{p \mid \exists \pi : S_\pi(\mathcal{B} \circ \kappa_\top) = p\} \\ &= \mathcal{D}_\eta(\mathcal{B} \circ \kappa_\top).\end{aligned}$$

This proves Tautology for $\mathcal{D}_\eta$. □

Regarding the properties corresponding to principles for measuring inconsistency from Proposition 2, $\mathcal{D}_\eta$ satisfies all of them except Adjunction Invariance (Example 7).

Proposition 12. *$\mathcal{D}_\eta$ satisfies Monotony, Dominance and Safe Formula Independence.*

Proof. Dominance: Let $\mathcal{B}_1 = (\kappa \sqcup \{f\}, \kappa_2, \ldots, \kappa_n)$ and $\mathcal{B}_2 = (\kappa \sqcup \{g\}, \kappa_2, \ldots, \kappa_n)$ and assume that $f \models g$ and $f \not\models \bot$. Then $v \models g$ whenever $v \models f$ for all valuations v. Therefore $P_\pi(f) \leq P_\pi(g)$ for all probability distributions π. Hence, $S_\pi(\kappa \sqcup \{f\}) \leq S_\pi(\kappa \sqcup \{g\})$ and $S_\pi(\mathcal{B}_1) \leq S_\pi(\mathcal{B}_2)$. This then implies $\mathcal{D}_\eta(\mathcal{B}_1) \geq \mathcal{D}_\eta(\mathcal{B}_2)$.

Safe Formula Independence: Consider a belief profile $\mathcal{B} = (\kappa_1, \kappa_2, \ldots, \kappa_n)$ and assume that $f \in \mathcal{L}$ is safe in

$\bigsqcup_{i=1}^{n} \kappa_i$. Then $\bigsqcup_{i=1}^{n} \kappa_i$ does not contain any atoms appearing in f. Consider an arbitrary probability distribution π. If π assigns probability 0 to all v such that $v \not\models f$, we have $S_\pi(\kappa_1) = S_\pi(\kappa_1 \sqcup \{f\})$. Otherwise, construct π' from π as follows: for all valuations v such that $\pi(v) = 0$ or $v \models f$, we let $\pi'(v) = \pi(v)$. For all remaining valuations v with $\pi(v) > 0$ and $v \not\models f$, we let $\pi'(v) = 0$ and modify π' by adding $\pi(v)$ to $\pi(v')$, where v' is an interpretation obtained from v by interpreting the atoms in f such that $v' \models f$. We will then have $S_\pi(\kappa_1) = S_{\pi'}(\kappa_1)$ and since π' assigns probability 0 to all v such that $v \not\models f$, $S_\pi(\kappa_1) = S_{\pi'}(\kappa_1 \sqcup \{f\})$. Hence, for all probability distributions π, there is a probability distribution π' such that $S_\pi(\kappa_1) = S_{\pi'}(\kappa_1 \sqcup \{f\})$. Therefore, $\mathcal{D}_\eta(\kappa_1 \sqcup \{f\}, \kappa_2, \ldots, \kappa_n) = \mathcal{D}_\eta(\kappa_1, \kappa_2, \ldots, \kappa_n)$.

Monotony: Consider a belief profile $\mathcal{B} = (\kappa_1, \kappa_2, \ldots, \kappa_n)$ and a knowledge base κ'. Then for all probability distributions π, $S_\pi(\kappa) = \min\{P_\pi(F) \mid F \in \kappa\} \geq \min\{P_\pi(F) \mid F \in \kappa \sqcup \kappa'\} = S_\pi(\kappa \sqcup \kappa')$ and therefore $S_\pi(\mathcal{B}) \geq S_\pi(\kappa_1 \sqcup \kappa', \kappa_2, \ldots, \kappa_n)$. Hence, $\mathcal{D}_\eta(\kappa_1, \kappa_2, \ldots, \kappa_n) \leq \mathcal{D}_\eta(\kappa_1 \sqcup \kappa', \kappa_2, \ldots, \kappa_n)$. □

We already know that $\mathcal{D}_\eta$ yields 0 if and only if all knowledge bases in the profile are consistent with each other. In the following proposition, we explain in what cases it takes the maximum value 1.

Proposition 13. *Let $\mathcal{B} \in \mathcal{K}^n$. We have $\mathcal{D}_\eta(\mathcal{B}) = 1$ iff all κ_i contain at least one contradictory formula.*

Proof. If $\mathcal{D}_\eta(\mathcal{B}) = 1$, then $S_\pi(\mathcal{B}) = 0$ for all probability distributions π. Hence, for all $\kappa \in \mathcal{B}$ there must be a $F \in \kappa$ such that $P_\pi(F) = 0$ for all π. Again, this is only possible if F is contradictory. Conversely, if each $\kappa \in \mathcal{B}$ contains a contradictory formula, we can conclude that $S_\pi(\mathcal{B}) = 0$. □

Intuitively, if there is a knowledge base that does not contain any contradictory formulas, then all beliefs of one agent can be partially satisfied and the disagreement value with respect to $\mathcal{D}_\eta$ cannot be 1. So the degree of disagreement can only be maximal if each agent has contradictory beliefs.

In some applications, we may want to restrict to belief profiles with consistent knowledge bases. We can rescale $\mathcal{D}_\eta$ for this purpose. Proposition 10 gives us the following upper bounds on the disagreement value.

Corollary 4. *Let $\mathcal{B} = (\kappa_1, \ldots, \kappa_n)$. If some κ_i is consistent, then $\mathcal{D}_\eta(\mathcal{B}) \leq 1 - \frac{1}{n}$.*

Proof. If κ_i is consistent, $\{i\}$ is a consistent coalition of size 1 and Proposition 10 implies the claim. □

The bound in Corollary 4 is actually tight even if all knowledge bases in the profile are individually consistent as we explain in the following example.

Example 8. *For $n = 2$ agents, we have $\mathcal{D}_\eta(\{a\}, \{\neg a\}) = \frac{1}{2}$. For $n = 3$, we have $\mathcal{D}_\eta(\{a \wedge b\}, \{\neg a \wedge b\}, \{\neg b\}) = \frac{2}{3}$. In general, if we have n satisfiable but pairwise inconsistent ($F_i \wedge F_j \equiv \bot$) formulas $F_1, \ldots, F_n$, then $\mathcal{D}_\eta(\{F_1\}, \ldots, \{F_n\}) = 1 - \frac{1}{n}$.*

Hence, if we want to restrict to consistent knowledge bases, we can renormalize $\mathcal{D}_\eta$ by multiplying by $\frac{n}{n-1}$. The disagreement value will then be maximal whenever all agents have pairwise inconsistent beliefs.

As explained in Proposition 9, computing $\mathcal{D}_\eta(\mathcal{B})$ is a linear optimization problem. Interior-point methods can solve these problems in polynomial time in the number of optimization variables and constraints (Matousek and Gärtner 2007). While the number of optimization variables is exponential in the number of atoms $|\mathcal{A}|$ of our language, the number of constraints is linear in the number of formulas in all knowledge bases in the profile. Roughly speaking, computing $\mathcal{D}_\eta(\mathcal{B})$ is very sensitive to the number of atoms, but scales well with respect to the number of agents. In the language of parameterized complexity theory (Flum and Grohe 2006), computing $\mathcal{D}_\eta(\mathcal{B})$ is fixed-parameter tractable (that is, polynomial if we fix the number of atoms) .

Proposition 14. *Computing $\mathcal{D}_\eta(\mathcal{B})$ is fixed-parameter tractable with parameter $|\mathcal{A}|$.*

Proof. For interior-point methods, the total number of bit operations is bounded by $O(N^3 \cdot L)$ per iteration, where N is the number of optimization variables and L is the maximum bit size of coefficients in the linear program (Matousek and Gärtner 2007). For our linear programs, N is exponential in the number of atoms, whereas L is polynomial in the size of the belief profile (the overall number of formulas in the belief profile). The total number of iterations is bounded by $O(\sqrt{N} \cdot L)$ in the worst-case. The overall worst-case runtime is therefore $O(N^{3.5} \cdot L^2)$, which is exponential in the number of atoms, but polynomial in the size of the belief profile. Strictly speaking, the runtime also depends on the desired accuracy ϵ of the solution since interior-point methods are numerical algorithms. This adds an additional factor of $O(\log \frac{1}{\epsilon})$ to the analysis (Matousek and Gärtner 2007). However, the runtime is also polynomial in the accuracy. If we choose an accuracy of $\epsilon = 10^{-a}$, the additional factor will be in the order of $O(\log 10^a) = O(a)$. Therefore, computing $\mathcal{D}_\eta(\mathcal{B})$ is fixed-parameter tractable with parameter $|\mathcal{A}|$. □

While interior-point methods give us a polynomial worst-case guarantee, they are often outperformed in practice by the simplex algorithm. The simplex algorithm has exponential runtime for some artificial examples, but empirically runs in time linear in the number of optimization variables (exponential in $|\mathcal{A}|$) and quadratic in the number of constraints (quadratic in the overall number of formulas in the belief profile) (Matousek and Gärtner 2007).

In the long-term, our goal is to reason over belief profiles that contain conflicts among agents. While we must leave a detailed discussion for future work, we will now sketch how the η-disagreement measure can be used for this purpose. The optimal solutions of the linear optimization problem corresponding to $\mathcal{D}_\eta$ form a topologically closed and convex set of probability distributions. This allows us to compute lower and upper bound on the probability (or more intuitively, the degree of belief) of formulas with respect to the optimal solutions that minimize disagreement. This is

similar to the probabilistic entailment problem (Hansen and Jaumard 2000), where we compute lower and upper bounds with respect to probability distributions that satisfy probabilistic knowledge bases. If, for a belief profile $\mathcal{B}$, the lower bound of the formula F is l and the upper bound is u, we write $P_{\mathcal{B}}(F) = [l, u]$. If $l = u$, we just write $P_{\mathcal{B}}(F) = l$. We call $P_{\mathcal{B}}$ the *aggregated group belief*.

Example 9. *Suppose we have 100 reviews about a restaurant. While most reviewers agree that the food (f) and the service (s) are good, two reviewers disagree about the interior design (d) of the restaurant. Let us assume that* $\mathcal{B} = (\big(\{\{d, f, s\}, \{\neg d, f, s\}\} \circ^{95} \{\{f, s\}\big) \circ^{3} \{\{\neg f, \neg s\}\})$. *We have* $\mathcal{D}_{\eta}(\mathcal{B}) \approx 0.03$. *Intuitively, the degree of disagreement among agents is low because the majority of agents seem not to care about the interior design. The aggregated group beliefs for the atoms in this example are* $P_{\mathcal{B}}(d) = 0.5$, $P_{\mathcal{B}}(f) = 1$, $P_{\mathcal{B}}(s) = 1$.

We can use $P_{\mathcal{B}}$ to define an entailment relation. For instance, we could say that $\mathcal{B}$ entails F iff the lower bound is strictly greater than 0.5. Then, in Example 9, $P_{\mathcal{B}}$ entails f and s, but neither d nor $\neg d$. We can also use $P_{\mathcal{B}}$ to shift our focus from measuring disagreement among agents to measuring disagreement about formulas. For instance, we could measure the disagreement among formulas by measuring how well we can bound the aggregated beliefs away from 0.5 (the lower bound should be close to 1 or the upper bound close to 0).

6 Related Work

The authors in (Whitworth and Felton 1999) considered the problem of measuring disagreement in limited choice problems, where each agent can choose from a finite set of alternatives. The measures are basically defined by counting the decisions and relating the counts. The authors give intuitive justification for their measures, but do not consider general principles. In order to transfer their approach to our setting, one may identify atomic formulas with alternatives in their framework, but it is not clear how this approach could be extended to knowledge bases that contain complex formulas.

Some other conflict measures have been considered in non-classical frameworks. These measures are often closer to distance measures because they mainly compare how close two quantitative belief representations like probability functions, belief functions or fuzzy membership functions are (Liu 2006; Castiñeira, Cubillo, and Montilla 2010; Jousselme and Maupin 2012). In (Thimm 2014), some compatibility measures for Markov logic networks have been proposed. The measures are normalized and the maximum degree of compatibility can be related to a notion of coherence of Markov logic networks. However, this notion cannot be transferred to classical knowledge bases easily.

As we discussed, measuring disagreement is closely related to measuring inconsistency (Knight 2002; Hunter and Konieczny 2008; Grant and Hunter 2013) and merging knowledge bases (Baral, Kraus, and Minker 1991; Liberatore and Schaerf 1998; Konieczny and Pérez 2011). See (Thimm 2016) and (Konieczny and Pérez 2011) for a survey of inconsistency measures and belief merging approaches, respectively. The principles *Majority* and *Majority Agreement in the Limit* from Section 4 are inspired by *Majority merging operators* that allow that a sufficiently large interest group can determine the merging outcome. The η-disagreement measure is perhaps most closely related to *model-based operators* and DA^2 *operators*, which attempt to minimize some notion of distance between interpretations and the models of the knowledge bases in the profile. Similar ideas have been considered in (Grant and Hunter 2013) for the purpose of measuring inconsistency. In contrast, the η-disagreement measure minimizes a probabilistic degree of dissatisfaction of the belief profile. A discussion of merging probabilistic knowledge bases can be found in (Wilmers 2015).

(Grégoire, Konieczny, and Lagniez 2016) introduced some entailment relations based on consensus in belief profiles. Roughly speaking, a consensus is a subset of all agents' beliefs that is consistent with all agents' individual knowledge bases. As explained in (Grégoire, Konieczny, and Lagniez 2016), this idea goes beyond just considering the maximal consistent subsets of the union of all knowledge bases. Consensus may also be interesting to define other incompatibility measures. We will investigate such measures and relationships between consensus-based entailment relations and entailment relations derived from the η-disagreement measure in future work.

7 Conclusions and Future Work

In this paper, we investigated approaches to measuring disagreement among knowledge bases. In principle, inconsistency measures can be applied for this purpose by transforming belief profiles to single knowledge bases. However, we noticed some problems with this approach. For instance, many measures that are naively induced from inconsistency measures violate Majority and Agreement in the Limit as explained in Corollary 2. Even though this problem does not apply to measures $\wedge$-induced from inconsistency measures that violate adjunction invariance, these induced measures show another problem: they may be unable to notice that a conflict can be resolved by giving up parts of agents' beliefs. For instance, the measures $\mathcal{D}^{\wedge}_{\mathcal{I}_{MI}}$ and $\mathcal{D}^{\wedge}_{\mathcal{I}_{\eta}}$ cannot distinguish the profiles $(\{\{a, b\}, \{\neg a, b\}\})$ and $(\{\{a\}, \{\neg a\}\})$ because $\mathcal{I}_{MI}$ and $\mathcal{I}_{\eta}$ cannot distinguish the knowledge bases $\{a \wedge b, \neg a \wedge b\}$ and $\{a, \neg a\}$.

The η-inconsistency measure $\mathcal{D}_{\eta}$ satisfies our principles for measuring disagreement and some other basic properties that correspond to principles for measuring inconsistency. Since $\mathcal{D}_{\eta}$ can perform satisfiability tests, we cannot expect to compute disagreement values in polynomial time with respect to the number of atoms. However, if our agents argue only about a moderate number of statements (we fix the number of atoms), the worst-case runtime is polynomial with respect to the number of agents.

In the long-term, we are in particular interested in reasoning over belief profiles that contain conflicts. We can use the η-inconsistency measure for this purpose as we sketched at the end of Section 5. However, the aggregated group belief $P_{\mathcal{B}}$ does not behave continuously. For instance, if we grad-

ually increase the support for $\neg s$ in Example 9, $P_{\mathcal{B}}(s)$ will not gradually go to 0, but will jump to an undecided state like 0.5 or will jump to 0 at some point. This is not a principal problem for defining an entailment relation that either says that a formula is entailed or not entailed by a profile. However, a continuous notion of group beliefs would allow us to shift the focus from measuring disagreement among agents to measuring disagreement about statements (logical formulas). We could do so by measuring how well we can bound the aggregated beliefs about the formulas in the profile away from 0.5. However, if $P_{\mathcal{B}}$ does not behave continuously, this approach will give us a rather coarse measure (basically three-valued). Therefore, an interesting question for future research is whether we can modify $\mathcal{D}_\eta$ or design other measures that give us an aggregated group belief with a more continuous behavior.

References

Arieli, O.; Avron, A.; and Zamansky, A. 2011. What is an ideal logic for reasoning with inconsistency?. In *IJCAI 2011*, 706–711.

Baral, C.; Kraus, S.; and Minker, J. 1991. Combining multiple knowledge bases. *IEEE transactions on knowledge and data engineering* 3(2):208–220.

Benferhat, S.; Dubois, D.; and Prade, H. 1997. Some syntactic approaches to the handling of inconsistent knowledge bases: A comparative study part 1: The flat case. *Studia Logica* 58(1):17–45.

Besnard, P. 2014. Revisiting postulates for inconsistency measures. In *European Workshop on Logics in Artificial Intelligence*, 383–396. Springer.

Castiñeira, E. E.; Cubillo, S.; and Montilla, W. 2010. Measuring incompatibility between atanassovs intuitionistic fuzzy sets. *Information Sciences* 180(6):820–833.

Flum, J., and Grohe, M. 2006. *Parameterized complexity theory*. Springer Science & Business Media.

Grant, J., and Hunter, A. 2006. Measuring inconsistency in knowledgebases. *Journal of Intelligent Information Systems* 27(2):159–184.

Grant, J., and Hunter, A. 2013. Distance-based measures of inconsistency. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, 230–241. Springer.

Grégoire, É.; Konieczny, S.; and Lagniez, J. 2016. On consensus extraction. In *IJCAI 2016, New York*, 1095–1101. AAAI Press.

Hansen, P., and Jaumard, B. 2000. Probabilistic satisfiability. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*. Springer. 321–367.

Hunter, A., and Konieczny, S. 2005. Approaches to measuring inconsistent information. In *Inconsistency tolerance*. Springer. 191–236.

Hunter, A., and Konieczny, S. 2008. Measuring inconsistency through minimal inconsistent sets. *KR 2008* 8:358–366.

Hunter, A., and Konieczny, S. 2010. On the measure of conflicts: Shapley inconsistency values. *Artificial Intelligence* 174(14):1007–1026.

Jousselme, A.-L., and Maupin, P. 2012. Distances in evidence theory: Comprehensive survey and generalizations. *International Journal of Approximate Reasoning* 53(2):118–145.

Kalyanpur, A.; Parsia, B.; Sirin, E.; and Cuenca-Grau, B. 2006. Repairing unsatisfiable concepts in owl ontologies. In *European Semantic Web Conference*, 170–184. Springer.

Knight, K. 2002. Measuring inconsistency. *Journal of Philosophical Logic* 31(1):77–98.

Konieczny, S., and Pérez, R. P. 2011. Logic based merging. *Journal of Philosophical Logic* 40(2):239–270.

Konieczny, S. 2000. On the difference between merging knowledge bases and combining them. In *KR*, 135–144.

Lehmann, J., and Bühmann, L. 2010. Ore-a tool for repairing and enriching knowledge bases. In *International Semantic Web Conference*, 177–193. Springer.

Liberatore, P., and Schaerf, M. 1998. Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering* 10(1):76–90.

Liu, W. 2006. Analyzing the degree of conflict among belief functions. *Artificial Intelligence* 170(11):909–924.

Matousek, J., and Gärtner, B. 2007. *Understanding and Using Linear Programming*. Universitext (1979). Springer.

Muiño, D. P. 2011. Measuring and repairing inconsistency in probabilistic knowledge bases. *International Journal of Approximate Reasoning* 52(6):828–840.

Potyka, N., and Thimm, M. 2015. Probabilistic reasoning with inconsistent beliefs using inconsistency measures. In *IJCAI 2015*, 3156–3163.

Potyka, N. 2014. Linear programs for measuring inconsistency in probabilistic logics. In *KR 2014*, 568–577.

Potyka, N. 2018. Measuring disagreement among knowledge bases. In *Scalable Uncertainty Management - 12th International Conference, SUM (to appear)*.

Priest, G. 2002. Paraconsistent logic. In *Handbook of philosophical logic*. Springer. 287–393.

Thimm, M. 2009. Measuring inconsistency in probabilistic knowledge bases. In *UAI 2009*, 530–537. AUAI Press.

Thimm, M. 2014. Coherence and compatibility of markov logic networks. In *ECAI 2014*, 891–896. IOS Press.

Thimm, M. 2016. On the compliance of rationality postulates for inconsistency measures: A more or less complete picture. *KI-Künstliche Intelligenz* 1–9.

Whitworth, B., and Felton, R. 1999. Measuring disagreement in groups facing limited-choice problems. *ACM SIGMIS Database* 30(3-4):22–33.

Wilmers, G. 2015. A foundational approach to generalising the maximum entropy inference process to the multi-agent context. *Entropy* 17(2):594–645.

Zwicker, W. 2016. Introduction to the theory of voting. *Handbook of computational social choice* 23–56.

Exploiting Treewidth for Counting Projected Answer Sets*

Johannes K. Fichte
TU Dresden
International Center for Computational Logic
Fakultät Informatik,
01062 Dresden, Germany

Markus Hecher
TU Wien
Insitute of Logic and Computation
Favoritenstraße 9-11 / E192
1040 Vienna, Austria

Abstract

In this paper, we introduce novel algorithms to solve *projected answer set counting (*#PAS*)*. #PAS asks to count the number of answer sets with respect to a given set of *projected atoms*, where multiple answer sets that are identical when restricted to the projected atoms count as only one projected answer set. Our algorithms exploit small treewidth of the primal graph of the input instance by dynamic programming (DP).

We establish a new algorithm for head-cycle-free programs and lift very recent results from projected model counting to #PAS when the input is restricted to head-cycle-free programs. Further, we show how established DP algorithms for disjunctive answer set programs can be extended to solve #PAS. Our algorithms run in time double exponential for head-cycle-free programs and triple exponential in the treewidth for disjunctive programs and polynomial in the input size of the instance.

Finally, we take the exponential time hypothesis (ETH) into account and establish lower bounds of bounded treewidth algorithms for #PAS. In particular, one can not expect (under ETH) to solve #PAS for head-cycle-free or disjunctive programs in polynomial time in the instance size while being single or double exponential in the treewidth, respectively.

Introduction

Answer Set Programming (ASP) (Brewka, Eiter, and Truszczyński 2011) is an active research area of artificial intelligence. It provides a logic-based declarative modelling language and problem solving framework (Gebser et al. 2012) for hard computational problems, which has been widely applied (Balduccini, Gelfond, and Nogueira 2006; Niemelä, Simons, and Soininen 1999; Nogueira et al. 2001; Guziolowski et al. 2013). In ASP, questions are encoded into rules and constraints that form a disjunctive (logic) program over atoms. Solutions to the program are so-called answer sets. Lately, two computational problems of ASP have received increasing attention, namely, #AS (Fichte et al. 2017a) and #PAS (Aziz 2015). The problem #AS asks to output the number of answer sets of a given disjunctive program. When considering computational complexity #AS can be classified as #·coNP-complete (Fichte et al. 2017a), which is even harder than counting the models of a Boolean formula. A natural abstraction of #AS is to consider projected counting where we ask to count the answer sets of a disjunctive program with respect to a given set of projected atoms (#PAS). Particularly, we consider multiple answer sets that are identical when reduced to the projected atoms as only one solution. Under standard assumptions the problem #PAS is complete for the class $\#\cdot\Sigma_2 P$. However, if we take all atoms as projected, then #PAS is again #·coNP-complete and if there are no projected atoms then it is simply Σ_2^p-complete. But some fragments of ASP have lower complexity. A prominent example is the class of *head-cycle-free* programs (Ben-Eliyahu and Dechter 1994), which requires the absence of cycles in a certain graph representation of the program. There the classical complexity of deciding whether a program has an answer set (or considering #PAS without projected atoms) is simply NP-complete.

A way to solve computationally hard problems is to employ parameterized algorithmics (Cygan et al. 2015), which exploits certain structural restrictions in a given input instance. Because structural properties of an input instance often allow for algorithms that solve problems in polynomial time in the size of the input and exponential time in a measure of the structure, whereas under standard assumptions an efficient algorithm is not possible if we consider only the size of the input. In this paper, we consider the treewidth of a graph representation associated with the given input program as structural restriction, namely the *treewidth of the primal graph* (Jakl, Pichler, and Woltran 2009). Generally speaking, treewidth measures the closeness of a graph to a tree, based on the observation that problems on trees are often easier to solve than on arbitrary graphs.

Our results are as follows:

- We perform the classical complexity analysis of counting projected answer sets (#PAS).
- We establish a novel algorithm that solves ASP problems by exploiting treewidth when the input program is restricted to head-cycle-free programs in runtime single exponential in the treewidth.
- We introduce a framework that allows for counting projected answer sets by exploiting treewidth. Therefore, we lift recent results from projected model counting in the domain of Boolean formulas to counting projected answer

*The work has been supported by Austrian Science Fund (FWF) Grant Y698 and German Science Fund (DFG) Grant HO 1294/11-1. The authors are also affiliated with University of Potsdam, Germany.

sets. We establish algorithms that are (i) double exponential in the treewidth if the input is restricted to head-cycle-free programs and (ii) triple exponential in the treewidth if we allow arbitrary disjunctive programs.

- Using the exponential time hypothesis (ETH) for lower bounds, we establish that (i) projected ASP for head-cycle-free programs *cannot* be solved in time single exponential time in the treewidth and (ii) projected ASP for arbitrary disjunctive programs is *not expected* to be solved in time double exponential in the treewidth.

Related Work. Gebser, Kaufmann, and Schaub (2009) considered projected enumeration for ASP. Aziz (2015) introduced techniques to modify modern ASP-solvers in order to count projected answer sets. Jakl, Pichler, and Woltran (2009) presented DP algorithms that solve ASP counting in time double exponential in the treewidth. Pichler et al. (2014) investigated the complexity of extended programs and also presented DP algorithms for it. We employ ideas from their algorithms to solve head-cycle-free programs. Fichte et al. (2017a; 2017c) presented algorithms to solve ASP counting for the full standard syntax of modern ASP solvers (including minimization). Recently Fichte et al. (2018) gave DP algorithms for projected #SAT and showed lower bounds.

Preliminaries

Basics and Combinatorics. For a set X, let 2^X be the *power set of* X consisting of all subsets Y with $\emptyset \subseteq Y \subseteq X$. Let $\vec{s}$ be a sequence of elements of X. When we address the i-th element of the sequence $\vec{s}$ for a given positive integer i, we simply write $\vec{s}_{(i)}$. The sequence $\vec{s}$ *induces* an *ordering* $<_{\vec{s}}$ on the elements in X by defining the relation $<_{\vec{s}} := \{(\vec{s}_{(i)}, \vec{s}_{(j)}) \mid 1 \leq i < j \leq |\vec{s}|\}$. Given some integer n and a family of finite subsets X_1, X_2, ..., X_n. Then, the generalized combinatorial inclusion-exclusion principle (Graham, Grötschel, and Lovász 1995) states that the number of elements in the union over all subsets is $\left|\bigcup_{j=1}^{n} X_j\right| = \sum_{I \subseteq \{1,\ldots,n\}, I \neq \emptyset} (-1)^{|I|-1} \left|\bigcap_{i \in I} X_i\right|$.

Computational Complexity. We assume familiarity with standard notions in computational complexity (Papadimitriou 1994) and use counting complexity classes as defined by Durand, Hermann, and Kolaitis (2005). For parameterized complexity, we refer to standard texts (Cygan et al. 2015). We recall some basic notions. Let Σ and Σ' be some finite alphabets. We call $I \in \Sigma^*$ an *instance* and $\|I\|$ denotes the size of I. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq \Sigma'^* \times \mathbb{N}$ be two parameterized problems. An *fpt-reduction* r from L to L' is a many-to-one reduction from $\Sigma^* \times \mathbb{N}$ to $\Sigma'^* \times \mathbb{N}$ such that for all $I \in \Sigma^*$ we have $(I, k) \in L$ if and only if $r(I, k) = (I', k') \in L'$ such that $k' \leq g(k)$ for a fixed computable function $g : \mathbb{N} \to \mathbb{N}$, and there is a computable function f and a constant c such that r is computable in time $O(f(k)\|I\|^c)$. If additionally g is linear, then r is referred to as *fpl-reduction*. A *witness function* is a function $\mathcal{W}\colon \Sigma^* \to 2^{\Sigma'^*}$ that maps an instance $I \in \Sigma^*$ to a finite subset of Σ'^*. We call the set $\mathcal{W}(I)$ the *witnesses*. A *parameterized counting problem* $L : \Sigma^* \times \mathbb{N} \to \mathbb{N}_0$ is a function that maps a given instance $I \in \Sigma^*$ and an integer $k \in \mathbb{N}$ to the cardinality of its witnesses $|\mathcal{W}(I)|$. Let $\mathcal{C}$ be a decision complexity class, e.g., P. Then, $\# \cdot \mathcal{C}$ denotes the class of all counting problems whose witness function $\mathcal{W}$ satisfies (i) there is a function $f : \mathbb{N}_0 \to \mathbb{N}_0$ such that for every instance $I \in \Sigma^*$ and every $W \in \mathcal{W}(I)$ we have $|W| \leq f(\|I\|)$ and f is computable in time $\mathcal{O}(\|I\|^c)$ for some constant c and (ii) for every instance $I \in \Sigma^*$ the decision problem $\mathcal{W}(I)$ belongs to the complexity class $\mathcal{C}$. Then, $\# \cdot \mathrm{P}$ is the complexity class consisting of all counting problems associated with decision problems in NP. Let L and L' be counting problems with witness functions $\mathcal{W}$ and $\mathcal{W}'$. A *parsimonious reduction* from L to L' is a polynomial-time reduction $r : \Sigma^* \to \Sigma'^*$ such that for all $I \in \Sigma^*$, we have $|\mathcal{W}(I)| = |\mathcal{W}'(r(I))|$. It is easy to see that the counting complexity classes $\# \cdot \mathcal{C}$ defined above are closed under parsimonious reductions. It is clear for counting problems L and L' that if $L \in \# \cdot \mathcal{C}$ and there is a parsimonious reduction from L' to L, then $L' \in \# \cdot \mathcal{C}$.

Answer Set Programming (ASP). We follow standard definitions of propositional disjunctive ASP. For comprehensive foundations, we refer to introductory literature (Brewka, Eiter, and Truszczyński 2011; Janhunen and Niemelä 2016). Let ℓ, m, n be non-negative integers such that $\ell \leq m \leq n$, a_1, ..., a_n be distinct propositional atoms. Moreover, we refer by *literal* to an atom or the negation thereof (negation as failure). A *program* Π is a finite set of *rules* of the form $a_1 \vee \cdots \vee a_\ell \leftarrow a_{\ell+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n$. For a rule r, we let $H_r := \{a_1, \ldots, a_\ell\}$, $B_r^+ := \{a_{\ell+1}, \ldots, a_m\}$, and $B_r^- := \{a_{m+1}, \ldots, a_n\}$. We denote the sets of *atoms* occurring in a rule r or in a program Π by $\mathrm{at}(r) := H_r \cup B_r^+ \cup B_r^-$ and $\mathrm{at}(\Pi) := \bigcup_{r \in \Pi} \mathrm{at}(r)$. Let Π be a program. A program Π' is a *sub-program of* Π if $\Pi' \subseteq \Pi$. The program Π is *normal*, if $|H_r| \leq 1$ for every $r \in \Pi$. The *positive dependency digraph* D_Π of Π is the directed graph defined on the set of atoms from $\bigcup_{r \in \Pi} H_r \cup B_r^+$, where for every rule $r \in \Pi$ two atoms $a \in B_r^+$ and $b \in H_r$ are joined by an edge (a, b). A head-cycle of D_Π is an $\{a, b\}$-cycle[1] for two distinct atoms $a, b \in H_r$ for some rule $r \in \Pi$. The program Π is head-cycle-free if D_Π contains no head-cycle (Ben-Eliyahu and Dechter 1994).

An *interpretation* I is a set of atoms. I *satisfies* a rule r if $(H_r \cup B_r^-) \cap I \neq \emptyset$ or $B_r^+ \setminus I \neq \emptyset$. I is a *model* of Π if it satisfies all rules of Π, in symbols $I \models \Pi$. The *Gelfond-Lifschitz (GL) reduct* of Π under I is the program Π^I obtained from Π by first removing all rules r with $B_r^- \cap I \neq \emptyset$ and then removing all $\neg z$ where $z \in B_r^-$ from the remaining rules r (Gelfond and Lifschitz 1991). I is an *answer set* of a program Π if I is a minimal model of Π^I. Deciding whether a disjunctive program has an answer set is Σ_2^{P}-complete (Eiter and Gottlob 1995). The problem is called *consistency* of an ASP program. If the input is restricted to normal programs, the complexity drops to NP-complete (Bidoít and Froidevaux 1991; Marek and Truszczyński 1991). A head-cycle-free program Π

[1] Let $G = (V, E)$ be a digraph and $W \subseteq V$. Then, a cycle in G is a W-cycle if it contains all vertices from W.

can be translated into a normal program in polynomial time (Ben-Eliyahu and Dechter 1994). The following well-known characterization of answer sets is often invoked when considering normal programs (Lin and Zhao 2003). Given a model I of a normal program Π and an ordering σ of atoms over I. An atom $a \in I$ is *proven* if there is a rule $r \in \Pi$ with $a \in H_r$ where (i) $B_r^+ \subseteq I$, (ii) $b <_\sigma a$ for every $b \in B_r^+$, and (iii) $I \cap B_r^- = \emptyset$ and $I \cap (H_r \setminus \{a\}) = \emptyset$. Then, I is an *answer set* of Π if (i) I is a model of Π, and (ii) every atom $a \in I$ is proven. This characterization vacuously extends to head-cycle-free programs by applying the results of Ben-Eliyahu and Dechter (1994). Given a program Π, an atom a of any answer set of Π has to occur in some head of a rule of Π (Baral and Gelfond 1994), which we assume in the following.

Example 1. *Consider the following program:*

$$\Pi := \{\overbrace{a \vee b \leftarrow}^{r_1}; \overbrace{c \vee e \leftarrow}^{r_2}; \overbrace{d \vee e \leftarrow b}^{r_3}; \overbrace{b \leftarrow e, \neg d}^{r_4}; \overbrace{d \leftarrow \neg b}^{r_5}\}.$$

It is easy to see that Π is head-cycle-free. The set $A = \{b, c, d\}$ is an answer set of Π. Consider the ordering $\sigma = \langle b, c, d\rangle$, from which we can prove atom b by rule r_1, atom c by rule r_2, and atom d by rule r_3. Further answer sets are $B = \{a, c, d\}$, $C = \{b, e\}$, and $D = \{a, d, e\}$.

Counting Projected Answer Sets. An instance is a pair (Π, P), where Π is a program and P is a set of atoms such that $P \subseteq \text{at}(\Pi)$. We call the set P *projection atoms* of Π. The *projected answer sets count* of Π with respect to P is the number of subsets $I \subseteq P$ such that $I \cup J$ is an answer set of Π for some set $J \subseteq \text{at}(\Pi) \setminus P$. The *counting projected answer sets problem (*#PAS*)* asks to output the projected answer sets count of Π, i.e., $|\{I \cap P \mid I \in S\}|$ where S is the set of all answer sets of Π.

Example 2. *Consider program Π from Example 1 and its four answer sets $\{a, c, d\}$, $\{b, c, d\}$, $\{b, e\}$, and $\{a, d, e\}$, as well as the set $P := \{d, e\}$ of projection atoms. When we project the answer sets to the set P, we only have the three answer sets $\{d\}$, $\{e\}$, and $\{d, e\}$. Hence, while Π has 4 answer sets, the projected answer set count of (Π, P) is 3.*

Proposition 1. *The problem* #PAS *is #·Σ_2P-complete when we allow disjunctive programs as input and #·*NP*-complete when the input is restricted to head-cycle-free programs.*

Proof. Membership immediately holds as we can check for a given set $I \subseteq P$ whether there is an answer set $J \supseteq I$ of Π with $J \cap (P \setminus I) = \emptyset$ by checking if there is an answer set of program $\Pi \cup \bigcup_{i \in I}\{\leftarrow \neg i\} \cup \bigcup_{i \in P \setminus I}\{\leftarrow i\}$. Note that if Π is head-cycle-free, this program is head-cycle-free as well. Hardness follows by establishing a parsimonious reduction from #$\exists$-SAT or #$\exists\forall$-SAT[2], respectively. Assume that the input is restricted to head-cycle-free programs. Given an instance (Q, Z) with $Q = \exists X.\phi(X, Z)$. We reduce to the instance $(R(Q), Z)$ of #PAS, where $R(Q)$ is defined as follows. For each variable $v \in X \cup Z$, we add the rule $v \vee nv \leftarrow$. For each clause $\ell_1 \vee \ldots \vee \ell_k$ in $\phi(X, Z)$, we add a rule $\leftarrow \bar{\ell_1}, \ldots, \bar{\ell_k}$ where $\bar{\ell_i}$ corresponds to x if $\ell_i = \neg x$ for a variable x, and $\neg x$ otherwise. Then, a counter c solves (Q, Z) if and only if c solves $(R(Q), Z)$. Assume that we allow arbitrary disjunctive programs as input. Given an instance (Q, Z), where $Q = \exists X.\forall Y.\phi(X, Y, Z)$. We reduce to the instance $(R(Q'), Z)$ of #PAS, where $Q' = \exists X'.\forall Y.\phi(X, Y, Z)$, $X' = X \cup Z$, and $R(Q')$ is defined exactly as by Eiter and Gottlob (1995). Then, since R is a correct encoding of $\exists\forall$-SAT, the projected model count c of (Q, Z) is the projected answer sets count of $(R(Q'), Z)$ and vice versa. Consequently, the proposition sustains. □

[2]For quantified Boolean formulas (QBF) and its evaluation problem ($Q_1 \ldots Q_i$-SAT for alternating $Q_i \in \{\exists, \forall\}$) we refer to standard texts (Biere et al. 2009; Kleine Büning and Lettman 1999).

Tree Decompositions (TDs)

For basic terminology on graphs and digraphs, we refer to standard texts (Diestel 2012; Bondy and Murty 2008). For a tree $T = (N, A, n)$ with root n and a node $t \in N$, we let $\text{children}(t, T)$ be the sequence of all nodes t' in arbitrarily but fixed order, which have an edge $(t, t') \in A$. Let $G = (V, E)$ be a graph. A *tree decomposition (TD)* of graph G is a pair $\mathcal{T} = (T, \chi)$, where $T = (N, A, n)$ is a rooted tree, $n \in N$ the root, and χ a mapping that assigns to each node $t \in N$ a set $\chi(t) \subseteq V$, called a *bag*, such that the following conditions hold: (i) $V = \bigcup_{t \in N} \chi(t)$ and $E \subseteq \bigcup_{t \in N}\{\{u, v\} \mid u, v \in \chi(t)\}$; and (ii) for each r, s, t, such that s lies on the path from r to t, we have $\chi(r) \cap \chi(t) \subseteq \chi(s)$. Then, $\text{width}(\mathcal{T}) := \max_{t \in N} |\chi(t)| - 1$. The *treewidth* $tw(G)$ of G is the minimum $\text{width}(\mathcal{T})$ over all tree decompositions $\mathcal{T}$ of G. For arbitrary but fixed $w \geq 1$, it is feasible in linear time to decide if a graph has treewidth at most w and, if so, to compute a tree decomposition of width w (Bodlaender 1996). In order to simplify case distinctions in the algorithms, *we always use so-called nice tree decompositions*, which can be computed in linear time without increasing the width (Kloks 1994) and are defined as follows. For a node $t \in N$, we say that $\text{type}(t)$ is *leaf* if $\text{children}(t, T) = \langle\rangle$; *join* if $\text{children}(t, T) = \langle t', t''\rangle$ where $\chi(t) = \chi(t') = \chi(t'') \neq \emptyset$; *int* ("introduce") if $\text{children}(t, T) = \langle t'\rangle$, $\chi(t') \subseteq \chi(t)$ and $|\chi(t)| = |\chi(t')| + 1$; *rem* ("removal") if $\text{children}(t, T) = \langle t'\rangle$, $\chi(t') \supseteq \chi(t)$ and $|\chi(t')| = |\chi(t)| + 1$. If for every node $t \in N$, $\text{type}(t) \in \{\textit{leaf}, \textit{join}, \textit{int}, \textit{rem}\}$ and bags of leaf nodes and the root are empty, then the TD is called *nice*.

Example 3. *Figure 1 illustrates a graph G_1 and a tree decomposition of G_1 of width 2. By a basic property*[3] *of tree decompositions (Kloks 1994), the treewidth of G_1 is 2.*

Dynamic Programming on TDs

In order to use tree decompositions for ASP solving, we need a dedicated graph representation of ASP programs (Jakl, Pichler, and Woltran 2009). The *primal graph* G_Π of program Π has the atoms of Π as vertices and an edge $\{a, b\}$ if there exists a rule $r \in \Pi$ and $a, b \in \text{at}(r)$.

Example 4. *Recall program Π from Example 1 and observe that graph G_1 in Figure 1 is the primal graph of Π.*

[3]The vertices e,b,d that are all neighbors to each other in G_1.

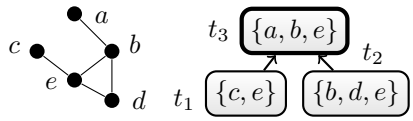


Figure 1: Graph G_1 and a tree decomposition of G_1.

Let $\mathcal{T} = (T, \chi)$ be a tree decomposition of primal graph G_Π of a program Π. Further, let $T = (N, \cdot, n)$ and $t \in N$. The *bag-program* is defined as $\Pi_t := \{r \mid r \in \Pi, \text{at}(r) \subseteq \chi(t)\}$, the *program below* t as $\Pi_{\leq t} := \{r \mid r \in \Pi_{t'}, t' \in \text{post-order}(T, t)\}$, and the *program strictly below* t as $\Pi_{<t} := \Pi_{\leq t} \setminus \Pi_t$. It holds that $\Pi_{\leq n} = \Pi_{<n} = \Pi$ (Fichte et al. 2017a). Analogously, we define the *atoms below* t by $\text{at}_{\leq t} := \bigcup_{t' \in \text{post-order}(T,t)} \chi(t')$, and the *atoms strictly below* t by $\text{at}_{<t} := \text{at}_{\leq t} \setminus \chi(t)$.

Algorithms that decide consistency or solve #AS (Fichte et al. 2017a; Jakl, Pichler, and Woltran 2009) proceed by dynamic programming along the tree decomposition (in post-order) where at each node of the tree information is gathered (Bodlaender and Kloks 1996) in a table by a (local) algorithm $\mathbb{A}$. The local algorithm is often also called *table algorithm*. More generally, a *table* is a set of rows, where a *row* $\vec{u}$ is a sequence of fixed length. Similar as for sequences when addressing the i-th element, for a set U of rows (table) we let $U_{(i)} := \{\vec{u}_{(i)} \mid \vec{u} \in U\}$. The actual length, content, and meaning of the rows depend on the algorithm $\mathbb{A}$. Since we later traverse the tree decomposition repeatedly running different algorithms, we explicitly state $\mathbb{A}$-*row* if rows of this *type* are syntactically used for algorithm $\mathbb{A}$ and similar $\mathbb{A}$-*table* for tables. In order to access tables computed at certain nodes after a traversal as well as to provide better readability, we attribute tree decompositions with an additional mapping to store tables. Formally, a *tabled tree decomposition (TTD)* of graph G is a triple $\mathcal{T} = (T, \chi, \tau)$, where (T, χ) is a tree decomposition of G and τ maps nodes t of T to tables. If not specified otherwise, we assume that $\tau(t) = \{\}$ for every node t of T. When a TTD has been computed using algorithm $\mathbb{A}$ after traversing the entire decomposition, we call the decomposition the $\mathbb{A}$-TTD of the given input instance. Then, the dynamic programming approach for ASP performs the following steps for a given program Π:

1. Construct the primal graph of Π.
2. Compute a tree decomposition of the graph.
3. Run algorithm $\texttt{DP}_\mathbb{A}$ (see Listing 1). It takes a tabled tree decomposition $\mathcal{T} = (T, \chi, \iota)$ with $T = (N, \cdot, n)$ and traverses T in post-order (post-order(T, n) provides this sequence of nodes for tree T rooted at n). At each node $t \in N$ it computes a new $\mathbb{A}$-table $o(t)$ by executing the algorithm $\mathbb{A}$. The algorithm $\mathbb{A}$ has a "local view" on the computation and can access only t, the atoms in the bag $\chi(t)$, the bag-program Π_t, and child $\mathbb{A}$-table $o(t')$ for any child t' of t.[4]
4. Output the $\mathbb{A}$-tabled tree decomposition (T, χ, o).

[4]Note that in Listing 1, $\mathbb{A}$ takes in addition as input the set P and table ι_t. We will later reuse this listing. Then, P represents the set of projected atoms and ι_t is a table at t from an earlier traversal.

Listing 1: Algorithm $\texttt{DP}_\mathbb{A}((\Pi, P), \mathcal{T})$
Dynamic programming on TTD $\mathcal{T}$, c.f., (Fichte et al. 2017a).

In: Problem instance (Π, P), TTD $\mathcal{T} = (T, \chi, \iota)$ of G_Π such that n is the root of T and children$(t, T) = \langle t_1, \ldots, t_\ell \rangle$.
Out: $\mathbb{A}$-TTD (T, χ, o) with $\mathbb{A}$-table mapping o

```
o ← empty mapping
for iterate t in post-order(T,n) do
    o(t) ← 𝔸(t, χ(t), ι(t), (Π_t, P), ⟨o(t_1), ..., o(t_ℓ)⟩)
return (T, χ, o)
```

5. Print the result by interpreting table $o(n)$ for root n of T.

Then, the actual computation of algorithm $\mathbb{A}$ is a somewhat technical case distinction of the types $\text{type}(t)$ we can have when considering node t. Algorithms for counting answer sets of disjunctive programs (Jakl, Pichler, and Woltran 2009) and its extensions (Fichte et al. 2017a) have already been established. Implementations of these algorithms can be useful also for solving (Fichte et al. 2017a; 2017c), but the running time is clearly double exponential time in the treewidth in the worst case. We, however, establish an algorithm ($\mathbb{PHC}$) that is restricted to head-cycle-free programs. The runtime of our algorithm is factorial in the treewidth and therefore faster than previous algorithms. Our constructions are inspired by ideas used in previous dynamic programming algorithms (Pichler et al. 2014). In the following, we first present the (local) algorithm for the problem of deciding whether a head-cycle-free program has an answer set (consistency). In the end, this algorithm outputs a new tabled tree decomposition, which we later reuse to solve our actual counting problem. Note that the tree decomposition itself remains the same, but for readability, we keep the computed tables and nodes aligned.

Consistency of Head-Cycle-Free Programs

We can use algorithm $\texttt{DP}_{\mathbb{PHC}}$ to decide the consistency problem for head-cycle-free programs and simply specify our new local algorithm ($\mathbb{PHC}$) that "transforms" tables from one node to another. As graph representation we use the primal graph. The idea is to implicitly apply along the tree decomposition the characterization of answer sets by Lin and Zhao (2003) extended to head-cycle-free programs (Ben-Eliyahu and Dechter 1994). To this end, we store in table $o(t)$ at each node t rows of the form $\langle I, \mathcal{P}, \sigma \rangle$. The first position consists of an interpretation I restricted to the bag $\chi(t)$. For a sequence $\vec{u}$, we write $\mathcal{I}(\vec{u}) := \vec{u}_{(1)}$ to address the *interpretation part*. The second position consists of a set $\mathcal{P} \subseteq I$ that represents atoms in I for which we know that they have already been proven. The third position σ is a sequence of the atoms in I such that there is a super-sequence σ' of σ, which induces an ordering $<_{\sigma'}$. Our local algorithm $\mathbb{PHC}$ stores interpretation parts always restricted to bag $\chi(t)$ and ensures that an interpretation can be extended to a model of sub-program $\Pi_{\leq t}$. More precisely, it guarantees that interpretation I can be extended to a model $I' \supseteq I$ of $\Pi_{\leq t}$ and that the atoms in $I' \setminus I$ (and the atoms in $\mathcal{P} \subseteq I$) have already been *proven*, using some induced ordering $<_{\sigma'}$ where σ is a sub-sequence of σ'. In the end, an interpretation $\mathcal{I}(\vec{u})$ of a row $\vec{u}$ of the table $o(n)$ at the root n proves that there is a

Listing 2: Table algorithm $\mathbb{PHC}(t, \chi_t, \cdot, (\Pi_t, \cdot), \langle \tau_1, \ldots \rangle)$.

In: Node t, bag χ_t, bag-program Π_t, $\langle \tau_1, \ldots \rangle$ is the sequence of $\mathbb{PHC}$-tables of children of t. **Out:** $\mathbb{PHC}$-table τ_t.

1 **if** $\mathrm{type}(t) = \mathit{leaf}$ **then** $\tau_t \leftarrow \{\langle \emptyset, \emptyset, \langle\rangle \rangle\}$
2 **else if** $\mathrm{type}(t) = \mathit{int}$ and $a \in \chi_t$ *is the introduced atom* **then**
3 $\quad \tau_t \leftarrow \{\langle J, \mathcal{P} \cup \mathrm{proven}(J, \sigma', \Pi_t), \sigma' \rangle \mid \langle I, \mathcal{P}, \sigma \rangle \in \tau_1, J \in \{I, I_a^+\}, J \models \Pi_t, \sigma' \in \mathrm{ords}(\sigma, \{a\} \cap J)\}$
4 **else if** $\mathrm{type}(t) = \mathit{rem}$ and $a \notin \chi_t$ *is the removed atom* **then**
5 $\quad \tau_t \leftarrow \{\langle I_a^-, \mathcal{P}_a^-, \sigma_a^\sim \rangle \mid \langle I, \mathcal{P}, \sigma \rangle \in \tau_1, a \in \mathcal{P} \cup (\{a\} \setminus I)\}$
6 **else if** $\mathrm{type}(t) = \mathit{join}$ **then**
7 $\quad \tau_t \leftarrow \{\langle I, \mathcal{P}_1 \cup \mathcal{P}_2, \sigma \rangle \mid \langle I, \mathcal{P}_1, \sigma \rangle \in \tau_1, \langle I, \mathcal{P}_2, \sigma \rangle \in \tau_2\}$
8 **return** τ_t

$\sigma_{\sigma_i}^\sim := \langle \sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_k \rangle$ where $\sigma = \langle \sigma_1, \ldots, \sigma_k \rangle$, $S_e^+ := S \cup \{e\}$, and $S_e^- := S \setminus \{e\}$.

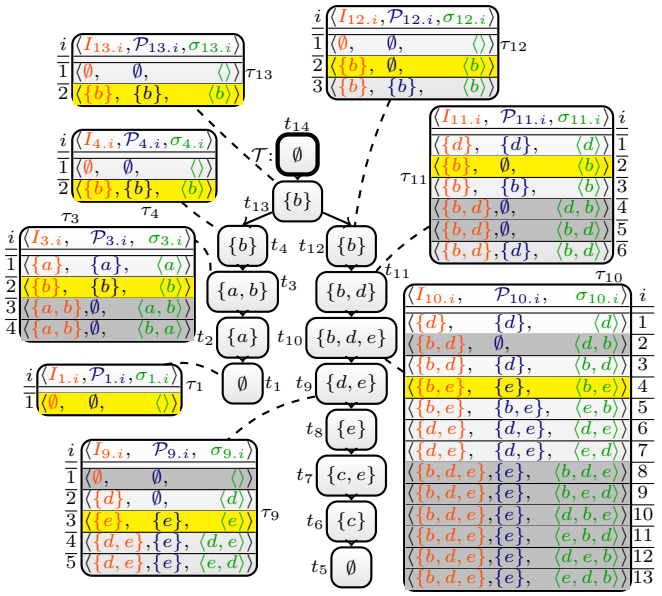


Figure 2: Selected tables of τ obtained by $\mathtt{DP}_{\mathbb{PHC}}$ on TD $\mathcal{T}$.

superset $I' \supseteq \mathcal{I}(\vec{u})$ that is an answer set of $\Pi = \Pi_{\leq n}$.

Listing 2 presents the algorithm $\mathbb{PHC}$. Intuitively, whenever an atom a is introduced (*int*), we decide whether we include a in the interpretation, determine bag atoms that can be proven in consequence of this decision, and update the sequence σ accordingly. To this end, we define for a given interpretation I and a sequence σ the set $\mathrm{proven}(I, \sigma, \Pi_t) := \bigcup_{r \in \Pi_t, a \in H_r} \{a \mid B_r^+ \subseteq I, I \cap B_r^- = \emptyset, I \cap (H_r \setminus \{a\}) = \emptyset, B_r^+ <_\sigma a\}$ where $B_r^+ <_\sigma a$ holds if $b <_\sigma a$ is true for every $b \in B_r^+$. Moreover, given a sequence $\sigma = \langle \sigma_1, \ldots, \sigma_k \rangle$ and a set A of atoms, we compute the potential sequences involving A. Therefore, we let $\mathrm{ords}(\sigma, A) := \{\sigma \mid A = \emptyset\} \cup \bigcup_{a \in A} \{\langle a, \sigma_1, \ldots, \sigma_k \rangle, \ldots, \langle \sigma_1, \ldots, \sigma_k, a \rangle\}$. When removing (*rem*) an atom a, we only keep those rows where a has been proven (contained in $\mathcal{P}$) and then restrict remaining rows to the bag (not containing a). In case the node is of type *join*, we combine two rows in two different child tables, intuitively, we are enforced to agree on the interpretations I and sequences σ. However, concerning the individual proofs $\mathcal{P}$, it suffices that an atom is proven in one of the rows.

Example 5. *Recall program Π from Example 1. Figure 2 depicts a TD $\mathcal{T} = (T, \chi)$ of the primal graph G_1 of Π. Further, the figure illustrates a snippet of tables of the TTD (T, χ, τ), which we obtain when running $\mathtt{DP}_{\mathbb{PHC}}$ on program Π and TD $\mathcal{T}$ according to Listing 2. In the following, we briefly discuss some selected rows of those tables. Note that for simplicity and space reasons, we write τ_j instead of $\tau(t_j)$ and identify rows by their node and identifier i in the figure. For example, the row $\vec{u}_{13.3} = \langle I_{13.3}, \mathcal{P}_{13.3}, \sigma_{13.3} \rangle \in \tau_{13}$ refers to the third row of table τ_{13} for node t_{13}. Node t_1 is of type leaf. Table τ_1 has only one row, which consists of the empty interpretation, empty set of proven atoms, and the empty sequence (Line 1). Node t_2 is of type int and introduces atom a. Executing Line 3 results in $\tau_2 = \{\langle \emptyset, \emptyset, \langle\rangle \rangle, \langle \{a\}, \emptyset, \langle a \rangle \rangle\}$. Node t_3 is of type int and introduces b. Then, bag-program at node t_3 is $\Pi_{t_3} = \{a \vee b \leftarrow\}$. By construction (Line 3) we ensure that interpretation $I_{3.i}$ is a model of Π_{t_3} for every row $\langle I_{3.i}, \mathcal{P}_{3.i}, \sigma_{3.i} \rangle$ in τ_3. Node t_4 is of type rem. Here, we restrict the rows such that they contain only atoms occurring in bag $\chi(t_4) = \{b\}$. To this end, Line 5 takes only rows $\vec{u}_{3.i}$ of table τ_3 where atoms in $I_{3.i}$ are also proven, i.e., contained in $\mathcal{P}_{3.i}$. In particular, every row in table τ_4 originates from at least one row in τ_3 that either proves $a \in \mathcal{P}_{3.i}$ or where $a \notin I_{3.i}$. Basic conditions of a TD ensure that once an atom is removed, it will not occur in any bag at an ancestor node. Hence, we also encountered all rules where atom a occurs. Nodes t_5, t_6, t_7, and t_8 are symmetric to nodes t_1, t_2, t_3, and t_4. Nodes t_9 and t_{10} again introduce atoms. Observe that $\mathcal{P}_{10.4} = \{e\}$ since $\sigma_{10.4}$ does not allow to prove b using atom e. However, $\mathcal{P}_{10.5} = \{b, e\}$ as the sequence $\sigma_{10.5}$ allows to prove b. In particular, in row $\vec{u}_{10.5}$ atom e is used to derive b. As a result, atom b can be proven, whereas ordering $\sigma_{10.4} = \langle b, e \rangle$ does not serve in proving b. We proceed similar for nodes t_{11} and t_{12}. At node t_{13} we join tables τ_4 and τ_{12} according to Line 7. Finally, we have $\tau_{14} \neq \emptyset$. Hence, Π has an answer set. We can construct the answer set $\{b, e\}$ by combining the interpretation parts I of the yellow marked rows of Figure 2.*

Next, we provide a notion to reconstruct answer sets from a computed TTD, which allows for computing for a given row its predecessor rows in the corresponding child tables, c.f., (Fichte et al. 2018). Let Π be a program, $\mathcal{T} = (T, \chi, \tau)$ be an $\mathbb{A}$-TTD of G_Π, and t be a node of T where $\mathrm{children}(t, T) = \langle t_1, \ldots, t_\ell \rangle$. Given a sequence $\vec{s} = \langle s_1, \ldots, s_\ell \rangle$, we let $\langle\!\{\vec{s}\}\!\rangle := \langle \{s_1\}, \ldots, \{s_\ell\} \rangle$. For a given $\mathbb{A}$-row $\vec{u}$, we define the originating $\mathbb{A}$-rows of $\vec{u}$ in node t by $\mathbb{A}\text{-origins}(t, \vec{u}) := \{\vec{s} \mid \vec{s} \in \tau(t_1) \times \cdots \times \tau(t_\ell), \vec{u} \in \mathbb{A}(t, \chi(t), \cdot, (\Pi_t, \cdot), \langle\!\{\vec{s}\}\!\rangle)\}$. We extend this to an $\mathbb{A}$-table ρ by $\mathbb{A}\text{-origins}(t, \rho) := \bigcup_{\vec{u} \in \rho} \mathbb{A}\text{-origins}(t, \vec{u})$.

Example 6. *Consider program Π and $\mathbb{PHC}$-tabled tree decomposition (T, χ, τ) from Example 5. We focus on $u_{1.1}^\rightarrow = \langle \emptyset, \emptyset, \langle\rangle \rangle$ of table τ_1 of leaf t_1. The row $u_{1.1}^\rightarrow$ has no preceding row, since $\mathrm{type}(t_1) = \mathit{leaf}$. Hence, we have $\mathbb{PHC}\text{-origins}(t_1, u_{1.1}^\rightarrow) = \{\langle\rangle\}$. The origins of row $u_{11.1}^\rightarrow$ of table τ_{11} are given by $\mathbb{PHC}\text{-origins}(t_{11}, u_{11.1}^\rightarrow)$, which correspond to the preceding rows in table τ_{10} that lead to row $u_{11.1}^\rightarrow$ of table τ_{11} when running algorithm $\mathbb{PHC}$, i.e., $\mathbb{PHC}\text{-origins}(t_{11}, u_{11.1}^\rightarrow) = \{\langle u_{10.1}^\rightarrow \rangle, \langle u_{10.6}^\rightarrow \rangle, \langle u_{10.7}^\rightarrow \rangle\}$. Ori-*

gins of row $\vec{u}_{12.2}$ *are given by* $\mathbb{PHC}\text{-origins}(t_{12}, u_{12.2}) = \{\langle u_{11.2}\rangle, \langle u_{11.6}\rangle\}$. *Note that* $u_{11.4}$ *and* $u_{11.5}$ *are not among those origins, since* d *is not proven. Observe that* $\mathbb{PHC}\text{-origins}(t_j, \vec{u}) = \emptyset$ *for any row* $\vec{u} \notin \tau_j$. *For node* t_{13} *of type join and row* $u_{13.2}$, *we obtain* $\mathbb{PHC}\text{-origins}(t_{13}, u_{13.2}) = \{\langle u_{4.2}, u_{12.2}\rangle, \langle u_{4.2}, u_{12.3}\rangle\}$.

Next, we provide statements on correctness and a runtime analysis of our algorithm.

Theorem 1 (⋆[5])**.** *The algorithm* $\mathtt{DP}_{\mathbb{PHC}}$ *is correct.*
In other words, given a head-cycle-free program Π *and a TTD* $\mathcal{T} = (T, \chi, \cdot)$ *of* G_Π *where* $T = (N, \cdot, n)$ *with root* n. *Then, algorithm* $\mathtt{DP}_{\mathbb{PHC}}((\Pi, \cdot), \mathcal{T})$ *returns the* $\mathbb{PHC}$*-TTD* (T, χ, τ) *such that* Π *has an answer set if and only if* $\langle \emptyset, \emptyset, \langle\rangle\rangle \in \tau(n)$. *Further, we can construct all the answer sets of* Π *from transitively following the origins of* $\tau(n)$.

Proof (Idea). For soundness, we state an invariant and establish that this invariant holds for every node $t \in N$. For each row $\vec{u} = \langle I, \mathcal{P}, \sigma\rangle \in \tau(t)$, we have $I \subseteq \chi(t), \mathcal{P} \subseteq I$, and σ is a sequence over atoms in I. Intuitively, we ensure that $I \models \Pi_{\leq t}$ and that exactly the atoms in $\mathrm{at}_{<t}$ and $\mathcal{P}$ can be proven using a super-sequence σ' of σ. By construction, we guarantee that we can decide consistency if row $\langle \emptyset, \emptyset, \langle\rangle\rangle \in \tau(n)$. Further, we can even reconstruct answer sets, by following $\mathbb{PHC}$-origins of this single row back to the leaves. For completeness, we show that we obtain all the rows required to output all the answer sets of Π. □

Theorem 2. *Given a head-cycle-free program* Π *and a tree decomposition* $\mathcal{T} = (T, \chi)$ *of* G_Π *of width* k *with* g *nodes. Algorithm* $\mathtt{DP}_{\mathbb{PHC}}$ *runs in time* $\mathcal{O}(3^k \cdot k! \cdot g)$.

Proof (Sketch). Let $d = k + 1$ be maximum bag size of the tree decomposition $\mathcal{T}$. The table $\tau(t)$ has at most $3^d \cdot d!$ rows, since for a row $\langle I, \mathcal{P}, \sigma\rangle$ we have $d!$ many sequences σ, and by construction of algorithm $\mathbb{PHC}$, an atom can be either in I, both in I and $\mathcal{P}$, or neither in I nor in $\mathcal{P}$. In total, with the help of efficient data structures, e.g., for nodes t with $\mathrm{type}(t) =$ *join*, one can establish a runtime bound of $\mathcal{O}(3^d \cdot d!)$. Then, we apply this to every node t of the tree decomposition, which resulting in running time $\mathcal{O}(3^d \cdot d! \cdot g) \subseteq \mathcal{O}(3^k \cdot k! \cdot g)$. □

A natural question is whether we can significantly improve this algorithm for fixed k. To this end, we take the *exponential time hypothesis (ETH)* into account, which states that there is some real $s > 0$ such that we cannot decide satisfiability of a given 3-CNF formula F in time $2^{s \cdot |F|} \cdot \|F\|^{\mathcal{O}(1)}$.

Proposition 2. *Unless ETH fails, consistency of head-cycle-free program* Π *cannot be decided in time* $2^{o(k)} \cdot \|\Pi\|^{o(k)}$ *where* k *is the treewidth of the primal graph of* Π.

Proof. The result follows by a reduction from SAT to ASP (head-cycle-free) similar to the proof of Proposition 1. □

[5]Auxiliary content and proof of statements marked with "⋆" are in an authors self-archived version (Fichte, and Hecher 2018).

[6]ν contains rows obtained by recursively following origins of $\tau(n)$.

[7]Later we use (among others) $\mathtt{PCNT}_{\mathbb{PHC}}$ where $\mathbb{A} = \mathbb{PHC}$.

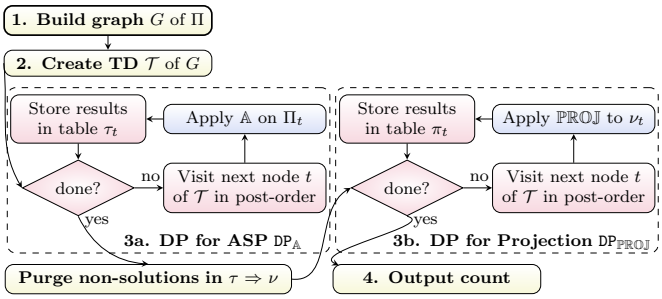


Figure 3: Algorithm $\mathtt{PCNT}_{\mathbb{A}}$ consists of $\mathtt{DP}_{\mathbb{A}}$ and $\mathtt{DP}_{\mathbb{PROJ}}$.

In the construction above, we store an arbitrary but fixed ordering on the involved atoms. We believe that we cannot avoid these orderings in general, since we have to compensate arbitrarily "bad" orderings induced by the decomposition, which leads us to the following conjecture.

Conjecture 1. *Unless ETH fails, consistency of a head-cycle-free program* Π *cannot be decided in time* $2^{o(k \cdot \log(k))} \cdot \|\Pi\|^{o(k)}$ *where* k *is the treewidth of the primal graph of* Π.

In other words, we claim that consistency for head-cycle-free programs is slightly superexponential. We would like to mention that Lokshtanov, Marx, and Saurabh (2011) argue that whenever we cannot avoid an ordering the problem is expected to be slightly superexponential. If the conjecture holds, our algorithm is asymptotically worst-case optimal, even for fixed treewidth k since $\mathtt{DP}_{\mathbb{PHC}}$ runs in time $\mathcal{O}(2^{k \cdot \log(k)} \cdot g)$, where number g of decomposition nodes is linear in the size of the instance (Bodlaender 1996).

Dynamic Programming for #PAS

In this section, we present our dynamic programming algorithm[7] $\mathtt{PCNT}_{\mathbb{A}}$, which allows for solving the projected answer set counting problem (#PAS). $\mathtt{PCNT}_{\mathbb{A}}$ is based on an approach of projected counting for Boolean formulas (Fichte et al. 2018) where TDs are traversed multiple times. We show that ideas from that approach can be fruitfully extended to answer set programming. Figure 3 illustrates the steps of $\mathtt{PCNT}_{\mathbb{A}}$. First, we construct the primal graph G_Π of the input program Π and compute a TD of Π. Then, we traverse the TD a first time by running $\mathtt{DP}_{\mathbb{A}}$ (Step 3a), which outputs a TTD $\mathcal{T}_{\text{cons}} = (T, \chi, \tau)$. Afterwards, we traverse $\mathcal{T}_{\text{cons}}$ in pre-order and remove all rows from the tables that cannot be extended to an answer set (*"Purge non-solutions"*). In other words, we keep only rows $\vec{u}$ of table $\tau(t)$ at node t, if $\vec{u}$ is involved in those rows that are used to construct an answer set of Π, and let the resulting TTD be $\mathcal{T}_{\text{purged}} = (T, \chi, \nu)$[6]. We refer to ν as *purged table mapping*. In Step 3b ($\mathtt{DP}_{\mathbb{PROJ}}$), we traverse $\mathcal{T}_{\text{purged}}$ to count interpretations with respect to the projection atoms and obtain $\mathcal{T}_{\text{proj}} = (T, \chi, \pi)$. From the table $\pi(n)$ at the root n of T, we can then read the projected answer sets count of the input instance. In the following, we only describe the local algorithm ($\mathbb{PROJ}$), since the traversal in $\mathtt{DP}_{\mathbb{PROJ}}$ is the same as before. For $\mathbb{PROJ}$, a row at a node t is a pair $\langle \rho, c\rangle \in \pi(t)$, where $\rho \subseteq \nu(t)$ is an $\mathbb{A}$-table and c is a non-negative integer. In fact, integer c stores the number of intersecting solutions (ipasc). However, we aim for the pro-

jected answer sets count (pasc), whose computation requires a few additional definitions. Therefore, we can simply widen definitions from very recent work (Fichte et al. 2018).

In the remainder, we assume (Π, P) to be an instance of #PAS, (T, χ, τ) to be an $\mathbb{A}$-TTD of G_Π and the mappings τ, ν, and π as used above. Further, let t be a node of T with $\text{children}(t, T) = \langle t_1, \ldots, t_\ell \rangle$ and let $\rho \subseteq \nu(t)$. The relation $=_\text{P} \subseteq \rho \times \rho$ considers equivalent rows with respect to the projection of its interpretations by $=_\text{P} := \{(\vec{u}, \vec{v}) \mid \vec{u}, \vec{v} \in \rho, \mathcal{I}(\vec{u}) \cap P = \mathcal{I}(\vec{v}) \cap P\}$. Let $\text{buckets}_P(\rho)$ be the set of equivalence classes induced by $=_\text{P}$ on ρ, i.e., $\text{buckets}_P(\rho) := (\rho / =_\text{P}) = \{[\vec{u}]_P \mid \vec{u} \in \rho\}$, where $[\vec{u}]_P = \{\vec{v} \mid \vec{v} =_\text{P} \vec{u}, \vec{v} \in \rho\}$ (Wilder 1965). Further, $\text{sub-buckets}_P(\rho) := \{S \mid \emptyset \subsetneq S \subseteq B, B \in \text{buckets}_P(\rho)\}$.

Example 7. *Consider program Π, set P of projection atoms, TTD (T, χ, τ), and table τ_{10} from Example 2 and Figure 2. Note that during purging rows $u_{10.2}^{\rightarrow}$ and $u_{10.8}^{\rightarrow}, \ldots, u_{10.13}^{\rightarrow}$ are removed (highlighted gray), since they are not involved in any answer set, resulting in table ν_{10}. Then, $u_{10.4}^{\rightarrow} =_P u_{10.5}^{\rightarrow}$ and $u_{10.6}^{\rightarrow} =_P u_{10.7}^{\rightarrow}$. The set $\nu_{10} / =_\text{P}$ of equivalence classes of ν_{10} is $\text{buckets}_P(\nu_{10}) = \{\{u_{10.1}^{\rightarrow}\}, \{u_{10.3}^{\rightarrow}\}, \{u_{10.4}^{\rightarrow}, u_{10.5}^{\rightarrow}\}, \{u_{10.6}^{\rightarrow}, u_{10.7}^{\rightarrow}\}\}$.*

Later, we require to construct already computed projected counts for tables of children of a given node t. Therefore, we define the *stored* ipasc of a table $\rho \subseteq \nu(t)$ in table $\pi(t)$ by $\text{s-ipasc}(\pi(t), \rho) := \sum_{\langle \rho, c \rangle \in \pi(t)} c$. We extend this to a sequence $s = \langle \pi(t_1), \ldots, \pi(t_\ell) \rangle$ of tables of length ℓ and a set $O = \{\langle \rho_1, \ldots, \rho_\ell \rangle, \langle \rho'_1, \ldots, \rho'_\ell \rangle, \ldots\}$ of sequences of ℓ tables by $\text{s-ipasc}(s, O) = \prod_{i \in \{1, \ldots, \ell\}} \text{s-ipasc}(s_{(i)}, O_{(i)})$. In other words, we select the i-th position of the sequence together with sets of the i-th positions from the set of sequences.

Intuitively, when we are at a node t in algorithm $\texttt{DP}_{\mathbb{PROJ}}$ we have already computed $\pi(t')$ of $\mathcal{T}_\text{proj}$ for every node t' below t. Then, we compute the projected answer sets count of $\rho \subseteq \nu(t)$. Therefore, we apply the inclusion-exclusion principle to the stored projected answer sets count of origins. We define $\text{pasc}(t, \rho, \langle \pi(t_1), \ldots \rangle) := \sum_{\emptyset \subsetneq O \subseteq \mathbb{A}\text{-origins}(t, \rho)} (-1)^{(|O|-1)} \cdot \text{s-ipasc}(\langle \pi(t_1), \ldots \rangle, O)$. Vaguely speaking, pasc determines the $\mathbb{A}$-origins of table ρ, goes over all subsets of these origins and looks up the stored counts (s-ipasc) in the $\mathbb{PROJ}$-tables of the children t_i of t.

Example 8. *Consider again program Π and TD $\mathcal{T}$ from Example 1 and Figure 2. First, we compute the projected count $\text{pasc}(t_4, \{u_{4.1}^{\rightarrow}\}, \langle \pi(t_3) \rangle)$ for row $u_{4.1}^{\rightarrow}$ of table $\nu(t_4)$, where $\pi(t_3) := \{\langle \{u_{3.1}^{\rightarrow}\}, 1 \rangle, \langle \{u_{3.2}^{\rightarrow}\}, 1 \rangle, \langle \{u_{3.1}^{\rightarrow}, u_{3.2}^{\rightarrow}\}, 1 \rangle\}$ with $u_{3.1}^{\rightarrow} = \langle \emptyset, \emptyset, \langle\rangle \rangle$ and $u_{3.2}^{\rightarrow} = \langle \{a\}, \emptyset, \langle a \rangle \rangle$. Note that t_5 has only the child t_4 and therefore the product in s-ipasc consists of only one factor. Since $\mathbb{PHC}\text{-origins}(t_4, u_{4.1}^{\rightarrow}) = \{\langle u_{3.1}^{\rightarrow} \rangle\}$, only the value of s-ipasc for set $\{\langle u_{3.1}^{\rightarrow} \rangle\}$ is non-zero. Hence, we obtain $\text{pasc}(t_4, \{u_{4.1}^{\rightarrow}\}, \langle \pi(t_3) \rangle) = 1$. Next, we compute $\text{pasc}(t_4, \{u_{4.1}^{\rightarrow}, u_{4.2}^{\rightarrow}\}, \langle \pi(t_3) \rangle)$. Observe that $\mathbb{PHC}\text{-origins}(t_4, \{u_{4.1}^{\rightarrow}, u_{4.2}^{\rightarrow}\}) = \{\langle u_{3.1}^{\rightarrow} \rangle, \langle u_{3.2}^{\rightarrow} \rangle\}$. We sum up the values of s-ipasc for sets $\{u_{4.1}^{\rightarrow}\}$ and $\{u_{4.2}^{\rightarrow}\}$ and subtract the one for set $\{u_{4.1}^{\rightarrow}, u_{4.2}^{\rightarrow}\}$. Hence, we obtain $\text{pasc}(t_4, \{u_{4.1}^{\rightarrow}, u_{4.2}^{\rightarrow}\}, \langle \pi(t_3) \rangle) = 1 + 1 - 1 = 1$.*

Next, we provide a definition to compute ipasc, which can be computed at a node t for given table $\rho \subseteq \nu(t)$ by computing the pasc for children t_i of t using stored ipasc values from tables $\pi(t_i)$, subtracting and adding ipasc values for subsets $\emptyset \subsetneq \varphi \subsetneq \rho$ accordingly. Formally, $\text{ipasc}(t, \rho, s) := 1$ if $\text{type}(t) = \textit{leaf}$ and otherwise $\text{ipasc}(t, \rho, s) := \big| \text{pasc}(t, \rho, s) + \sum_{\emptyset \subsetneq \varphi \subsetneq \rho} (-1)^{|\varphi|} \cdot \text{ipasc}(t, \varphi, s) \big|$ where $s = \langle \pi(t_1), \ldots \rangle$. In other words, if a node is of type *leaf* the ipasc is one, since bags of leaf nodes are empty. Otherwise, we compute the "non-overlapping" count of given table $\rho \subseteq \nu(t)$ with respect to P, by exploiting the inclusion-exclusion principle on $\mathbb{A}$-origins of ρ such that we count every projected answer set only once. Then we have to subtract and add ipasc values ("all-overlapping" counts) for strict subsets φ of ρ, accordingly.

Listing 3: Table algorithm $\mathbb{PROJ}(t, \cdot, \nu_t, (\cdot, P), \langle \pi_1, \ldots \rangle)$ for projected counting.

In: Node t, purged table mapping ν_t, set P of projection atoms, $\langle \pi_1, \ldots \rangle$ is the sequence of $\mathbb{PROJ}$-tables of children of t.
Out: $\mathbb{PROJ}$-table π_t consisting of pairs $\langle \rho, c \rangle$, where $\rho \subseteq \nu_t$ and $c \in \mathbb{N}$.

```
π_t ← { ⟨ρ, ipasc(t, ρ, ⟨π_1, ...⟩)⟩ | ρ ∈ sub-buckets_P(ν_t) }
return π_t
```

Finally, Listing 3 presents the local algorithm $\mathbb{PROJ}$, which stores $\pi(t)$ consisting of every sub-bucket of the given table $\nu(t)$ together with its ipasc.

Example 9. *Recall instance (Π, P), TD $\mathcal{T}$, and tables τ_1, …, τ_{14} from Examples 2, 5, and Figure 2. Figure 4 depicts selected tables of $\pi_1, \ldots, \pi_{14}$ obtained after running $\texttt{DP}_{\mathbb{PROJ}}$ for counting projected answer sets. We assume that row i in table π_t corresponds to $v_{t.i}^{\rightarrow} = \langle \rho_{t.i}, c_{t.i} \rangle$ where $\rho_{t.i} \subseteq \nu(t)$. Recall that for some nodes t, there are rows among different $\mathbb{PHC}$-tables that are removed (highlighted gray in Figure 2) during purging. By purging we avoid to correct stored counters (backtracking) whenever a row has no "succeeding" row in the parent table.*

Next, we discuss selected rows obtained by $\texttt{DP}_{\mathbb{PROJ}}((\Pi, P), (T, \chi, \nu))$. Tables π_1, …, π_{14} are shown in Figure 4. Since $\text{type}(t_1) = \textit{leaf}$, we have $\pi_1 = \langle \{\langle \emptyset, \emptyset, \langle\rangle \rangle\}, 1 \rangle$. Intuitively, at t_1 the row $\langle \emptyset, \emptyset, \langle\rangle \rangle$ belongs to 1 bucket. Node t_2 introduces atom a, which results in table $\pi_2 := \{\langle \{u_{2.1}^{\rightarrow}\}, 1 \rangle, \langle \{u_{2.2}^{\rightarrow}\}, 1 \rangle, \langle \{u_{2.1}^{\rightarrow}, u_{2.2}^{\rightarrow}\}, 1 \rangle\}$, where $u_{2.1}^{\rightarrow} = \langle \emptyset, \emptyset, \langle\rangle \rangle$ and $u_{2.2}^{\rightarrow} = \langle \{a\}, \emptyset, \langle a \rangle \rangle$ (derived similarly to table π_4 as in Example 8). Node t_{10} introduces projected atom e, and node t_{11} removes e. For row $v_{11.1}^{\rightarrow}$ we compute the count $\text{ipasc}(t_{11}, \{u_{11.1}^{\rightarrow}\}, \langle \pi_{10} \rangle)$ by means of pasc. Therefore, take for φ the singleton set $\{u_{11.1}^{\rightarrow}\}$. We simply have $\text{ipasc}(t_{11}, \{u_{11.1}^{\rightarrow}\}, \langle \pi_{10} \rangle) = \text{pasc}(t_{11}, \{u_{11.1}^{\rightarrow}\}, \langle \pi_{10} \rangle)$. To compute $\text{pasc}(t_{11}, \{u_{11.1}^{\rightarrow}\}, \langle \pi_{10} \rangle)$, we take for O the sets $\{u_{10.1}^{\rightarrow}\}$, $\{u_{10.6}^{\rightarrow}\}$, $\{u_{10.7}^{\rightarrow}\}$, and $\{u_{10.6}^{\rightarrow}, u_{10.7}^{\rightarrow}\}$ into account, since all other non-empty subsets of origins of $u_{11.1}^{\rightarrow}$ in ν_{10} do not occur in π_{10}. Then, we take the sum over the values $\text{s-ipasc}(\langle \pi_{10} \rangle, \{u_{10.1}^{\rightarrow}\}) = 1$, $\text{s-ipasc}(\langle \pi_{10} \rangle, \{u_{10.6}^{\rightarrow}\}) = 1$, $\text{s-ipasc}(\langle \pi_{10} \rangle, \{u_{10.7}^{\rightarrow}\}) = 1$ and subtract $\text{s-ipasc}(\langle \pi_{10} \rangle, \{u_{10.6}^{\rightarrow}, u_{10.7}^{\rightarrow}\}) = 1$. This results in $\text{pasc}(t_{11}, \{u_{11.1}^{\rightarrow}\}, \langle \pi_{10} \rangle) = c_{10.1} + c_{10.7} + c_{10.8} - c_{10.9} = 2$. We proceed similarly for row $v_{11.2}$, resulting in $c_{11.2} = 1$.

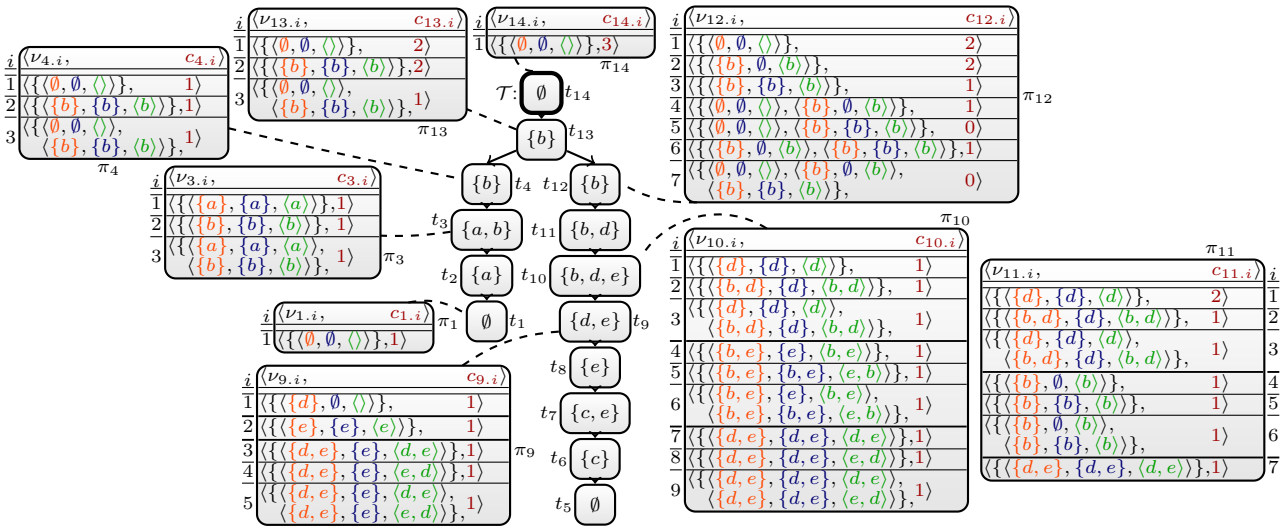


Figure 4: Selected tables of π obtained by $\mathtt{DP}_{\mathbb{PROJ}}$ on TD $\mathcal{T}$ and purged table mapping ν (obtained by purging on τ, c.f, Figure 2).

Then for row $v_{11.3}$, $\mathrm{ipasc}(t_{11}, \{u_{11.1}^{\rightarrow}, u_{11.6}^{\rightarrow}\}, \langle \pi_{10} \rangle) = |\mathrm{pasc}(t_{11}, \{u_{11.1}^{\rightarrow}, u_{11.6}^{\rightarrow}\}, \langle \pi_{10} \rangle) - \mathrm{ipasc}(t_{11}, \{u_{11.1}^{\rightarrow}\}, \langle \pi_{10} \rangle) - \mathrm{ipasc}(t_{11}, \{u_{11.6}^{\rightarrow}\}, \langle \pi_{10} \rangle)| = |2 - c_{11.1} - c_{11.2}| = |2 - 2 - 1| = |-1| = 1 = c_{11.3}$. *Hence,* $c_{11.3} = 1$ *represents the number of projected answer sets, both rows* $u_{11.1}^{\rightarrow}$ *and* $u_{11.6}^{\rightarrow}$ *have in common. We then use it for table* t_{12}. *Node* t_{12} *removes projection atom* d. *For node* t_{13} *where* $\mathrm{type}(t_{13}) =$ *join one multiplies stored* s-ipasc *values for* $\mathbb{A}$*-rows in the two children of* t_{13} *accordingly. In the end, the projected answer sets count of* Π *corresponds to* $\text{s-ipasc}(\langle \pi_{14} \rangle, u_{14.1}^{\rightarrow}) = 3$.

Runtime Analysis and Correctness

Next, we present asymptotic upper bounds on the runtime of our Algorithm $\mathtt{DP}_{\mathbb{PROJ}}$. We assume $\gamma(n)$ to be the number of operations that are required to multiply two n-bit integers, which can be achieved in time $n \cdot \log n \cdot \log \log n$ (Knuth 1998; Harvey, van der Hoeven, and Lecerf 2016). Often, even constant-time multiplication is assumed.

Theorem 3. *Given a* #PAS *instance* (Π, P) *and a tabled tree decomposition* $\mathcal{T}_{purged} = (T, \chi, \nu)$ *of* G_Π *of width* k *with* g *nodes. Then,* $\mathtt{DP}_{\mathbb{PROJ}}$ *runs in time* $\mathcal{O}(2^{4m} \cdot g \cdot \gamma(\|\Pi\|))$ *where* $m := \max(\{\nu(t) \mid t \in N\})$.

Proof. Let $d = k+1$ be maximum bag size of the TD $\mathcal{T}$. For each node t of T, we consider the table $\nu(t)$ of $\mathcal{T}_{\text{purged}}$. Let TDD (T, χ, π) be the output of $\mathtt{DP}_{\mathbb{PROJ}}$. In worst case, we store in $\pi(t)$ each subset $\rho \subseteq \nu(t)$ together with exactly one counter. Hence, we have at most 2^m many rows in ρ. In order to compute ipasc for ρ, we consider every subset $\varphi \subseteq \rho$ and compute pasc. Since $|\rho| \leq m$, we have at most 2^m many subsets φ of ρ. Finally, for computing pasc, we consider in the worst case each subset of the origins of φ for each child table, which are at most $2^m \cdot 2^m$ because of nodes t with $\mathrm{type}(t) =$ *join*. In total, we obtain a runtime bound of $\mathcal{O}(2^m \cdot 2^m \cdot 2^m \cdot 2^m \cdot \gamma(\|\Pi\|)) \subseteq \mathcal{O}(2^{4m} \cdot \gamma(\|\Pi\|))$ due to multiplication of two n-bit integers for nodes t with $\mathrm{type}(t) =$ *join* at costs $\gamma(n)$. Then, we apply this to every node of T resulting in runtime $\mathcal{O}(2^{4m} \cdot g \cdot \gamma(\|\Pi\|))$. □

Corollary 1. *Given an instance* (Π, P) *of* #PAS *where* Π *is head-cycle-free and has treewidth* k. *Then,* $\mathtt{PCNT}_{\mathbb{PHC}}$ *runs in time* $\mathcal{O}(2^{3^{k+1.27} \cdot k!} \cdot \|\Pi\| \cdot \gamma(\|\Pi\|))$.

Proof. We can compute in time $2^{\mathcal{O}(k^3)} \cdot \|G_\Pi\|$ a TD $\mathcal{T}'$ with $g \leq \|\Pi\|$ nodes of width at most k (Bodlaender 1996). Then, we can simply run $\mathtt{DP}_{\mathbb{PHC}}$, which runs in time $\mathcal{O}(3^k \cdot k! \cdot \|\Pi\|)$ by Theorem 2 and since the number of nodes of a tree decomposition is linear in the size of the input instance (Bodlaender 1996). Then, we again traverse the TD for purging and output $\mathcal{T}_{\text{purged}}$, which runs in time single exponential in the treewidth and linear in the instance size. Finally, we run $\mathtt{DP}_{\mathbb{PROJ}}$ and obtain by Theorem 3 that the runtime bound $\mathcal{O}(2^{4 \cdot 3^k \cdot k!} \cdot \|\Pi\| \cdot \gamma(\|\Pi\|)) \subseteq \mathcal{O}(2^{3^{k+1.27} \cdot k!} \cdot \|\Pi\| \cdot \gamma(\|\Pi\|))$. Hence, the corollary holds. □

The next result establishes lower bounds.

Theorem 4. *Unless ETH fails,* #PAS *cannot be solved in time* $2^{2^{o(k)}} \cdot \|\Pi\|^{o(k)}$ *for a given instance* (Π, P), *where* k *is the treewidth of the primal graph of* Π.

Proof. Assume for proof by contradiction that there is such an algorithm. We show that this contradicts a very recent result (Lampis and Mitsou 2017; Fichte et al. 2018), which states that one cannot decide the validity of a QBF $\forall V_1 . \exists V_2 . E$ in time $2^{2^{o(k)}} \cdot \|E\|^{o(k)}$, where E is in CNF. Let $(\forall V_1 . \exists V_2 . E, k)$ be an instance of $\forall\exists$-SAT parameterized by the treewidth k. Then, we reduce to an instance $((\Pi, P), 2k)$ of the decision version #PAS-exactly-$2^{|V_1|}$ when parameterized by treewidth of G_Π such that $P = V_1$, the number of solutions is exactly $2^{|V_1|}$, and Π is as follows. For each $v \in V_1 \cup V_2$, program Π contains rule $v \vee nv \leftarrow$. Each clause $x_1 \vee \ldots \vee x_i \vee \neg x_{i+1} \vee \ldots \vee \neg x_j$ results

in one additional rule $\leftarrow\ \neg x_1, \ldots, \neg x_i, x_{i+1}, \ldots, x_j$. It is easy to see that the reduction is correct and therefore instance $((\Pi, P), 2k)$ is a yes instance of #PAS-exactly-$2^{|V_1|}$ if and only if $(\forall V_1.\exists V_2.E, k)$ is a yes instance of problem $\forall\exists$-SAT. In fact, the reduction is also an fpl-reduction, since the treewidth of Π at most doubles due to duplication of atoms. Note that we require an *fpl*-reduction here, as results do not carry over from simple fpt-reductions. This concludes the proof and establishes the theorem. □

Finally, we state that indeed $\texttt{PCNT}_{\mathbb{PHC}}$ gives the projected answer sets count of a given head-cycle-free program Π.

Proposition 3 (⋆). *Algorithm* $\texttt{PCNT}_{\mathbb{PHC}}$ *is correct and outputs for any instance of* #PAS *its projected answer sets count.*

Proof. Soundness follows by establishing an invariant for any row of $\pi(t)$ guaranteeing that the values of ipasc indeed capture "all-overlapping" counts of $\Pi_{\leq t}$. One can show that the invariant is a consequence of the properties of $\mathbb{PHC}$ and the additional "purging" step, which neither destroys soundness nor completeness of $\texttt{DP}_{\mathbb{PHC}}$. Further, completeness guarantees that indeed all the required rows are computed. □

Solving #PDAS for Disjunctive Programs

In this section, we extend our algorithm to solve the projected answer set counting problem (#PDAS) for disjunctive programs. Therefore, we simply use a local algorithm $\mathbb{PRIM}$ for disjunctive ASP that was introduced in the literature (Fichte et al. 2017a; Jakl, Pichler, and Woltran 2009). Recall algorithm $\texttt{PCNT}_{\mathbb{A}}$ illustrated in Figure 3. First, we construct a graph representation and heuristically compute a tree decomposition of this graph. Then, we run $\texttt{DP}_{\mathbb{PRIM}}$ as first traversal resulting in TTD (T, χ, τ). Next, we purge rows of τ, which can not be extended to an answer set resulting in TTD (T, χ, ν). Finally, we compute the projected answer sets count by $\texttt{DP}_{\mathbb{PROJ}}$ and obtain TTD (T, χ, π).

Proposition 4 (⋆). $\texttt{PCNT}_{\mathbb{PRIM}}$ *is correct, i.e., it outputs the projected answer sets count for any instance of* #PDAS.

The following corollary states the runtime results.

Corollary 2. *Given an instance* (Π, P) *of* #PDAS *where* Π *is a disjunctive program of treewidth* k*. Then,* $\texttt{PCNT}_{\mathbb{PRIM}}$ *runs in time* $\mathcal{O}(2^{2^{2^{k+3}}} \cdot \|\Pi\| \cdot \gamma(\|\Pi\|))$.

Proof. The first two steps follow the proof of Corollary 1. However, $\texttt{DP}_{\mathbb{PRIM}}$ runs in time $\mathcal{O}(2^{2^{2^{k+2}}} \cdot \|\Pi\|)$ (Fichte et al. 2017a). Finally, we run $\texttt{DP}_{\mathbb{PROJ}}$ and obtain by Theorem 3 that $\mathcal{O}(2^{4 \cdot 2^{2^{k+2}}} \cdot \|\Pi\| \cdot \gamma(\|\Pi\|)) \subseteq \mathcal{O}(2^{2^{2^{k+3}}} \cdot \|\Pi\| \cdot \gamma(\|\Pi\|))$. □

Again, we are interested in whether we can improve the algorithm significantly. While we obtain lower bounds from the ETH for SAT (single-exponential) and for $\forall\exists$-SAT/$\exists\forall$-SAT (double-exponential), to our knowledge it is unproven whether this extends to $\forall\exists\forall$-SAT and $\exists\forall\exists$-SAT (triple-exponential). Since it was anticipated by Marx and Mitsou (2016) that it follows just by assuming ETH, we state this as hypothesis. In particular, they claimed that alternating quantifier alternations are the reason for large dependence on treewidth. However, the proofs can be quite involved, trading an additional alternation for exponential compression.

Hypothesis 1. *The* $\forall\exists\forall$-SAT *problem for a QBF* Q *in DNF of treewidth* k *can not be decided in time* $2^{2^{2^{o(k)}}} \cdot \|Q\|^{o(k)}$.

Theorem 5. *Unless Hypothesis 1 fails,* #PDAS *for disjunctive programs* Π *cannot be solved in time* $2^{2^{2^{o(k)}}} \cdot \|\Pi\|^{o(k)}$ *for given instance* (Π, P) *of treewidth* k.

Proof. Assume for proof by contradiction that there is such an algorithm. We show that this contradicts Hypothesis 1, i.e., we cannot decide the validity of a QBF $Q = \forall V_1.\exists V_2.\forall V_3.E$ in time $2^{2^{2^{o(k)}}} \cdot \|E\|^{o(k)}$ where E is in DNF. Assume we have given such an instance when parameterized by the treewidth k. In the following, we employ a well-known reduction R (Eiter and Gottlob 1995), which transforms $\exists V_2.\forall V_3.E$ into $\Pi = R(\exists V_2.\forall V_3.E)$ and gives a yes instance Π of consistency if and only if $\exists V_2.\forall V_3.E$ is a yes instance of $\exists\forall$-SAT. Then, we reduce instance (Q, k) via a reduction S to an instance $((\Pi', V_1), 2k + 2)$, where $\Pi' = R(\exists V_2'.\forall V_3.E)$, $V_2' := V_1 \cup V_2$, of the decision version #PDAS-exactly-$2^{|V_1|}$ of #PDAS when parameterized by treewidth such that the number of projected answer sets is exactly $2^{|V_1|}$. It is easy to see that reduction S gives a yes instance (Π', V_1) of #PDAS-exactly-$2^{|V_1|}$ if and only if $\forall V_1.\exists V_2.\forall V_3.E$ is a yes instance of $\forall\exists\forall$-SAT. However, it remains to show that the reduction S indeed increases the treewidth only linearly. Therefore, let $\mathcal{T} = (T, \chi)$ be TD of E. We transform $\mathcal{T}$ into a TD $\mathcal{T}' = (T, \chi')$ of $G_{\Pi'}$ as follows. For each bag $\chi(t)$ of $\mathcal{T}$, we add vertices for the atoms w and w' (two additional atoms introduced in reduction R) and in addition we duplicate each vertex v in $\chi(t)$ (due to corresponding duplicate atoms introduced in reduction R). Observe that $\mathrm{width}(\mathcal{T}') \leq 2 \cdot \mathrm{width}(\mathcal{T}) + 2$. By construction of R, $\mathcal{T}'$ is then a TD of $G_{\Pi'}$. Hence, S is also an fpl-reduction. □

Then, the runtime of algorithm $\texttt{PCNT}_{\mathbb{PRIM}}$ is asymptotically worst-case optimal, depending on multiplication costs $\gamma(n)$.

Conclusions

We introduced novel algorithms to count the projected answer sets (#PAS) of head-cycle-free or arbitrary disjunctive programs. Our algorithms employ dynamic programming and exploit small treewidth of the primal graph of the input program. The second algorithm, which solves arbitrary disjunctive programs, is expected asymptotically optimal assuming the exponential time hypothesis (ETH). More precisely, runtime is triple exponential in the treewidth and polynomial in the size of the input instance. When we restrict the input to head-cycle-free programs, the runtime drops to double exponential.

Our results extend previous work to answer set programming and we believe that it can be applicable to other hard combinatorial problems, such as circumscription (Durand, Hermann, and Kolaitis 2005), quantified Boolean formulas (QBF) (Charwat and Woltran 2016), or default logic (Fichte, Hecher, and Schindler 2018).

References

Aziz, R. A. 2015. *Answer Set Programming: Founded Bounds and Model Counting*. Ph.D. Thesis, Department of Computing and Information Systems, The University of Melbourne.

Balduccini, M.; Gelfond, M.; and Nogueira, M. 2006. Answer set based design of knowledge systems. *AMAI* 47(1-2):183–219.

Ben-Eliyahu, R., and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *AMAI* 12(1):53–87.

Bidoít, N., and Froidevaux, C. 1991. Negation by default and unstratifiable logic programs. *TCS* 78(1):85–112.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, volume 185 of *FAIA*.

Bodlaender, H. L., and Kloks, T. 1996. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms* 21(2):358–402.

Bodlaender, H. L. 1996. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* 25(6):1305–1317.

Bondy, J. A., and Murty, U. S. R. 2008. *Graph theory*, volume 244 of *Graduate Texts in Mathematics*. Springer.

Brewka, G.; Eiter, T.; and Truszczyński, M. 2011. Answer set programming at a glance. *CACM* 54(12):92–103.

Charwat, G., and Woltran, S. 2016. Dynamic programming-based QBF solving. QBF'16.

Cygan, M.; Fomin, F. V.; Kowalik, Ł.; Lokshtanov, D.; Dániel Marx, M. P.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Diestel, R. 2012. *Graph Theory, 4th Edition*, volume 173 of *Graduate Texts in Mathematics*. Springer.

Durand, A.; Hermann, M.; and Kolaitis, P. G. 2005. Subtractive reductions and complete problems for counting complexity classes. *TCS* 340(3):496–513.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *AMAI.* 15(3–4):289–323.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2017a. Answer set solving with bounded treewidth revisited. LPNMR'17. Springer, Extended Version: *CoRR* abs/cs/arXiv:1702.02890.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2017c. DynASP2.5: Dynamic programming on tree decompositions in action. IPEC'17.

Fichte, J. K.; Hecher, M.; Morak, M.; and Woltran, S. 2018. Exploiting treewidth for projected model counting and its limits.. SAT'18. 10929:165–184.

Fichte, J. K.; and Hecher, M. 2018. Exploiting Treewidth in Projected Answer Set Counting.

Fichte, J. K.; Hecher, M.; and Schindler, I. 2018. Default Logic and Bounded Treewidth. LATA'18. Springer.

Gebser, M.; Kaminski, R.; Kaufmann, B.; and Schaub, T. 2012. *Answer Set Solving in Practice*. Morgan & Claypool.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2009. Solution enumeration for projected boolean search problems. CPAIOR'09. Springer.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Comput.* 9(3/4):365–386.

Graham, R. L.; Grötschel, M.; and Lovász, L. 1995. *Handbook of combinatorics*, volume I. Elsevier Science Pub.

Guziolowski, C.; Videla, S.; Eduati, F.; Thiele, S.; Cokelaer, T.; Siegel, A.; and Saez-Rodriguez, J. 2013. Exhaustively characterizing feasible logic models of a signaling network using answer set programming. *Bioinformatics* 29(18):2320–2326. Erratum see Bioinformatics 30, 13, 1942.

Harvey, D.; van der Hoeven, J.; and Lecerf, G. 2016. Even faster integer multiplication. *J. Complexity* 36:1–30.

Jakl, M.; Pichler, R.; and Woltran, S. 2009. Answer-set programming with bounded treewidth. In *IJCAI'09*, 816–822.

Janhunen, T., and Niemelä, I. 2016. The answer set programming paradigm. *AI Magazine* 37(3):13–24.

Kleine Büning, H., and Lettman, T. 1999. *Propositional logic: deduction and algorithms*. New York, CUP.

Kloks, T. 1994. *Treewidth. Computations and Approximations*, volume 842 of *LNCS*. Springer.

Knuth, D. E. 1998. How fast can we multiply? In *The Art of Computer Programming*, volume 2 of *Seminumerical Algorithms*. Addison-Wesley. chapter 4.3.3, 294–318.

Lampis, M., and Mitsou, V. 2017. Treewidth with a quantifier alternation revisited. IPEC'17.

Lin, F., and Zhao, J. 2003. On tight logic programs and yet another translation from normal logic programs to propositional logic. IJCAI'03.

Lokshtanov, D.; Marx, D.; and Saurabh, S. 2011. Slightly superexponential parameterized problems. In *SODA*, 760–776. SIAM.

Baral, C., and Gelfond, M. 1994. Logic programming and knowledge representation. *The Journal of Logic Programming* 19-20:73–148.

Marek, W., and Truszczyński, M. 1991. Autoepistemic logic. *J. of the ACM* 38(3):588–619.

Marx, D., and Mitsou, V. 2016. Double-Exponential and Triple-Exponential Bounds for Choosability Problems Parameterized by Treewidth. ICALP'16, LIPIcs, 28:1–28:15.

Niemelä, I.; Simons, P.; and Soininen, T. 1999. Stable model semantics of weight constraint rules. LPNMR'99.

Nogueira, M.; Balduccini, M.; Gelfond, M.; Watson, R.; and Barry, M. 2001. An A-Prolog decision support system for the Space Shuttle. PADL'01.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Pichler, R.; Rümmele, S.; Szeider, S.; and Woltran, S. 2014. Tractable answer-set programming with weight constraints: bounded treewidth is not enough. *TPLP.* 14(2).

Wilder, R. L. 1965. *Introduction to the Foundations of Mathematics*. John Wiley & Sons, 2nd edition edition.

Lower Bound Founded Logic of Here-and-There: A Preliminary Report

Pedro Cabalar
University of Corunna, Spain
cabalar@udc.es

Jorge Fandinno
University of Toulouse, France
jorge.fandinno@irit.fr

Torsten Schaub and **Sebastian Schellhorn**
University of Potsdam, Germany
{torsten, seschell}@cs.uni-potsdam.de

Abstract

A distinguishing feature of Answer Set Programming is that all atoms belonging to a stable model must be founded. That is, an atom must not only be true but provably true. This can be made precise by means of the constructive logic of Here-and-There, whose equilibrium models correspond to stable models. One way to look at foundedness is to regard Boolean truth values as ordered by letting true be greater than false. Then, each Boolean variable takes the smallest truth values that can be proven for it. This idea was generalized by Aziz to ordered domains and applied to constraint satisfaction problems. As before, the idea is that a, say integer, variable gets only assigned to the smallest integer that can be justified. In this paper, we present a logical reconstruction of Aziz' idea in the setting of the logic of Here-and-There. More precisely, we start by defining the logic of Here-and-There with lower bound founded variables along with its equilibrium models and elaborate upon their formal properties. We then define a logic program fragment dealing with linear constraints over integers and analyze it in terms of concepts from logic programming. Finally, we compare our approach with related ones and sketch future work.

1 Motivation

A distinguishing feature of Answer Set Programming (ASP; Baral 2003) is that all atoms belonging to a stable model must be founded. That is, an atom must not only be true but provably true. This can be made precise by means of the constructive logic of Here-and-There (HT; Heyting 1930), whose equilibrium models correspond to stable models (Pearce 2006). One way to look at foundedness is to regard Boolean truth values as ordered by letting true be greater than false. Then, each Boolean variable takes the smallest truth value that can be proven for it. This idea was generalized in (Aziz 2015) to ordered domains and applied to constraint satisfaction problems. As before, the idea is that a, say integer, variable gets only assigned to the smallest integer that can be justified. We refer to this idea by calling it *foundedness*. Note that ASP follows the rationality principle, which says that one shall only believe in things one is forced to. In the propositional case this principle amounts to foundedness, whereas for rules like $x \geq 42$ there are at least two ways of understanding. First, one believe in any value greater or equal than 42 for x. Second, one believe in value 42 for x if there is nothing else forcing to believe more than this. The latter one corresponds to our understanding of foundedness.

The literature of ASP contains several approaches dealing with atoms containing variables over non-Boolean domains, among them (Baselice, Bonatti, and Gelfond 2005), (Janhunen et al. 2017) and (Cabalar et al. 2016), but these approaches do not address foundedness in our sense. For instance, Constraint ASP ($CASP$) approaches like (Baselice, Bonatti, and Gelfond 2005) allow atoms with variables over non-Boolean domains in the body of a rule only. Thus, these atoms and the values of non-Boolean variables cannot be founded in terms of ASP.

Approaches like (Janhunen et al. 2017) and (Cabalar et al. 2016) allow any kind of atoms in heads and bodies. This allows atoms with variables over non-Boolean domains to be founded but their variables are not necessarily assigned to the smallest value that can be justified. Since in the approach of (Cabalar et al. 2016) atoms as well as the values of its variables are founded and defaults are possible, one could think about to use defaults or minimization to achieve foundedness. For instance, $x = 1 \leftarrow \neg(x \neq 1)$ assigns value 1 to x by default. If we add fact $x = 3$, then we deactivate the default and assign value 3 to x. Similarly, $x = 0 \leftarrow \neg(x > 0)$ assigns the value 0 by default. However, in general assigning a minimal value by default cannot be done by rules as the above. To point out the difference of foundedness and founded atoms, the following examples illustrate that minimizing assigned values does not restore foundedness either. Consider the rules

$$x \geq 0 \qquad y \geq 0 \qquad x \geq 42 \leftarrow y < 42 \quad (1)$$

The approach of (Cabalar et al. 2016) leads to solutions that assign values greater or equal than 42 to x and values greater or equal than 0 to y or vice versa, respectively. Thus, the two solutions with minimal values assign 42 to x and 0 to y and the other way around. Note that only the first one respects foundedness, since there is no reason to assign a value greater than 0 to y. Now, consider the rules

$$x \geq 1 \qquad x \geq 42 \leftarrow \neg(x \leq 1) \quad (2)$$

We expect two solutions in terms of foundedness. One assigns the value 1 to x and the other assigns value 42 to x, since a value greater than 1 forces the derivation of value 42. The

rules of (2) give us no reason to derive a value greater than 42. In contrast, the approach presented in (Cabalar et al. 2016) yield an intuitive understanding assigning value 1 or a value greater or equal than 42 to x. That is, the corresponding solution with the minimal value assigned to x assigns 1 to x. The second equally founded solution is not obtained.

The existing approach regarding foundedness of (Aziz 2015) behaves counter intuitive. For instance, consider rule $p \leftarrow \neg p$. Then, Aziz' approach yields a solution where p holds instead of no solution as expected in terms of ASP. To this end, we present in the following a logical reconstruction of Aziz' idea of foundedness in the setting of the logic of Here-and-There. More precisely, we start by defining the logic of Here-and-There with lower bound founded variables, short HT_{LB}, along with its equilibrium models. We elaborate upon the formal properties of HT_{LB} regarding persistence, negation and strong equivalence. Furthermore, we point out the relation of HT_{LB} to HT, and show that our approach corresponds to a straightforward extension of Ferraris' stable model semantics (Ferraris 2005). We then define a logic program fragment dealing with linear constraints over integers and analyze it in terms of concepts from logic programming. Finally, we compare our approach with related ones, to point out the benefits of HT_{LB} and sketch future work.

2 Background

Let $\mathcal{A}$ be the set of propositional atoms. A formula φ is a combination of atoms by logical connectives $\bot$, $\wedge$, $\vee$, and $\leftarrow$. As usual, we define $\top \stackrel{\text{def}}{=} \bot \rightarrow \bot$ and $\neg\varphi \stackrel{\text{def}}{=} \varphi \rightarrow \bot$. A theory is a set of formulas.

We denote an interpretation over $\mathcal{A}$ by $I \subseteq \mathcal{A}$ and an HT-interpretation over $\mathcal{A}$ by $\langle H, T\rangle$ where $H \subseteq T \subseteq \mathcal{A}$ are interpretations. Since we want to abstract from the specific form of atoms in the following sections, we rely upon denotations for fixing their semantics. A *denotation* of atoms in $\mathcal{A}$ is a function $[\![\cdot]\!]_{\mathcal{A}} : \mathcal{A} \rightarrow 2^{\mathcal{A}}$ mapping atoms in $\mathcal{A}$ to sets of interpretations over $\mathcal{A}$. Accordingly, $[\![p]\!]_{\mathcal{A}} \stackrel{\text{def}}{=} \{I \mid p \in I\}$ represents the sets of interpretations where atom p holds.

With it, we next define satisfaction of formulas in HT.

Definition 1 *Let $\langle H, T\rangle$ be an HT-interpretation over $\mathcal{A}$ and φ a propositional formula over $\mathcal{A}$. Then, $\langle H, T\rangle$ satisfies φ, written $\langle H, T\rangle \models \varphi$, if the following conditions hold:*

1. $\langle H, T\rangle \not\models \bot$
2. $\langle H, T\rangle \models p$ *iff* $H \in [\![p]\!]_{\mathcal{A}}$ *for propositional atom* $p \in \mathcal{A}$
3. $\langle H, T\rangle \models \varphi_1 \wedge \varphi_2$ *iff* $\langle H, T\rangle \models \varphi_1$ *and* $\langle H, T\rangle \models \varphi_2$
4. $\langle H, T\rangle \models \varphi_1 \vee \varphi_2$ *iff* $\langle H, T\rangle \models \varphi_1$ *or* $\langle H, T\rangle \models \varphi_2$
5. $\langle H, T\rangle \models \varphi_1 \rightarrow \varphi_2$ *iff* $\langle I, T\rangle \not\models \varphi_1$ *or* $\langle I, T\rangle \models \varphi_2$ *for both* $I \in \{H, T\}$

As usual, we call $\langle H, T\rangle$ an HT-model of a theory Γ, if $\langle H, T\rangle \models \varphi$ for all φ in Γ. The usual definition of HT satisfaction (cf. Pearce 2006) is obtained by replacing Condition 2 above by

2': $\langle H, T\rangle \models p$ *iff* $p \in H$ *for propositional atom* $p \in \mathcal{A}$

It is easy to see that both definitions of HT satisfaction coincide.

Proposition 1 *Let $\langle H, T\rangle$ be an HT-interpretation and φ a formula over $\mathcal{A}$. Then, $\langle H, T\rangle \models \varphi$ iff $\langle H, T\rangle \models \varphi$ by replacing Condition 2 by 2'.*

As usual, an equilibrium model of a theory Γ is a (total) HT-interpretation $\langle T, T\rangle$ such that $\langle T, T\rangle \models \Gamma$ and there is no $H \subset T$ such that $\langle H, T\rangle \models \Gamma$.

3 Lower Bound Founded Logic of Here-and-There

In what follows, we introduce the logic of Here-and-There with lower bound founded variables, short HT_{LB} and elaborate on some formal properties regarding satisfaction. We discuss the relation of complements of atoms regarding negation and we point out the relation between HT_{LB} and HT as well as a straightforward extension of Ferraris' stable model semantics.

3.1 HT_{LB} and its Properties

The language of HT_{LB} is defined over a set of atoms $\mathcal{A}_{\mathcal{X}}$ comprising variables, $\mathcal{X}$, and constants over an ordered domain $(\mathcal{D}, \succeq)$. For simplicity, we assume that each element of $\mathcal{D}$ is uniquely represented by a constant and abuse notation by using $\mathcal{D}$ to refer to the set of constants. Similarly, we identify $\succeq$ with its syntactic representative. The specific syntax of atoms is left open but assumed to refer to elements of $\mathcal{X}$ and $\mathcal{D}$. The only requirement is that we assume that an atom depends on a distinguished subset of variables of $\mathcal{X}$. An atoms can be understood to hold or not once all variables depending on it are substituted by domain elements. Intuitively, variables not occurring in an atom are understood as irrelevant for the atom evaluation. Examples of ordered domains are $(\{0, 1, 2, 3\}, \geq)$ and $(\mathbb{Z}, \geq)$, respectively; corresponding atoms are $x \geq 42$ and $x = y$. A formula φ is a propositional combination of atoms and logical connectives $\bot, \wedge, \vee, \rightarrow$. As usual, we define $\top \stackrel{\text{def}}{=} \bot \rightarrow \bot$ and $\neg\varphi \stackrel{\text{def}}{=} \varphi \rightarrow \bot$. A theory is a set of formulas. For instance, '$y < 42 \wedge \neg(x = y) \rightarrow x \geq 42$' is a formula. Let $vars(\varphi) \subseteq \mathcal{X}$ be the set of variables and $atoms(\varphi) \subseteq \mathcal{A}_{\mathcal{X}}$ the atoms occurring in a formula φ.

For capturing partiality, we introduce a special domain element $\boldsymbol{u}$, standing for *undefined*, and extend $(\mathcal{D}, \succeq)$ to $(\mathcal{D}_{\boldsymbol{u}}, \succeq_{\boldsymbol{u}})$ where $\mathcal{D}_{\boldsymbol{u}} \stackrel{\text{def}}{=} \mathcal{D} \cup \{\boldsymbol{u}\}$ and $\succeq_{\boldsymbol{u}} \stackrel{\text{def}}{=} \succeq \cup \{(c, \boldsymbol{u}) \mid c \in \mathcal{D}\}$. With it, we define a (partial) *valuation* over $\mathcal{X}, \mathcal{D}$ as a function $v : \mathcal{X} \rightarrow \mathcal{D}_{\boldsymbol{u}}$ mapping each variable to a domain value or undefined. For comparing valuations by set-based means, we alternatively represent them by subsets of $\mathcal{X} \times \mathcal{D}$. Basically, any function v is a set of pairs (x, c) such that $v(x) = c$ for $c \in \mathcal{D}$. In addition, we view a pair (x, c) as $x \succeq c$ and add its downward closure $(x{\downarrow}c) \stackrel{\text{def}}{=} \{(x, d) \mid c, d \in \mathcal{D}, c \succeq d\}$. Given this, a valuation v is represented by the set $\bigcup_{v(x)=c, x\in\mathcal{X}}(x{\downarrow}c)$.[1] As an example, consider variables x and y over domain $(\{0, 1, 2, 3\} \cup \{\boldsymbol{u}\}, \geq_{\boldsymbol{u}})$. The valuation $v = \{x \mapsto 2, y \mapsto 0\}$ can be represented by $v = (x{\downarrow}2) \cup (y{\downarrow}0) = \{(x, 0), (x, 1), (x, 2), (y, 0)\}$. Then, $v' = \{x \mapsto 1, y \mapsto \boldsymbol{u}\}$, viz. $\{(x, 0), (x, 1)\}$ in set notation, can be regarded as "smaller" than v because $v' \subseteq v$. The

[1] Note that $(x{\downarrow}\boldsymbol{u}) = \emptyset$, since $\boldsymbol{u} \notin \mathcal{D}$.

comparison of two valuations v and v' by their set-based means using $\subseteq$ amounts to a twofold comparison. That is, v and v' are compared regarding the occurrence of variables and their particular values wrt $\succeq$. We let $\mathfrak{V}_{\mathcal{X},\mathcal{D}}$ stand for the set of valuations over $\mathcal{X}$ and $\mathcal{D}$.

We define the satisfaction of formulas over $\mathcal{A}_\mathcal{X}$ wrt *atom denotations* over $\mathcal{X}, \mathcal{D}$, which are functions $[\![\cdot]\!]_{\mathcal{X},\mathcal{D}} : \mathcal{A}_\mathcal{X} \to 2^{\mathfrak{V}_{\mathcal{X},\mathcal{D}}}$ mapping atoms to sets of valuations. Let a be an atom of $\mathcal{A}_\mathcal{X}$ and $[\![a]\!]_{\mathcal{X},\mathcal{D}}$ its denotation. Then, $[\![a]\!]_{\mathcal{X},\mathcal{D}}$ is the set of valuations v so that a holds. Since a depends on variables $vars(a) \subseteq \mathcal{X}$, we have for each $v \in [\![a]\!]$ and valuation v' with $v(x) = v'(x)$ for $x \in vars(a)$ that $v' \in [\![a]\!]$. Intuitively, values of $\mathcal{X} \setminus vars(a)$ can vary freely without changing the membership of a valuation to $[\![a]\!]$. For simplicity, we drop indices $\mathcal{X}, \mathcal{D}$ whenever clear from context.

For instance, interpreting the atoms $x \geq 42$, $42 \geq 0$ and $0 \geq 42$ over $(\mathbb{Z}, \geq)$ yields the following denotations:

$$[\![x \geq 42]\!] \stackrel{\text{def}}{=} \{v \mid v(x) \geq 42\}$$
$$[\![42 \geq 0]\!] \stackrel{\text{def}}{=} \mathfrak{V}$$
$$[\![0 \geq 42]\!] \stackrel{\text{def}}{=} \emptyset.$$

In particular, $[\![x \geq 42]\!]$ is the set of valuations where x is assigned to a value greater or equal than 42 and all variables in $\mathcal{X} \setminus vars(x \geq 42)$ take any value of $\mathcal{D}_{\boldsymbol{u}}$, eg $(x \downarrow 45)$ and $(x \downarrow 45) \cup (y \downarrow 0)$ for $y \in \mathcal{X} \setminus vars(x \geq 42)$ are possible valuations. Interestingly, atoms like $x \succeq x$ with $[\![x \succeq x]\!] = \{v \mid v(x) \neq \boldsymbol{u}\}$ force variables to be defined over $\mathcal{D}$ per definition of $\succeq$. A valuation v is defined for a set of variables $\mathcal{Y} \subseteq \mathcal{X}$ if $v(x) \neq \boldsymbol{u}$ for all $x \in \mathcal{Y}$.

We define an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$ as a pair $\langle h, t \rangle$ of valuations over $\mathcal{X}, \mathcal{D}$ with $h \subseteq t$. We define satisfaction of a formula wrt an HT_{LB}-valuation as follows.

Definition 2 *Let $\langle h, t \rangle$ be an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$ and φ be a formula over $\mathcal{A}_\mathcal{X}$. Then, $\langle h, t \rangle$ satisfies φ, written $\langle h, t \rangle \models \varphi$, if the following holds:*

1. $\langle h, t \rangle \not\models \bot$
2. $\langle h, t \rangle \models a$ *iff* $v \in [\![a]\!]_{\mathcal{X},\mathcal{D}}$ *for atom* $a \in \mathcal{A}_\mathcal{X}$ *and for both* $v \in \{h, t\}$
3. $\langle h, t \rangle \models \varphi_1 \wedge \varphi_2$ *iff* $\langle h, t \rangle \models \varphi_1$ *and* $\langle h, t \rangle \models \varphi_2$
4. $\langle h, t \rangle \models \varphi_1 \vee \varphi_2$ *iff* $\langle h, t \rangle \models \varphi_1$ *or* $\langle h, t \rangle \models \varphi_2$
5. $\langle h, t \rangle \models \varphi_1 \to \varphi_2$ *iff* $\langle v, t \rangle \not\models \varphi_1$ *or* $\langle v, t \rangle \models \varphi_2$ *for both* $v \in \{h, t\}$

As usual, we call $\langle h, t \rangle$ an HT_{LB}-model of a theory Γ, if $\langle h, t \rangle \models \varphi$ for all φ in Γ. For a simple example, consider the theory containing atom $x \geq 42$ only. Then, every HT_{LB}-valuation $\langle h, t \rangle$ with $h, t \in [\![x \geq 42]\!]$ is an HT_{LB}-model of $x \geq 42$. Note that, different to HT, satisfaction of atoms in HT_{LB} forces satisfaction in both h and t, instead of h only. We discuss this in detail in Section 3.4.

Our first result shows that the characteristic properties of persistence and negation hold as well when basing satisfaction on valuations and denotations.

Proposition 2 *Let $\langle h, t \rangle$ and $\langle t, t \rangle$ be HT_{LB}-valuations over $\mathcal{X}, \mathcal{D}$, and φ be a formula over $\mathcal{A}_\mathcal{X}$. Then,*

1. $\langle h, t \rangle \models \varphi$ *implies* $\langle t, t \rangle \models \varphi$, *and*
2. $\langle h, t \rangle \models \varphi \to \bot$ *iff* $\langle t, t \rangle \not\models \varphi$.

Persistence implies that all atoms satisfied by $\langle h, t \rangle$ are also satisfied by $\langle t, t \rangle$. To make this precise, let $At(\langle h, t \rangle) \stackrel{\text{def}}{=} \{a \in \mathcal{A}_\mathcal{X} \mid h \in [\![a]\!] \text{ and } t \in [\![a]\!]\}$ be the set of atoms satisfied by $\langle h, t \rangle$.

Proposition 3 *Let $\langle h, t \rangle$ and $\langle t, t \rangle$ be HT_{LB}-valuations over $\mathcal{X}, \mathcal{D}$. Then, $At(\langle h, t \rangle) \subseteq At(\langle t, t \rangle)$*

Finally, we define an equilibrium model in HT_{LB}.

Definition 3 *An HT_{LB}-valuation $\langle t, t \rangle$ over $\mathcal{X}, \mathcal{D}$ is an HT_{LB}-equilibrium model of a theory Γ iff $\langle t, t \rangle \models \Gamma$ and there is no $h \subset t$ such that $\langle h, t \rangle \models \Gamma$.*

We refer an HT_{LB}-equilibrium model $\langle t, t \rangle$ of Γ as an HT_{LB}-stable model t of Γ. Let us reconsider the theory containing atom $x \geq 42$ only. Then, $t = (x \downarrow 42)$ is an HT_{LB}-stable model of $x \geq 42$, since $t \in [\![x \geq 42]\!]$ and there is no $h \subset t$ with $h \in [\![x \geq 42]\!]$. In contrast, neither HT_{LB}-model $\langle t', t' \rangle$ with $t' = (x \downarrow 42) \cup (y \downarrow 0)$ nor $\langle t'', t'' \rangle$ with $t'' = (x \downarrow 53)$ are HT_{LB}-stable models since t is a proper subset of both and $\langle t, t' \rangle \models x \geq 42$ as well as $\langle t, t'' \rangle \models x \geq 42$ holds. Hence, HT_{LB}-stable models make sure that each variable is assigned to its smallest founded value and does not take any value of possible valuations of corresponding denotations.

Note that HT_{LB}-equilibrium models induce the non-monotonic counterpart of the monotonic logic of HT_{LB}. Following well-known patterns, we show that HT_{LB} allows us to decide strong equivalence wrt HT_{LB}-equilibrium models.

Proposition 4 (Strong Equivalence) *Let Γ_1, Γ_2 and Γ be theories over $\mathcal{A}_\mathcal{X}$. Then, theories $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ have the same HT_{LB}-stable models for every theory Γ iff Γ_1 and Γ_2 have the same HT_{LB}-models.*

The idea is to prove the if direction by proving its contraposition, and the only if direction by proving its straightforward implication. The contraposition assumes that there exists an HT_{LB}-valuation that satisfies Γ_1 but not Γ_2 which implies that the stable models of $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ do not coincide. There are two cases to construct Γ in a way that $\Gamma_1 \cup \Gamma$ has a stable model which is not a stable model of $\Gamma_2 \cup \Gamma$ and the other way around, respectively. Let us consider an example to illustrate the idea of the construction of Γ. Let $h = (x \downarrow 0)$ and $t = (x \downarrow 2) \cup (y \downarrow 0)$ be HT_{LB}-valuation over $\{x, y\}, \{0, 1, 2, 3\}$ with $\langle h, t \rangle \models \Gamma_1$ and $\langle h, t \rangle \not\models \Gamma_2$. For the first case assume that $\langle t, t \rangle \not\models \Gamma_2$. Since t cannot be a model of $\Gamma_2 \cup \Gamma$ by assumption, we construct Γ in a way that t is a stable model of $\Gamma_1 \cup \Gamma$. Hence, let $\Gamma = \{z \succeq c \mid (z, c) \in t\} = \{x \succeq 0, x \succeq 1, x \succeq 2, y \succeq 0\}$ be the theory with the only stable model t. By persistence of $\langle h, t \rangle$ wrt Γ_1 and construction of Γ we get that t is a stable model of $\Gamma_1 \cup \Gamma$ but not of $\Gamma_2 \cup \Gamma$. For the second case we assume that $\langle t, t \rangle \models \Gamma_2$. Now we construct Γ in a way that t is a stable model of $\Gamma_2 \cup \Gamma$ but not of $\Gamma_1 \cup \Gamma$. By assumption we have that $\langle h, t \rangle \models \Gamma_1$ and $\langle h, t \rangle \not\models \Gamma_2$ as well as $\langle t, t \rangle \models \Gamma_2$, thus we want to have $\langle h, t \rangle$ and $\langle v, v' \rangle$ with $t \subseteq v \subseteq v'$ as the only models of Γ. Hence, let $\Gamma = \Gamma' \cup \Gamma''$ with $\Gamma' = \{z \succeq c \mid (z, c) \in h\} = \{x \succeq 0\}$ the theory that is satisfied by everything that is greater or equal than h, and $\Gamma'' = \{z \succeq t(z) \to z' \succeq t(z'),\ z \succeq c \to z \succeq t(z) \mid (z, c), (z, t(z)), (z', t(z')) \in t \setminus h, z \neq z'\} = \{x \succeq 2 \to$

$y \succeq 0,\ y \succeq 0 \rightarrow x \succeq 2,\ x \succeq 1 \rightarrow x \succeq 2,\ x \succeq 2 \rightarrow x \succeq 2\}$ the theory which derives values of t for each v'' with $h \subset v'' \subset t$. Since $\langle h, t\rangle \not\models \Gamma_2$ and by construction of Γ we get that t is a stable model of $\Gamma_2 \cup \Gamma$ but not of $\Gamma_1 \cup \Gamma$.

3.2 Negation in HT_{LB}

In the following, we elaborate on complements of atoms and its relation to negation, since $\mathcal{A}_\mathcal{X}$ may contain atoms like $x \geq 42$ and $x < 42$. Intuitively, one could expect that the strong negation of an atom holds whenever the atom itself does not hold. This can be easily expressed by defining the complement of valuations of an atom denotation. More formally, we characterize the complement $\overline{a}$ of atom a by its denotation $[\![\, \overline{a} \,]\!] \stackrel{\text{def}}{=} 2^\mathfrak{V} \setminus [\![\, a \,]\!]$.

To illustrate that the simple complement of an atom is not sufficient to yield something similar to strong negation let us take a closer look on propositional atoms in HT_{LB}. For mimicking Boolean truth values, we consider the domain $(\{\mathbf{t}, \mathbf{f}\}, \{\mathbf{t} \succeq \mathbf{f}\})$. Then, the denotation of propositional atoms in HT_{LB} can be defined as follows: $[\![\, p = \mathbf{t} \,]\!]_{\mathcal{A},\{\mathbf{t},\mathbf{f}\}} \stackrel{\text{def}}{=} \{v \mid v(p) = \mathbf{t}\}$ and $[\![\, p = \mathbf{f} \,]\!]_{\mathcal{A},\{\mathbf{t},\mathbf{f}\}} \stackrel{\text{def}}{=} \{v \mid v(p) = \mathbf{f}\}$. Note that $p = \mathbf{t}$ and $p = \mathbf{f}$ are regarded as strong negations of each other, as in standard case (Gelfond and Lifschitz 1990); its weak negations are given by $\neg(p = \mathbf{t})$ and $\neg(p = \mathbf{f})$, respectively. For instance, the complement $\overline{p = \mathbf{t}}$ is characterized by denotation $[\![\, \overline{p = \mathbf{t}} \,]\!] = 2^\mathfrak{V} \setminus [\![\, p = \mathbf{t} \,]\!] = \{v \mid v(p) \neq \mathbf{t}\}$. Note that this complement allows valuations v with $v(p) = \boldsymbol{u}$, which does not match $p = \mathbf{f}$.

To this end, we define another complement to exclude assigning value undefined to variables of the atom. First, we define a denotation $[\![\, a \,]\!]$ of an atom a as strict if each $v \in [\![\, a \,]\!]$ is defined for $vars(a)$. Then, we characterize the strict complement $\overline{a}^s$ of atom a by the strict denotation $[\![\, \overline{a}^s \,]\!] \stackrel{\text{def}}{=} 2^\mathfrak{V} \setminus ([\![\, a \,]\!] \cup \{v \mid v(x) = \boldsymbol{u} \text{ for some } x \in vars(a)\})$. Informally, the strict complement of an atom holds whenever all variables are defined and the atom itself does not hold. That is, atoms $p = \mathbf{f}$ and $p = \mathbf{t}$ are strict complements of each other.

More generally, an atom with strict denotation and its strict complement can be regarded as being strongly negated to each other. For instance, consider atom $x \geq 42$ and its strict denotation $[\![\, x \geq 42 \,]\!] = \{v \mid v(x) \geq 42\}$. Then, its strict complement $\overline{x \geq 42}^s$ is defined by $[\![\, \overline{x \geq 42}^s \,]\!] = \{v \mid \boldsymbol{u} \neq v(x) < 42\}$. As in the Boolean case, the strict complement $\overline{x \geq 42}^s$ can be seen as the strong negation of $x \geq 42$.

To make the relation of complements and negation precise, let us define entailments. A theory (or a single formula) Γ over $\mathcal{A}_\mathcal{X}$ entails a formula φ over $\mathcal{A}_\mathcal{X}$, written $\Gamma \models \varphi$, when all HT_{LB}-models of Γ are HT_{LB}-models of φ. Then, we have the following result.

Proposition 5 *Let a be an atom over $\mathcal{A}_\mathcal{X}$, and $\overline{a}$ and $\overline{a}^s$ its complement and its strict complement over $\mathcal{A}_\mathcal{X}$, respectively. Then, $\overline{a}^s \models \overline{a}$ and $\overline{a} \models \neg a$.*

This implies that the strict complement $\overline{a}^s$ of an atom a implies its negation $\neg a$, just as strong negation implies weak negation in the standard case (Pearce 2006). To illustrate that in general the negation of an atom does not entail its complement ($\neg a \not\models \overline{a}$), let us consider atom $x \leq 42$ with strict denotation $[\![\, x \leq 42 \,]\!] = \{v \mid \boldsymbol{u} \neq v(x) \leq 42\}$. Then, the complement $\overline{x \leq 42}$ is defined by denotation $[\![\, \overline{x \leq 42} \,]\!] = 2^\mathfrak{V} \setminus [\![\, x \leq 42 \,]\!] = \{v \mid v(x) = \boldsymbol{u} \text{ or } v(x) > 42\}$. For valuations $h = (x \downarrow 42)$ and $t = (x \downarrow 50)$ we have that $\langle h, t\rangle \models \neg(x \leq 42)$ since $(x{\downarrow}50) \notin [\![\, x \leq 42 \,]\!]$. In contrast, $\langle h, t\rangle \models \overline{x \leq 42}$ does not hold, since $(x{\downarrow}42) \notin [\![\, \overline{x \leq 42} \,]\!]$. Thus, the complement $\overline{a}$ of an atom a can be seen as a kind of negation in between of strong and weak negation.

3.3 HT_{LB} versus HT

Analogously to (Cabalar et al. 2016), we next show that HT can be seen as a special case of HT_{LB}.

Note that both types of denotations $[\![\, p \,]\!]_\mathcal{A}$ and $[\![\, p = \mathbf{t} \,]\!]_{\mathcal{A},\{\mathbf{t}\}}$ of a propositional atom p collect interpretations and valuations assigning true to p, respectively. To this end, we define a transformation τ relating each propositional atom p with corresponding atom $p = \mathbf{t}$ by $\tau(p) \stackrel{\text{def}}{=} p = \mathbf{t}$. Let Γ be a propositional theory, then $\tau(\Gamma)$ is obtained by substituting each $p \in atoms(\Gamma)$ by $\tau(p)$. Moreover, we extend τ to interpretations I by $\tau(I) \stackrel{\text{def}}{=} \{(p, \mathbf{t}) \mid p \in I\}$ to obtain a corresponding valuation over $\mathcal{A}, \{\mathbf{t}\}$. The next proposition establishes that HT can be seen as a special case of HT_{LB}.

Proposition 6 *Let Γ be a theory over propositional atoms $\mathcal{A}$ and $\langle H, T\rangle$ an HT-interpretation over $\mathcal{A}$. Let $\tau(\Gamma)$ be a theory over atoms $\{p = \mathbf{t} \mid p \in \mathcal{A}\}$ and $\langle \tau(H), \tau(T)\rangle$ an HT_{LB}-valuation over $\mathcal{A}, \{\mathbf{t}\}$. Then, $\langle H, T\rangle \models \Gamma$ iff $\langle \tau(H), \tau(T)\rangle \models \tau(\Gamma)$.*

This can be generalized to any arbitrary singleton domain $\{d\}$ and corresponding atoms $p = d$ and the relationship still holds.

We obtain the following results relating HT_{LB} and HT:

Proposition 7 *Let Γ be a theory over $\mathcal{A}_\mathcal{X}$ and $\langle h, t\rangle$ an HT_{LB}-model of Γ over $\mathcal{X}, \mathcal{D}$. Then, $\langle At(\langle h, t\rangle), At(\langle t, t\rangle)\rangle$ is an HT-model of Γ over $\mathcal{A}_\mathcal{X}$.*

That is, the collected atoms satisfied by an HT_{LB}-model of Γ can be seen as an HT-model of Γ by interpreting $\mathcal{A}_\mathcal{X}$ as propositional atoms. For instance, consider the theory containing only atom $x \neq y$ and its denotation $[\![\, x \neq y \,]\!] \stackrel{\text{def}}{=} \{v \mid \boldsymbol{u} \neq v(x) \neq v(y) \neq \boldsymbol{u}\}$. Let $h = (x{\downarrow}0) \cup (y{\downarrow}4)$ and $t = (x{\downarrow}0) \cup (y{\downarrow}42)$ be valuations and hence $At(\langle h, t\rangle) = At(\langle t, t\rangle) = \{x \neq y\}$ interpretations. Then, $\langle h, t\rangle \models x \neq y$ in HT_{LB} and $\langle At(\langle h, t\rangle), At(\langle t, t\rangle)\rangle \models x \neq y$ in HT.

Furthermore, we relate tautologies in HT and HT_{LB}.

Proposition 8 *Let φ be a tautology over $\mathcal{A}$ and φ' a formula over $\mathcal{A}_\mathcal{X}$ obtained by replacing all atoms in φ by atoms of $\mathcal{A}_\mathcal{X}$. Then, φ' is a tautology in HT_{LB}.*

That is, tautologies in HT are independent of any form of atoms.

3.4 HT_{LB}-stable versus Ferraris-style stable models

As mentioned, in Definition 2 satisfaction of atoms differs from HT by forcing satisfaction in both h and t, instead of h only. This is necessary to satisfy persistence in HT_{LB}. In fact, let HT_{LB}-valuation $\langle h, t\rangle$ satisfy atom a in $\mathcal{A}_\mathcal{X}$, and by persistence HT_{LB}-valuation $\langle t, t\rangle$ satisfies a as well, but

not necessarily each HT_{LB}-valuation $\langle v, t\rangle$ with $h \subset v \subset t$ satisfies a. For instance, consider atom $x \neq 42$ with $[\![x \neq 42]\!] \stackrel{\text{def}}{=} \{v \mid \boldsymbol{u} \neq v(x) \neq 42\}$. Let $h = (x \downarrow 0)$ and $t = (x \downarrow 53)$ be valuations. Then, $\langle h, t\rangle \models x \neq 42$ and $\langle t, t\rangle \models x \neq 42$, but for $v = (x \downarrow 42)$ with $h \subset v \subset t$ we have $\langle v, t\rangle \not\models x \neq 42$.

A question that arises now from the above is whether HT_{LB} behaves as expected in terms of stable models semantics. To this end, we give a straightforward definition of classical satisfaction and of the reduct put by Ferraris in (Ferraris 2005) in our setting and show that equilibrium models correspond to stable models according to the resulting Ferraris'-like stable model semantics. We define the counterpart of classical satisfaction as follows.

Definition 4 *Let t be a valuation over $\mathcal{X}, \mathcal{D}$ and φ a formula over $\mathcal{A}_\mathcal{X}$. Then, t satisfies φ, written $t \models_{cl} \varphi$, if the following holds:*

1. $t \not\models_{cl} \bot$
2. $t \models_{cl} a$ *iff* $t \in [\![a]\!]_{\mathcal{X},\mathcal{D}}$ *for atom* $a \in \mathcal{A}_\mathcal{X}$
3. $t \models_{cl} \varphi_1 \wedge \varphi_2$ *iff* $t \models_{cl} \varphi_1$ *and* $t \models_{cl} \varphi_2$
4. $t \models_{cl} \varphi_1 \vee \varphi_2$ *iff* $t \models_{cl} \varphi_1$ *or* $t \models_{cl} \varphi_2$
5. $t \models_{cl} \varphi_1 \rightarrow \varphi_2$ *iff* $t \not\models_{cl} \varphi_1$ *or* $t \models_{cl} \varphi_2$.

We call t a classical model of a theory Γ, if $t \models_{cl} \varphi$ for all φ in Γ. We define a Ferraris-like reduct, short F-reduct, wrt atoms $\mathcal{A}_\mathcal{X}$ as follows.

Definition 5 *Let φ be a formula over $\mathcal{A}_\mathcal{X}$ and t a valuation over $\mathcal{X}, \mathcal{D}$. Then, the F-reduct of φ over t, written φ^t, is given by*

$$\varphi^t \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } t \not\models_{cl} \varphi \\ a & \text{if } t \models_{cl} \varphi \text{ and } \varphi = a \text{ atom of } \mathcal{A}_\mathcal{X} \\ {\varphi_1}^t \otimes {\varphi_2}^t & \text{if } t \models_{cl} \varphi \text{ and } \varphi = (\varphi_1 \otimes \varphi_2) \\ & \text{for } \otimes \in \{\wedge, \vee, \rightarrow\} \end{cases}$$

For theory Γ and HT_{LB}-valuation t, we define $\Gamma^t \stackrel{\text{def}}{=} \{\varphi^t \mid \varphi \in \Gamma\}$. Note that in case of propositional atoms the F-reduct corresponds to Ferraris' reduct.

We define an F-stable model as expected according to classical satisfaction and the F-reduct above.

Definition 6 *A valuation t over $\mathcal{X}, \mathcal{D}$ is an F-stable model of theory Γ over $\mathcal{A}_\mathcal{X}$ iff $t \models_{cl} \Gamma^t$ and there is no $h \subset t$ such that $h \models_{cl} \Gamma^t$.*

The next propositions shows that models in HT_{LB} can be alternatively characterized in the style of Ferraris, rephrasing (Ferraris 2005, Lemma 1):

Proposition 9 *Let $\langle h, t\rangle$ be an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$ and Γ a theory over $\mathcal{A}_\mathcal{X}$. Then, $h \models_{cl} \Gamma^t$ iff $\langle h, t\rangle \models \Gamma$.*

As a special case, we obtain that every HT_{LB}-stable model corresponds to an F-stable model and vice versa.

Corollary 1 *Let t be a valuation over $\mathcal{X}, \mathcal{D}$ and Γ a theory over $\mathcal{A}_\mathcal{X}$. Then, t is an HT_{LB}-stable model of Γ iff t is an F-stable model of Γ.*

The last two results have shown that our logic follows well known patterns wrt different representations of stable models.

4 Bound Founded Programs with Linear Constraints

In this section, we focus on atoms representing linear constraints over integers and analyze them in terms of concepts known from ASP. Due to space limitations, we present proofs and some preliminaries needed for the following results in an extended version of this work. We illustrate the modelling capabilities of this fragment of HT_{LB} on an example of error diagnosis.

4.1 Programs and its Properties

Reconsider the ordered domain of integers $(\mathbb{Z}, \geq)$. We define a linear constraint atom as

$$\sum_{i=1}^{m} w_i x_i \prec k$$

where $w_i, k \in \mathbb{Z}$ are constants, $x_i \in \mathcal{X}$ are distinct variables, and $\prec \in \{\geq, \leq, \neq, =\}$[2] is a binary relation. By $\mathcal{L}_\mathcal{X}$ we denote the set of linear constraint atoms wrt $\mathcal{X}$ and $\mathbb{Z}$. The denotation of a linear constraint atom is given by

$$[\![\sum_{i=1}^{m} w_i x_i \prec k]\!] \stackrel{\text{def}}{=} \{v \mid \sum_{i=1}^{m} w_i v(x_i) \prec k,\ v(x_i) \neq \boldsymbol{u}\}.$$

A linear constraint atom a and its negation $\neg a$ are called linear constraint literals. In the following, we just say atoms and literals.

We define logic programs as follows.

Definition 7 *A formula over $\mathcal{L}_\mathcal{X}$ is called a rule if it is of form*

$$a_1 \vee \cdots \vee a_n \leftarrow l_1 \wedge \cdots \wedge l_{n'} \tag{3}$$

where a_i is an atom for $1 \leq i \leq n$ and l_j is a literal for $1 \leq j \leq n'$ both over $\mathcal{L}_\mathcal{X}$.

A logic program is a theory of rules of form (3). Following logic programming syntax, we use ',' and ';' as alternative representations of $\wedge$ and $\vee$, respectively. Moreover, in this context we write $\varphi_1 \leftarrow \varphi_2$ for $\varphi_2 \rightarrow \varphi_1$ for formulas φ_1 and φ_2. Examples of programs over $\mathcal{L}_\mathcal{X}$ are given in the introduction.

Let r be a rule of form (3). Then, we define by $head(r) \stackrel{\text{def}}{=} \{a_i \mid 1 \leq i \leq n\}$ and $body(r) \stackrel{\text{def}}{=} \{l_j \mid 1 \leq j \leq n'\}$ the set of literals of the left and right hand side of r, respectively. Whenever $body(r) = \emptyset$, then we drop $\leftarrow$ and call r fact. If $head(r) = \emptyset$ we write $\bot \leftarrow l_1, \ldots, l_{n'}$. Rules of latter form are called integrity constraints; they eliminate all models satisfying their body. The following result is related to integrity constraints.

Proposition 10 *Let P be a program over $\mathcal{L}_\mathcal{X}$ containing a rule of form $a \leftarrow \neg a$ and for each HT_{LB}-stable model v of $P \setminus \{a \leftarrow \neg a\}$ over $\mathcal{X}, \mathbb{Z}$ we have that $\langle v, v\rangle \not\models a$.*

Then, P has no HT_{LB}-stable model.

[2] As usual, $w_1 x_1, + \cdots + w_n x_n < k$ and $w_1 x_1, + \cdots + w_n x_n > k$ can be expressed by $w_1 x_1, + \cdots + w_n x_n \leq k - 1$ and $w_1 x_1, + \cdots + w_n x_n \geq k + 1$, respectively.

This proposition seems to be trivial, but we show in Section 5 that Aziz' original approach does not satisfy this property.

In basic ASP, normal programs are of special interest, since their stable models are subset minimal.[3] In the following, we define and study normal programs in terms of HT_{LB}. Similar to ASP, we force the conclusion of normal rules to be not ambiguous, thus forbidding for instance disjunctive heads. We restrict heads to include exactly one atom and additionally exactly one variable as well. For instance, let P be a program consisting of fact $x + y \geq 42$ over $\{x, y\}, \mathbb{Z}$ only. Then, P has infinitely many stable models $\{v \mid v(x) + v(y) = 42\}$, eg $(x{\downarrow}0) \cup (y{\downarrow}42)$ and $(x{\downarrow}42) \cup (y{\downarrow}0)$. Hence, P should not be a normal program.

To illustrate that it is not enough to restrict heads for defining normal programs, let us reconsider program P with rules (2) of the introduction. Then, P has stable models $(x{\downarrow}1)$ and $(x \downarrow 42)$. Let us take a closer look on how to get them. First, we note that $v_1 = (x \downarrow 1)$ and $v_2 = (x \downarrow 42)$ are candidates of stable models, since both satisfy P. It is easy to see that there is no $v' \subset v_1$ with $v' \in [\![x \geq 1]\!]$ and hence v_1 is a stable model of P. Furthermore, consider valuation $v'' \subset v_2$. Then, $\langle v'', v_2 \rangle \models x \geq 42 \leftarrow \neg(x \leq 1)$ iff either both $v'' \in [\![x \geq 42]\!]$ and $v_2 \in [\![x \geq 42]\!]$ or $v_2 \in [\![x \leq 1]\!]$ holds. This boils down to $v'' \in [\![x \geq 42]\!]$, which implies that $v'' \subset v_2$ is contradicted. That is, v_2 is a stable model as well and $(x{\downarrow}1) \subset (x{\downarrow}42)$ holds. Hence, the stable models of P are not subset minimal, P should not be a normal program.

The issue shown in the previous example arises, due to the monotonicity of atoms. We define an atom a as *monotonic* (resp. *anti-monotonic*) wrt variable x if $v \in [\![a]\!]$ implies $v' \in [\![a]\!]$ for every valuation v' with $v \subseteq v'$ (resp. $v' \subseteq v$ with $v'(x) \neq \boldsymbol{u}$), where $v(y) = v'(y)$ for all $y \in vars(a) \setminus \{x\}$.[4] We define an atom a as *monotonic* (resp. *anti-monotonic*) if it is monotonic (resp. anti-monotonic) wrt all variables in $vars(a)$, and non-monotonic otherwise. Analogously, a program P is *monotonic* (resp. *anti-monotonic*) if all atoms occurring in it are monotonic (resp. anti-monotonic). We call a program P *directed* if no atom in it is non-monotonic. For instance, atom $x \geq 42$ is monotonic, $y < 42$ is anti-monotonic, and $x - y \geq 42$ is non-monotonic, since x is monotonic and y is anti-monotonic, respectively.

Thus, we define normal programs as follows.

Definition 8 *A rule over $\mathcal{L}_\mathcal{X}$ is normal if it is of form*

$$a_0 \leftarrow a_1, \ldots, a_n, \neg a_{n+1}, \ldots, \neg a_{n'} \tag{4}$$

where $|vars(a_0)| = 1$ and each atom a_i is monotonic for $n + 1 \leq i \leq n'$.

A normal program is a set of rules of form (4). As the program in (2) illustrates, programs containing rule bodies with not monotonic atoms in the scope of negation, like $\neg(x \leq 1)$, may lead to stable models which are not subset minimal. As in ASP, we have that stable models of normal programs are subset minimal.

[3]The fact that stable models are subset minimal is also known as anti-chain property.

[4]Note that our definition of monotonicity of atoms differs from Aziz' ones (Aziz 2015), due to different concepts of valuations.

Proposition 11 *Let P be a normal program over $\mathcal{L}_\mathcal{X}$. Then, each HT_{LB}-stable model of P over $\mathcal{X}, \mathbb{Z}$ is subset minimal.*

To elaborate more on the influence of atomic monotonicity on programs, let us consider the following example. Let P be a directed program, in which no atom occurs in the scope of negation:

$$x \geq 0 \qquad\qquad x \geq 42 \leftarrow y < 42$$
$$y \geq 0 \qquad\qquad y \geq 42 \leftarrow x < 42$$

Then, P has the two stable models $(x \downarrow 42) \cup (y \downarrow 0)$ and $(x \downarrow 0) \cup (y \downarrow 42)$. Compare this with the ASP program $\{a \leftarrow \neg b.\ b \leftarrow \neg a.\}$ formulating an "even loop" yielding stable models $\{a\}$ and $\{b\}$. Both programs behave similarly, since assigning x (or y) to 42 disables the foundedness of 42 for y (or x) in the same way as assigning a (or b) to true disables the foundedness of true for b (or a). That is, not monotonic atoms implicitly involve negation.

The previous example motivates us to define positive programs. To this end, we first define the positive and negative body of a rule. Let r be a normal rule of form (4), then we define the positive body of r as $body^+(r) \stackrel{\text{def}}{=} \{a_i \mid 1 \leq i \leq n, a_i \text{ monotonic}\}$ and its negative body as $body^-(r) \stackrel{\text{def}}{=} body(r) \setminus body^+(r)$, respectively. That is, atoms like $x < 42$ not occurring in the scope of negation belong to the negative body, since they are not monotonic.

Then, we define positive programs as follows.

Definition 9 *A normal rule r over $\mathcal{L}_\mathcal{X}$ is positive if $head(r)$ is monotonic and $body^-(r) = \emptyset$.*

A positive program is a set of positive rules.

The following result shows that a positive program has a unique stable model, just as in ASP (Apt, Blair, and Walker 1987).

Proposition 12 *Let P be a positive program over $\mathcal{L}_\mathcal{X}$. Then, P has exactly one HT_{LB}-stable model over $\mathcal{X}, \mathbb{Z}$.*

The proof follows the well-known idea of applying a fix point calculation using a continuous and monotonic operator.

In ASP, a program is stratified if it is free of recursion through negation (Apt, Blair, and Walker 1987), also referred to "negative loops". This idea remains the same in case of HT_{LB}. Note that we drop in this work the preliminaries needed for the following results, due to space limitations. That is, we give the definitions of dependency graph, loop, stratification and splitting set in terms of HT_{LB} in an extended work of this version.

The next results generalize the calculation of a stable model to stratified programs.

Proposition 13 *Let P be a stratified program over $\mathcal{L}_\mathcal{X}$ with monotonic heads only. Then, P has exactly one HT_{LB}-stable model over $\mathcal{X}, \mathbb{Z}$.*

Interestingly, allowing not monotonic atoms in the head may eliminate stable models but it does not produce further stable models. That is, if we drop the additional condition on heads, then we can still apply a fix point calculation and get the following result.

Proposition 14 *Let P be a stratified program over $\mathcal{L}_\mathcal{X}$. Then, P has at most one HT_{LB}-stable model over $\mathcal{X}, \mathbb{Z}$.*

For instance, the program consisting of facts $x \geq 42$ and $x < 42$ only has no HT_{LB}-stable model.

4.2 Modelling Capabilities

In this section, we go into an example of error diagnosis to illustrate some modelling features of HT_{LB} in terms of programs. In particular, the following example illustrates foundedness and default valuations.

Let $N = \{1, \ldots, n\} \subseteq \mathbb{Z}$ be an index set. We represent events by constants e_i and identify them with value i for i in N. Consider program P_{err} given by

$$error \geq \sum_{i \in X} e_i \leftarrow \bigwedge_{i \in X} occur(e_i) = 1 \quad \text{for all } X \subseteq N \quad (5)$$

$$occur(e_2) = 1 \leftarrow occur(e_3) = 1,\ error \geq 4 \quad (6)$$

$$occur(e_4) = 1 \leftarrow temperature \leq 42 \quad (7)$$

$$temperature = 60 \leftarrow \neg temperature \neq 60 \quad (8)$$

Rules of (5) express that the value of $error$ is greater or equal than the sum of occurred events.[5] The empty sum means that we have no error and is defined by 0. Rule (6) models the dependency of event e_2 regarding e_3 and the comparison if the value of $error$ is beyond some threshold value 4. If the value of $temperature$ falls below 42 degrees, then event e_4 occurs, modelled by Rule (7). Rule (8) sets the default value of $temperature$ to 60 degrees.

To illustrate the behaviour of P_{err} let us consider the specific instance I_{err} containing fact $occur(e_3) = 1$. Then, we get the single stable model $(temperature \downarrow 60) \cup (error \downarrow 3) \cup (occur(e_3) \downarrow 1)$ of $P_{err} \cup I_{err}$. The minimal founded value of $temperature$ is the default value 60. Since e_3 is the only event that occurs, by (5) we derive $error \geq e_3$ and thus the minimal founded value for $error$ is 3.

Let us extend I_{err} to I'_{err} by adding $temperature \geq 42$. Then we get stable models $(temperature \downarrow 60) \cup (error \downarrow 3) \cup (occur(e_3) \downarrow 1))$ and $(temperature \downarrow 42) \cup (error \downarrow 9) \cup (occur(e_2) \downarrow 1) \cup (occur(e_3) \downarrow 1) \cup (occur(e_4) \downarrow 1)$ of $P_{err} \cup I'_{err}$. Note that for one stable model the default valuation of $temperature$ is founded and for the other one not, due to non-monotonic atom $temperature \neq 60$ in the scope of negation. Hence, we derive $error \geq e_3$ and $error \geq e_2 + e_3 + e_4$, respectively.

5 Related Work

In this section, we compare HT_{LB} to existing formalisms.

5.1 $BFASP$

First, let us compare HT_{LB} to Aziz' bound founded ASP ($BFASP$; Aziz 2015), since both share the same motivation to generalize the idea of foundedness to ordered domains.

Let us point out some differences of both approaches. In $BFASP$ an arbitrary formula is called constraint and a rule is defined as a pair of a constraint and a variable called head. The constraint needs to be increasing wrt its head variable. A constraint is increasing in one of its variables if the constraint holds for a substitution of its variables by domain values and it holds for each substitution where the value of the particular variable is increased and rest stays the same as before. [6] Note that the definition of increasing is made for constraints and does not differentiate between the monotonicity of atoms and logic connectives. In case of atoms Aziz' definitions of increasing and ours of monotonic coincide. Stable models are defined in $BFASP$ via a reduct depending on the monotonicity of constraints wrt their variables and by applying a fix point operation.

Both, $BFASP$ and HT_{LB} assign variables to their smallest domain value per default. Interestingly, they differ in their understanding of smallest domain values. In HT_{LB}, the smallest domain value is always the value undefined to capture partiality, whereas in $BFASP$ partiality is not considered if the value undefined is not explicitly part of a given domain.

However, the value of the head variable is derived by the constraint even if it contains no implication. For instance, let $\mathbb{Z}_0^+$ be the variable domain of positive integers with 0 and $(x + y \geq 42, x)$ a rule in $BFASP$. Then, $BFASP$ yields one stable model assigning x to 42 and y to 0. The value of x is derived from the value of $42 - y$, obtained by the smallest value of y. Per default the value of y is 0, since y appears never as an head. This is different from HT_{LB} where the fact $x + y \geq 42$ results in two stable models $(x \downarrow 0) \cup (y \downarrow 42)$ and $(x \downarrow 42) \cup (y \downarrow 0)$. In HT_{LB}, the variables of a fact are treated in an equal way instead of an implicatory way by declaring one of them as head.

Now, we show that $BFASP$ does not satisfy the same well-known properties as HT_{LB}. In particular, $BFASP$ does not satisfying Proposition 10 in its turn. That is, in $BFASP$ we may get unintuitive stable models. For instance, consider ASP rule $p \leftarrow \neg p$. This rule has no stable model in ASP and HT_{LB}, since if p holds then we cannot derive p any more and if p not holds then we need to derive p. In contrast, $BFASP$ yields the stable model assigning p to true, since the reduct will never replace head variables and produce the rule as it is. Hence, $BFASP$ yields the stable model assigning p to true, since it is the minimal (and only) model of the rule.

5.2 HT_C

Next, we compare our approach to the logic of Here-and-There with constraints (HT_C; Cabalar et al. 2016).

First, note that both are based on HT and capture theories over (constraint) atoms in a non-monotonic setting and can express default values. The difference is that HT_{LB} inherently minimizes valuations wrt foundedness. This is achieved by additionally comparing valuations wrt the values assigned to the variables. Hence, we represent valuations by sets of downward closed tuples regarding the assignments to yield a comparison of values in a set based mean using standard subset relation. For instance, consider the fact $x \geq 42$ over $\{x\}, \mathbb{Z}$ and valuations v and v' with $v(x) = 42$ and $v'(x) = 43$. Then, in HT_C we have $v \neq v'$, whereas in HT_{LB} we have $v \subseteq v'$. Hence, v and v' are stable models in HT_C but only the first one is HT_{LB}-stable model wrt foundedness.

[5] Note that (5) leads to exponentially many rules; it is also possible to write this in a more compact way using nested expressions, what we not do in this work for reasons of simplicity.

[6] For more details see (Aziz 2015).

On a first look, HT_{LB} seems like HT_C with value minimization on top. However, this is insufficient, since it does not yield foundedness. Recall program P in (2) with HT_{LB}-stable models $(x\downarrow 1)$ and $(x\downarrow 42)$. In contrast, the minimal stable model in HT_C assigns x to 1. This eliminates the second HT_{LB}-stable model. Moreover, program P in (1) has the sole HT_{LB}-stable model $(x\downarrow 42) \cup (y\downarrow 0)$. Whereas in HT_C, we get two stable models with minimal values: one assigns x to 42 and y to 0, and the other x to 0 and y to 42.

However, both HT_{LB} and HT_C define atomic satisfaction in terms of atom denotations. A difference is that in HT_C denotations need to be closed.[7] Informally, a denotation is closed if for each valuation of the denotation every valuation which is a superset is in the denotation as well. For HT_{LB} this cannot be maintained, due to the additional comparison of valuations regarding values. For instance, consider atom $x \neq 42$ with $[\![x \neq 42]\!] = \{v \mid \boldsymbol{u} \neq v(x) \neq 42\}$ over $\{x\}, \mathbb{Z}$. Then, valuations v and v' with $v(x) = 0$ and $v'(x) = 99$ are part of the denotation, but v'' with $v''(x) = 42$ and $v \subseteq v'' \subseteq v'$ is not. The reason to be closed or not is that v, v' and v'' are different in HT_C but subsets in HT_{LB}, respectively.

The closure of denotions is significant to satisfy persistence in HT_C. In contrast, in HT_{LB} persistence is maintained by forcing atomic satisfaction in both h and t, instead of h only as in HT_C. The corresponding benefit is that this allows us to consider atoms in HT_{LB} which are not allowed in HT_C, like $x \doteq y$ with $[\![x \doteq y]\!] \stackrel{\text{def}}{=} \{v \mid v(x) = v(y)\}$ which is not closed in HT_C as well.

With HT_C and HT_{LB} we have two different paradigms, where one is maybe better suited than the other for a particular application area. We plan to further elaborate on possible application areas and the relation of HT_C and HT_{LB}.

5.3 Other Formalisms

ILP Let us compare Integer Linear Programming (*ILP*; Schrijver 1999) with HT_{LB}.

Note that *ILP* is a monotone theory. Hence, compared to *ASP* it is not intuitive to model recursion like reachability using *ILP*. For instance, in (Liu, Janhunen, and Niemelä 2012) it is mentioned that it is not easy to represent loop formulas in *ILP* which are needed for this purpose.

To overcome this shortcoming, approaches like HT_{LB} and HT_C tried to integrate monotone theories as *ILP* in a non-monotonic setting. In other words, these approaches can be seen as non-monotonic counterparts of *ILP* which support an intuitive modelling of reachability and thus recursion, like in *ASP*. That is, the benefit of an intuitive modelling is a key difference of HT_{LB} to *ILP*.

ASP **modulo Theories** Let us compare HT_{LB} to *ASP* modulo Theories approaches like in (Janhunen et al. 2017).

The idea of those approaches is to integrate monotone theories as linear programming in the non-monotonic setting of *ASP*. Informally, the theories are wrapped by ASP.

These approaches extend stable model semantics (Gelfond and Lifschitz 1991) by following the approach of lazy theory solving (Barrett et al. 2009). The idea is that a stable model is a set of atoms which needs to be valid regarding the underlying theory. Technically, in (Janhunen et al. 2017) a program over a theory is extended by rules depending on possible assignments wrt the theory to determine the stable models. The assignments for variables are obtained by particular theory solvers if the atoms are valid in the theory. It is interesting to note that there are two ways of interpreting atoms which do not occur in a model: one way is to assume that the opposite needs to hold and the other way is to let it open.

Similar to HT_C, the main difference of *ASP* modulo Theory approaches to HT_{LB} is that atoms are founded but per definition foundedness regarding values is not achieved for its comprised variables, since stable models in *ASP* modulo Theory rely on any possible valid assignment for variables.

Aggregates Aggregates are extensions of *ASP* allowing us to perform set operations like counting and summing on elements of a respective set. Aggregates can be treated by translating them into *ASP* rules. For instance, sum aggregates can be translated by adapting well-known techniques translating pseudo-Boolean constraints into SAT, cf (Sinz 2005) and (Bomanson and Janhunen 2013).

The syntax of an aggregate is given by $f\{c_1 : \varphi_1, \ldots, c_m : \varphi_m\} \prec k$, where f is an aggregate symbol, c_i, k constants, φ_i propositional formulas also called conditions with $1 \leq i \leq m$, and $\prec \in \{\leq, <, >, \geq, =, \neq\}$ a binary relation.

The community comes up with different semantics for aggregates like in (Ferraris 2011; Gelfond and Zhang 2014; Son and Pontelli 2007). Informally, a constant belongs to the set if its condition holds. An aggregate holds if its relation holds for all constants that belong to its set.

Obviously, (sum) aggregates are related to (linear constraint) atoms of HT_{LB}. As we will show in an extended version of this work, aggregates under Ferraris' semantics (Ferraris 2011) can be represented by atoms in HT_{LB}. To this end, we restrict conditions of aggregates to propositional atoms. Note that this is not a very limiting restriction, since these atoms can be seen as auxiliaries for arbitrary formulas.

This is interesting, since it means that aggregates are no longer an extension of an existing approach, instead aggregates under Ferraris' semantics are now already integrated as atoms of an approach. Hence, the results shown in this work allow us to view aggregates in a new setting and give us a possibly better way to elaborate on their properties like monotonicity. Maybe the view on aggregates as atoms in context of HT_{LB} helps us to better understand the existing discussion of different aggregate semantics and their properties.

6 Conclusion

We presented the idea of foundedness for minimal values of variables over ordered domains in the setting of the logic of Here-and-There. We elaborated on important properties like persistence, negation and strong equivalence and showed that they hold in our approach. Furthermore, we pointed out that the base logic HT can be seen as a special case of HT_{LB}. To prove if our approach follows well-known patterns, we showed that HT_{LB}-stable models correspond to stable models according to a Ferraris'-like stable model semantics.

[7] Please see (Cabalar et al. 2016) for more details.

To elaborate on our approach in terms of logic programming and modelling, we isolated a fragment dealing with linear constraints over integers. In this context, we analyzed the influence of monotonicity of atoms on programs and concepts like normal, stratified and positive. Moreover, we illustrated the features of foundedness and defaults with the example of error diagnosis.

Finally, we compared our approach to related ones and showed that foundedness is a non-trivial key feature of HT_{LB}. We showed that HT_{LB} and $BFASP$ have the same starting motivation but differ in their treatments of undefined and monotonicity. Furthermore, we pointed out that HT_{LB} can be seen as non-monotonic counterpart of monotonic theories. We also mentioned that HT_{LB} offers a new view of aggregates under Ferraris' semantics as atoms with its corresponding monotonic properties. Thus, aggregates are integrated in HT_{LB} instead of being an extension of an existing approach.

In an extended version we plan to present a fix point operator, dependency graph, (odd and even) loops, stratification, splitting sets, and the relation to aggregates in detail.

References

Apt, K.; Blair, H.; and Walker, A. 1987. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Publishers. chapter 2, 89–148.

Aziz, R. 2015. *Answer Set Programming: Founded Bounds and Model Counting*. Ph.D. Dissertation, University of Melbourne.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Barrett, C.; Sebastiani, R.; Seshia, S.; and Tinelli, C. 2009. Satisfiability modulo theories. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. chapter 26, 825–885.

Baselice, S.; Bonatti, P.; and Gelfond, M. 2005. Towards an integration of answer set and constraint solving. In Gabbrielli, M., and Gupta, G., eds., *Proceedings of the Twenty-first International Conference on Logic Programming (ICLP'05)*, volume 3668 of *Lecture Notes in Computer Science*, 52–66. Springer-Verlag.

Bomanson, J., and Janhunen, T. 2013. Normalizing cardinality rules using merging and sorting constructions. In Cabalar, P., and Son, T., eds., *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*, volume 8148 of *Lecture Notes in Artificial Intelligence*, 187–199. Springer-Verlag.

Cabalar, P.; Kaminski, R.; Ostrowski, M.; and Schaub, T. 2016. An ASP semantics for default reasoning with constraints. In Kambhampati, R., ed., *Proceedings of the Twenty-fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*, 1015–1021. IJCAI/AAAI Press.

Ferraris, P. 2005. Answer sets for propositional theories. In Baral, C.; Greco, G.; Leone, N.; and Terracina, G., eds., *Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05)*, volume 3662 of *Lecture Notes in Artificial Intelligence*, 119–131. Springer-Verlag.

Ferraris, P. 2011. Logic programs with propositional connectives and aggregates. *ACM Transactions on Computational Logic* 12(4):25.

Gelfond, M., and Lifschitz, V. 1990. Logic programs with classical negation. In Warren, D., and Szeredi, P., eds., *Proceedings of the Seventh International Conference on Logic Programming (ICLP'90)*, 579–597. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Gelfond, M., and Zhang, Y. 2014. Vicious circle principle and logic programs with aggregates. *Theory and Practice of Logic Programming* 14(4-5):587–601.

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. In *Sitzungsberichte der Preussischen Akademie der Wissenschaften*. Deutsche Akademie der Wissenschaften zu Berlin. 42–56. Reprint in Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik, Akademie-Verlag, 1986.

Janhunen, T.; Kaminski, R.; Ostrowski, M.; Schaub, T.; Schellhorn, S.; and Wanko, P. 2017. Clingo goes linear constraints over reals and integers. *Theory and Practice of Logic Programming* 17(5-6):872–888.

Liu, G.; Janhunen, T.; and Niemelä, I. 2012. Answer set programming via mixed integer programming. In Brewka, G.; Eiter, T.; and McIlraith, S., eds., *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*, 32–42. AAAI Press.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence* 47(1-2):3–41.

Schrijver, A. 1999. *Theory of linear and integer programming*. Discrete mathematics and optimization. John Wiley & sons.

Sinz, C. 2005. Towards an optimal CNF encoding of Boolean cardinality constraints. In van Beek, P., ed., *Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming (CP'05)*, volume 3709 of *Lecture Notes in Computer Science*, 827–831. Springer-Verlag.

Son, T., and Pontelli, E. 2007. A constructive semantic characterization of aggregates in answer set programming. *Theory and Practice of Logic Programming* 7(3):355–375.

Appendix of Proofs

Proof of Proposition 1 To prove that $\langle H, T\rangle \models \varphi$ holds under Definition 1 iff it holds when replacing Condition *2* by *2'* for $\langle H, T\rangle$ HT-interpretation over $\mathcal{A}$ and φ a propositional formula over $\mathcal{A}$, it is enough to prove equivalence of base cases *2* and *2'*, since the rest follows directly by structural induction. Per definition of denotation we have for propositional atom $p \in \mathcal{A}$ that

$$H \in [\![p]\!]_{\mathcal{A}} \Leftrightarrow H \in \{I \mid p \in I\} \Leftrightarrow p \in H$$

□

Proof of Proposition 2 It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each formula over $\mathcal{A}_\mathcal{X}$. Let $\langle h, t\rangle$ an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$ and a atom of $\mathcal{A}_\mathcal{X}$.

First, we prove persistence, represented by *1* of the proposition. We have

$$\langle h, t\rangle \models a \Leftrightarrow h \in [\![a]\!] \wedge t \in [\![a]\!] \Rightarrow t \in [\![a]\!] \Leftrightarrow \langle t, t\rangle \models a$$

Subsequently, we prove negation, represented by *2* of the proposition. We have

$\langle h, t\rangle \models a \rightarrow \bot$
$\Leftrightarrow (\langle h, t\rangle \models \bot \vee \langle h, t\rangle \not\models a) \wedge (\langle t, t\rangle \models \bot \vee \langle t, t\rangle \not\models a)$
$\Leftrightarrow \langle h, t\rangle \not\models a \wedge \langle t, t\rangle \not\models a$
$\Leftrightarrow (h \notin [\![a]\!] \vee t \notin [\![a]\!]) \wedge (t \notin [\![a]\!])$
$\Leftrightarrow \langle t, t\rangle \not\models a$ □

Proof of Proposition 3 For any $a \in At(\langle h, t\rangle) = \{a \in \mathcal{A}_\mathcal{X} \mid h \in [\![a]\!] \text{ and } t \in [\![a]\!]\}$ we have $h \in [\![a]\!]$ and $t \in [\![a]\!]$, thus we conclude $a \in At(\langle t, t\rangle) = \{a \in \mathcal{A}_\mathcal{X} \mid t \in [\![a]\!]\}$. □

Proof of Proposition 4 Let Γ_1, Γ_2 and Γ be theories over $\mathcal{A}_\mathcal{X}$. First, we prove "$\Leftarrow$" of the proposition. For each HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ we have $\langle h, t\rangle \models \Gamma_1$ iff $\langle h, t\rangle \models \Gamma_2$. This implies that $\langle h, t\rangle \models \Gamma_1 \cup \Gamma$ iff $\langle h, t\rangle \models \Gamma_2 \cup \Gamma$ for any Γ. Hence, $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ have the same HT_{LB}-stable models for every Γ.

Secondly, we prove "$\Rightarrow$" by contradiction. Without loss of generality, assume that $\langle h, t\rangle$ is HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$ with $\langle h, t\rangle \models \Gamma_1$ and $\langle h, t\rangle \not\models \Gamma_2$. Then, we differ two cases.

Case 1: Let $\langle t, t\rangle \not\models \Gamma_2$. We have $\langle h, t\rangle \models \Gamma_1$ and thus by persistence (Proposition 2) $\langle t, t\rangle \models \Gamma_1$. Let $\Gamma = \{x \succeq c \mid (x, c) \in t\}$. Then, $\langle t, t\rangle \models \Gamma_1 \cup \Gamma$ is HT_{LB}-stable model. But $\langle t, t\rangle \not\models \Gamma_2 \cup \Gamma$ by assumption.

Case 2: Let $\langle t, t\rangle \models \Gamma_2$. Moreover, let $\Gamma = \Gamma' \cup \Gamma''$ with $\Gamma' = \{x \succeq c \mid (x, c) \in h\}$ and $\Gamma'' = \{x \succeq t(x) \rightarrow y \succeq t(y),\ x \succeq c \rightarrow x \succeq t(x) \mid (x, c), (x, t(x)), (y, t(y)) \in t \setminus h, x \neq y\}$. Then, $\langle t, t\rangle \models \Gamma_2 \cup \Gamma$ by assumption and $h \subseteq t$. Note there is no $v \subset t$ with $\langle v, t\rangle \models \Gamma_2 \cup \Gamma$, since by $\langle h, t\rangle \not\models \Gamma_2$ we get that $h \subset v \subset t$ need to hold, and thus there exists at least one pair $a_1, a_2 \in atoms(\Gamma'')$ with $v \in [\![a_1]\!]$ and $v \notin [\![a_2]\!]$. Hence, $\langle v, t\rangle \not\models \Gamma_2 \cup \Gamma$ for $h \subseteq v \subset t$. Thus, $\langle t, t\rangle$ is HT_{LB}-stable model of $\Gamma_2 \cup \Gamma$. By assumption and construction, we have that $\langle h, t\rangle \models \Gamma_1$ and $\langle h, t\rangle \models \Gamma'$, respectively. Moreover, we have that $\langle h, t\rangle \not\models a$ for every $a \in atoms(\Gamma'')$. Hence, $\langle h, t\rangle \models \Gamma''$ and thus $\langle h, t\rangle \models \Gamma_1 \cup \Gamma$. Note that since $\langle h, t\rangle \not\models \Gamma_2$ and $\langle t, t\rangle \models \Gamma_2$ we have $\langle h, t\rangle \neq \langle t, t\rangle$, which implies that $h \subset t$. Finally, $\langle t, t\rangle$ is no HT_{LB}-stable model of $\Gamma_1 \cup \Gamma$. □

Proof of Proposition 5 Let a be an atom over $\mathcal{A}_\mathcal{X}$, and $\overline{a}$ and $\overline{a}^s$ its complement and its strict complement over $\mathcal{A}_\mathcal{X}$, respectively.

First, we prove $\overline{a}^s \models \overline{a}$. For any HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ we have

$\langle h, t\rangle \models \overline{a}^s$
$\Leftrightarrow h \in [\![\overline{a}^s]\!] \wedge t \in [\![\overline{a}^s]\!]$ with $[\![\overline{a}^s]\!] = 2^\mathfrak{V} \setminus ([\![a]\!] \cup \{v \mid v(x) = \boldsymbol{u} \text{ for some } x \in vars(a)\})$
$\Rightarrow h \in 2^\mathfrak{V} \setminus [\![a]\!] \wedge t \in 2^\mathfrak{V} \setminus [\![a]\!]$
$\Leftrightarrow \langle h, t\rangle \models \overline{a}$

Secondly, we prove $\overline{a} \models \neg a$. For any HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ we have

$\langle h, t\rangle \models \overline{a}$
$\Leftrightarrow h \in [\![\overline{a}]\!] \wedge t \in [\![\overline{a}]\!]$ with $[\![\overline{a}]\!] = 2^\mathfrak{V} \setminus [\![a]\!]$
$\Leftrightarrow h \notin [\![a]\!] \wedge t \notin [\![a]\!]$
$\Rightarrow t \notin [\![a]\!]$
$\overset{\text{Proposition 2}}{\Leftrightarrow} \langle h, t\rangle \models \neg a$ □

Proof of Proposition 6 It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each theory over $\mathcal{A}$.

Let Γ be a theory over propositional atoms $\mathcal{A}$ and $\langle H, T\rangle$ an HT-interpretation over $\mathcal{A}$. Let $\tau(\Gamma)$ be a theory over atoms $\{p = \mathbf{t} \mid p \in \mathcal{A}\}$ and $\langle \tau(H), \tau(T)\rangle$ an HT_{LB}-valuation over $\mathcal{A}, \{\mathbf{t}\}$. Then we have

$\langle H, T\rangle \models p$
$\Leftrightarrow H \in [\![p]\!]_\mathcal{A}$
$\overset{H \subseteq T}{\Leftrightarrow} H \in [\![p]\!]_\mathcal{A} \wedge T \in [\![p]\!]_\mathcal{A}$
$\Leftrightarrow \tau(H) \in [\![p = \mathbf{t}]\!]_{\mathcal{A},\{\mathbf{t}\}} \wedge \tau(T) \in [\![p = \mathbf{t}]\!]_{\mathcal{A},\{\mathbf{t}\}}$
$\Leftrightarrow \langle \tau(H), \tau(T)\rangle \models p = \mathbf{t}$ □

Proof of Proposition 7 It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each theory over $\mathcal{A}_\mathcal{X}$.

First, note that the pair $\langle H, T\rangle$ over $\mathcal{A}_\mathcal{X}$ with $H = At(\langle h, t\rangle)$ and $T = At(\langle t, t\rangle)$ is a well formed HT-interpretation, since $H \subseteq T$ holds by $h \subseteq t$ and Proposition 3. Then we have

$\langle h, t\rangle \models a$
$\Leftrightarrow h \in [\![a]\!]_{\mathcal{X},\mathcal{D}} \wedge t \in [\![a]\!]_{\mathcal{X},\mathcal{D}}$
$\Rightarrow H \in [\![a]\!]_{\mathcal{A}_\mathcal{X}} \wedge T \in [\![a]\!]_{\mathcal{A}_\mathcal{X}}$
$\Rightarrow \langle H, T\rangle \models a$ □

Proof of Proposition 8 Let φ over $\mathcal{A}$ be an arbitrary tautology in HT. This means that for every HT-interpretation $\langle H, T\rangle$ over $\mathcal{A}$ holds $\langle H, T\rangle \models \varphi$. Thus, we conclude that formula φ' over $\mathcal{A}_\mathcal{X}$ obtained by replacing $atoms(\varphi)$ in φ by atoms of $\mathcal{A}_\mathcal{X}$, is a tautology as well (for every HT_{LB}-valuation $\langle h, t\rangle$ over $\mathcal{X}, \mathcal{D}$ holds $\langle h, t\rangle \models \varphi'$), since the semantics of the atoms may change the truth value of a single atom but can not affect the truth of the formula itself. □

Proof of Proposition 9 It is enough to prove the proposition for the base case, since the rest follows directly by structural induction for each theory over $\mathcal{A}_\mathcal{X}$.

Let Γ be a theory over $\mathcal{A}_\mathcal{X}$ and $\langle h, t\rangle$ an HT_{LB}-valuation over $\mathcal{X}, \mathcal{D}$. Then, we have

$h \models_{cl} a^t$
$\Leftrightarrow h \models_{cl} a \wedge t \models_{cl} a$
$\Leftrightarrow h \in [\![a]\!] \wedge t \in [\![a]\!]$
$\Leftrightarrow \langle h, t\rangle \models a$ □

Consistency in Justification Theory*

Simon Marynissen Niko Passchyn Bart Bogaerts Marc Denecker
KU Leuven, Departement of Computer Science, Leuven, Belgium

Abstract

Justification frameworks constitute a unifying theory to describe semantics of non-monotonic logics. They have applications, among others, in logic programming, abstract argumentation, and nested definitions. This framework is built on the notion of a justification. Intuitively, this is a graph that explains the truth values of certain facts. However this introduces a potential problem: the justification status of an atom and its negation can be conflicting. So, in a well-defined semantics we want that these statuses are opposite. This is captured in two axioms, which correspond to our main theorem: when a fact, positive or negative has a good justification, its negation should not have a good justification, and if a fact has no good justification, then its negation should have a good justification. We prove that this holds for completion, Kripke-Kleene, stable and well-founded semantics. The semantics of a justification framework is determined by what is called a branch evaluation. We introduce dual branch evaluations, which give rise to dual semantics, such as, co-well-founded and co-stable semantics. This duality is similar to the duality between induction and co-induction. Moreover, justifications in dual semantics are linked to justifications in the original one. Furthermore, we define a notion of splittable branch evaluation and show that under such an evaluation, justifications can be "glued" together, essentially resulting in a single justification for all true facts.

1 Introduction

In the field of Knowledge Representation (KR), novel logics and new semantics are constantly proposed. In order to keep an overview of the various logics, the semantics developed for a single logic and their relationships, unifying frameworks are developed (Denecker, Marek, and Truszczyński 2000; Denecker, Brewka, and Strass 2015). Such frameworks provide a uniform way to determine a broad class of semantics for all covered logics. Sometimes, this shines new light on the formal relationship between logics (Denecker, Marek, and Truszczyński 2011); sometimes, this gives rise to new semantics, such as for parametric versions of logic programs(Denecker, Brewka, and Strass 2015).

One such framework for non-monotonic logics is the justification theory from (Denecker, Brewka, and Strass 2015), which is built on the notion of a *justification*. Intuitively, this is a graph that explains the truth values of certain facts in a structure. Not any justification is considered "good". A good justification can be seen as an explanation for a fact. In order to determine the quality of a justification, justification theory makes use of *branch evaluations*. Different branch evaluations give rise to different semantics. Furthermore, so-called *nesting* of justification frames allows for a modular semantics and greatly reduces the efforts needed to introduce new language constructs (e.g., aggregates) into logics covered by justification theory. KR languages covered by justification theory include logic programs and answer set programs, abstract argumentation, inductive definitions, and nested definitions.

In addition, justifications are also used for implementation purposes. They are used to compute unfounded sets in modern ASP solvers (Gebser, Kaufmann, and Schaub 2012; De Cat et al. 2013), can be used to check for relevance of atoms in complete search algorithms (Jansen et al. 2016), and recent lazy grounding algorithms are built on top of them (De Cat et al. 2015; Bogaerts and Weinzierl 2018).

In all frameworks we know off, only truthness of atoms is explained. However, falsity of atoms, or equivalently truthness of negative literals, is not explained. This creates an asymmetry between atoms and negative literals. This asymmetry is resolved in justification theory by allowing justifications to also explain negative literals. However, this introduces a potential problem; the justification status of an atom and its negation can be conflicting. So, in a well-defined semantics we want that these statuses are opposite. This is captured in two axioms, which correspond to our main theorem: when a fact, positive or negative has a good justification, its negation should not have a good justification, and if a fact has no good justification, then its negation should have a good justification. This is obviously a fundamental problem, without which the framework fails. We investigate this matter in detail and positively answer this for completion, Kripke-Kleene, stable and well-founded semantics under the condition that the rules for a fact and its negation are related appropriately.

Besides this consistency of having a good justification, we introduce new concepts and prove results about these concepts. The first concept is splitting an interpretation into two sets of literals, which informally act as a lower and upper

*This paper is also submitted to the 13th Workshop on Logical and Semantic Frameworks with Applications (LSFA 2018).

bound on the interpretation. The two sets are each others dual in the sense that is defined in this text. This allows to reason on a level of sets instead of interpretations, which is easier from a mathematical point of view. Intuitively, however, one prefers to think in terms of interpretations. Most results are provided with an intuitive explanation in terms of interpretations.

Apart from duality on these sets, we also introduce a duality on branch evaluations, which amounts to a dual semantics in justification theory. One example is the duality between co-induction and induction, where the latter is tied with well-founded semantics. In addition, we also construct a co-stable semantics, which has the same relation with stable semantics as in the co-well-founded and well-founded case.

Moreover, we investigate when it is possible to "glue" good justifications together. To study this property, splittable and transitive branch evaluations are introduced, where the latter is a subclass of the former. Intuitively, these evaluations only depend on a start or tail of a branch. For splittable branch evaluations, we prove that "glueing" good justifications gives good justifications.

Denecker and Deschreye already established a result similar to our main theorem in (Denecker and De Schreye 1993) for tree-like justifications in a three-valued setting. Denecker et. al also use justifications for describing causal processes in an upcoming paper (Denecker, Bogaerts, and Vennekens 2018), and as such, our results are thus applicable for causal processes as well. The pasting of justifications introduces an operator between justifications. Similarly, in (Cabalar, Fandinno, and Fink 2014), an algebra of causal justifications is introduced.

The paper is structured as follows. In the preliminaries, we recall most of the definitions of justification theory, with small changes to ensure greater consistency and generality. In Section 3, we introduce complementary justification frames, which are based on complement closure as defined in (Denecker, Brewka, and Strass 2015). In Section 4, we first describe the lower and upper bounds on interpretations. Next, duality on these sets and dual branch evaluations are defined. Section 5 introduces splittable and transitive branch evaluations. In Section 6, we prove that justifications can be "glued" for splittable branch evaluations. We end in Section 7 by stating our main theorem. Due to page restrictions, proofs are omitted. A full version of this text, including most proofs, can be found at `https://bitbucket.org/simonmarynissen/consistencyinjustificationtheorynmr2018`. The results from Section 4 onwards were partially described in the master's thesis of the second author, see (Passchyn 2017), except for dual semantics and the proof of the main theorem in case of the well-founded semantics.

2 Preliminaries

In this section, we recall the basics of justification theory. Our presentation is based on the work of Denecker et al. (Denecker, Brewka, and Strass 2015). Small modifications to some existing definitions are made, to ensure greater consistency and generality; these changes have no major consequences and are clearly indicated.

In the rest of this paper, let $\mathcal{F}$ be a set, referred to as a *fact space*, such that $\mathcal{L} \coloneqq \{\mathbf{t}, \mathbf{f}, \mathbf{u}, \mathbf{i}\} \subseteq \mathcal{F}$[1], where $\mathbf{t}$, $\mathbf{f}$, $\mathbf{u}$ and $\mathbf{i}$ have the respective meaning *true*, *false*, *unknown* and *inconsistent*. The elements of $\mathcal{F}$ are called *facts*. The set $\mathcal{L}$ is equal to the four-valued logic of Belnap (Belnap 1977) with truth order $\mathbf{f} \leq_t \mathbf{u}, \mathbf{i} \leq_t \mathbf{t}$ and information order $\mathbf{u} \leq_k \mathbf{f}, \mathbf{t} \leq_k \mathbf{i}$. We assume that $\mathcal{F}$ is equipped with an involution $\sim : \mathcal{F} \to \mathcal{F}$ (i.e. a bijection that is its own inverse) such that $\sim\mathbf{t} = \mathbf{f}$, $\sim\mathbf{u} = \mathbf{u}$, $\sim\mathbf{i} = \mathbf{i}$ and $\sim x \neq x$ for all $x \in \mathcal{F} \setminus \{\mathbf{u}, \mathbf{i}\}$. The assumption that $\sim\mathbf{u} = \mathbf{u}$ and $\sim\mathbf{i} = \mathbf{i}$ differs from (Denecker, Brewka, and Strass 2015), where a positive and negative $\mathbf{u}$ and $\mathbf{i}$ are used. For any fact x, $\sim x$ is called the *complement* of x. An example of a fact space $\mathcal{F}$ is the set of literals over a propositional vocabulary Σ extended with $\mathcal{L}$ where $\sim$ maps a literal to its negation. For any set A we define $\sim A$ to be the set containing elements of the form $\sim a$ for $a \in A$.

We distinguish two types of facts: *defined* and *open* facts[2]. The former are accompanied by a set of rules that determine their truth value. The truth value of the latter is not governed by the rule system but comes from an external source or is fixed (as is the case for logical facts).

Definition 2.1. A *justification frame* $\mathcal{JF}$ is a tuple $\langle \mathcal{F}, \mathcal{F}_d, R \rangle$ such that

- $\mathcal{F}_d$ is a subset of $\mathcal{F}$ closed under $\sim$, i.e. $\sim\mathcal{F}_d = \mathcal{F}_d$; facts in $\mathcal{F}_d$ are called *defined*;
- no logical fact is defined: $\mathcal{L} \cap \mathcal{F}_d = \emptyset$;
- $R \subseteq \mathcal{F}_d \times 2^{\mathcal{F}}$;
- for each $x \in \mathcal{F}_d$, $(x, \emptyset) \notin R$ and there is an element $(x, A) \in R$.[3]

The set of *open* facts is denoted as $\mathcal{F}_o \coloneqq \mathcal{F} \setminus \mathcal{F}_d$. An element $(x, A) \in R$ is called a *rule* with *head* x and *body* (or *case*) A. The set of cases of x in $\mathcal{JF}$ is denoted as $\mathcal{JF}(x)$. Rules (x, A) are denoted as $x \leftarrow A$ and if $A = \{y_1, \ldots, y_n\}$, we often write $x \leftarrow y_1, \ldots, y_n$. Cases are allowed to be infinite; as such our framework covers also, for instance, non-ground logic programs where grounding can result in infinitely long bodies (as observed, e.g., by (Harrison, Lifschitz, and Yang 2013)).

A rule $x \leftarrow A$ expresses that if all facts of A are true, then x is true. Therefore, a case of x represents a sufficient condition for x to hold. Furthermore, the set of cases can be seen as a necessary condition for x to hold, that is, x only holds if there is a rule for x that can be applied. We illustrate the rest of notions and definitions with the the following running example.

[1] Denecker et al. (Denecker, Brewka, and Strass 2015) do not require that all logical facts are present in $\mathcal{F}$.

[2] In other literature, external, exogenous and parametric are also used to denote open facts. Defined facts are sometimes named internal or endogenous facts.

[3] Due to this property, justification frames here correspond to *proper* justification frames in (Denecker, Brewka, and Strass 2015). Any set of rules can be transformed into a proper justification frame; rules of the form $x \leftarrow \emptyset$ are replaced with $x \leftarrow \mathbf{t}$, and for $x \in \mathcal{F}_d$ that have no rules, we introduce $x \leftarrow \mathbf{f}$.

Example 2.2. In this example we build a justification frame to express the transitive closure of a graph. So let V be a set of nodes. Define $\mathcal{F}_o$ to be the set of elements $\mathrm{Edge}(v,w)$ and $\sim\mathrm{Edge}(v,w)$ with $v,w \in V$ and $\mathcal{F}_d$ to be the set of elements $\mathrm{Path}(v,w)$ and $\sim\mathrm{Path}(v,w)$ with $v,w \in V$. Define $\mathcal{F} = \mathcal{F}_d \cup \mathcal{F}_o$. The facts encoding the edges of a graph are in $\mathcal{F}_o$. This means that they can freely change and thus act as parameters, whereas the facts in $\mathcal{F}_d$ are constrained rules as seen below. In section 3 we construct the rules for $\sim\mathrm{Path}(v,w)$, but for now we only define the rules for $\mathrm{Path}(v,w)$:

- $\mathrm{Path}(v,w) \leftarrow \mathrm{Edge}(v,w)$;
- $\mathrm{Path}(v,w) \leftarrow \mathrm{Path}(v,x), \mathrm{Path}(x,w)$;

for all $v,w,x \in V$. This encodes that Path is the transitive closure of Edge. ▲

Definition 2.3. Two justification frames $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ and $\mathcal{JF}' = \langle \mathcal{F}, \mathcal{F}_d, R'\rangle$ are *equivalent* if for every rule $x \leftarrow A \in R$, there is a rule $x \leftarrow B \in R'$ such that $B \subseteq A$, and likewise for every rule $x \leftarrow B \in R'$, there is a rule $x \leftarrow A \in R$ such that $A \subseteq B$.

Equivalence is defined here in a different, but equivalent[4] way as in (Denecker, Brewka, and Strass 2015). A case $A \in \mathcal{JF}(x)$ so that there is a $B \in \mathcal{JF}(x)$ with $B \subseteq A$ is said to be *redundant*. Deleting a redundant rule results in an equivalent justification frame if the more general rule is kept. However, it is not always possible to remove all redundant rules without losing equivalence, see e.g. (Denecker, Brewka, and Strass 2015, Example 3).

Justifications and branch evaluations

Definition 2.4. A *justification* J in a justification frame $\langle \mathcal{F}, \mathcal{F}_d, R\rangle$ is a subset of R containing at most one rule for each $x \in \mathcal{F}_d$. If $x \leftarrow S \in J$, we denote $J(x) = S$.

This notation reveals that we can view a justification J as a partial function from $\mathcal{F}_d$ to $2^{\mathcal{F}}$ such that $x \leftarrow J(x) \in R$ if $x \in \mathrm{dom}(J)$, where $\mathrm{dom}(J)$ denotes the domain of J viewed as a total function. We use the two representations interchangeably.

A justification J gives a reason for facts x in its domain. This reason depends on a case of x. If this case $J(x)$ contains defined facts for which J does not give a reason, then our explanation is possibly incomplete. Therefore, we typically want that $J(x) \cap \mathcal{F}_d \subseteq \mathrm{dom}(J)$ for every $x \in \mathrm{dom}(J)$.

Definition 2.5. A justification is *locally complete*[5] if $J(x) \cap \mathcal{F}_d \subseteq \mathrm{dom}(J)$ for all $x \in \mathrm{dom}(J)$.

Given a justification J, we construct a directed graph G_J with vertices

$$\mathrm{dom}(J) \cup \{y \in \mathcal{F} \mid \exists x \in \mathrm{dom}(J) : y \in J(x)\}.$$

and edges

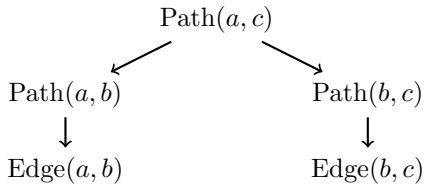
$$\{(x,y) \mid x \in \mathrm{dom}(J), y \in J(x)\}.$$

Since graphs are easily visualised, we obtain a natural and visual way to represent justifications.

Lemma 2.6. *A justification J is locally complete if and only if G_J has no defined leaves.*

Remark that the direction of the edges in G_J is opposite of the arrow direction in rule notation. We provide an example of how a justification could look like.

Example 2.7. Let $V = \{a,b,c\}$ in the setting of Example 2.2. Suppose that $\mathrm{Edge}(a,b)$ and $\mathrm{Edge}(b,c)$ hold. The following locally complete justification gives an explanation why $\mathrm{Path}(a,c)$ holds.



▲

Definition 2.8. Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ be a justification frame. A *$\mathcal{JF}$-branch* is either an infinite sequence in $\mathcal{F}_d$ or a finite non-empty sequence in $\mathcal{F}_d$ together with an element in $\mathcal{F}_o$. For a justification J in $\mathcal{JF}$, a *J-branch* starting in $x \in \mathcal{F}_d$ is a maximally long path (with at least one edge) in G_J starting in x.

For instance in Example 2.7, the path

$$\mathrm{Path}(a,c) \to \mathrm{Path}(a,b) \to \mathrm{Edge}(a,b)$$

is a finite branch of the given justification.

Remark 2.9. A finite $\mathcal{JF}$-branch can be represented as a finite sequence in $\mathcal{F}$ such that the last element is in $\mathcal{F}_o$ and all other elements in $\mathcal{F}_d$. As a consequence, a finite $\mathcal{JF}$-branch has at least two elements. Not all J-branches are $\mathcal{JF}$-branches since they can end in defined facts. However, if J is locally complete, any J-branch is also an $\mathcal{JF}$-branch by Lemma 2.6.

We denote a branch $\mathbf{b}$ as

$$\mathbf{b} : x_0 \to x_1 \to \cdots.$$

and define $\sim\mathbf{b}$ as $\sim x_0 \to \sim x_1 \to \cdots$. A J-branch can be seen as part of a possible explanation of a fact. Given a rule $p \leftarrow p$, the explanation that p is true because p is true is generally not deemed acceptable. Therefore, a qualitative measure of branches is required.

Definition 2.10. A *branch evaluation* $\mathcal{B}$ is a mapping that maps any $\mathcal{JF}$-branch to an element in $\mathcal{F}$ for all justification frames $\mathcal{JF}$. A justification frame $\mathcal{JF}$ together with a branch evaluation $\mathcal{B}$ form a *justification system* $\mathcal{JS}$, which is presented as a quadruple $\langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}\rangle$.

We start with some particular branch evaluations.

Example 2.11. The *supported* (or completion (Clark 1978)) branch evaluation $\mathcal{B}_{\mathrm{sp}}$ maps $x_0 \to x_1 \to \cdots$ to x_1. The *Kripke-Kleene* branch evaluation $\mathcal{B}_{\mathrm{KK}}$ maps finite branches to their last node and infinite branches to $\mathbf{u}$. ▲

[4] Equivalence only holds if the derivation operator in (Denecker, Brewka, and Strass 2015) is extended to all subsets of $\mathcal{F}$ in the obvious way.

[5] There is also the notion of a complete justification: $\mathcal{F}_d = \mathrm{dom}(J)$. However, such a justification has a rule for any defined fact and thus tries to explain every fact, even complementary facts.

The names of these branch evaluations are not arbitrary and the correspondence with the equally named semantics for logic programs is explained in (Denecker, Brewka, and Strass 2015). Later, we also define well-founded and stable branch evaluations. These examples show that a branch evaluation induces a semantics. This is manifested by fixed points of the extended support operator defined below, which is illustrated in (Denecker, Brewka, and Strass 2015).

Definition 2.12. Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a justification system, I a subset[6] of $\mathcal{F}$ and J a locally complete justification in $\mathcal{JS}$. We say $x \in \mathcal{F}_d$ is *supported* by J in I under $\mathcal{B}$ if $x \in \mathrm{dom}(J)$ and all J-branches starting in x are mapped into I under $\mathcal{B}$. The fact x is *supported* by $\mathcal{JS}$ in I if there exists a locally complete[7] justification J in $\mathcal{JS}$ that supports x in I under $\mathcal{B}$.

This definition is best clarified with an example.

Example 2.13. Let the setting be the same as in Example 2.7 and set $I = \{\mathrm{Edge}(a,b), \mathrm{Edge}(b,c)\}$. The fact $\mathrm{Path}(a,c)$ is supported in I under the Kripke-Kleene branch evaluation, a justification supporting $\mathrm{Path}(a,c)$ is already given in Example 2.7. ▲

Associated with a justification system, we get the *support operator* $\mathcal{S}_{\mathcal{JS}} : 2^{\mathcal{F}} \to 2^{\mathcal{F}_d}$ that maps $I \subseteq \mathcal{F}$ to

$$\mathcal{S}_{\mathcal{JS}}(I) \coloneqq \{x \in \mathcal{F}_d \mid x \text{ is supported by } \mathcal{JS} \text{ in } I\}.$$

If $\mathcal{JS}$ consists of a justification frame $\mathcal{JF}$ and a branch evaluation $\mathcal{B}$, this operator is also denoted as $\mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}$. The support operator can be extended to return subsets of $\mathcal{F}$ as follows:

$$\hat{\mathcal{S}}_{\mathcal{JS}} : 2^{\mathcal{F}} \to 2^{\mathcal{F}} : I \mapsto \mathcal{S}_{\mathcal{JS}}(I) \cup (I \cap \mathcal{F}_o).$$

In (Denecker, Brewka, and Strass 2015), various semantics are linked to fixed points of this *extended support operator*.

We also define what it means for a defined fact to be supported by a non locally complete justification.

Definition 2.14. A fact $x \in \mathcal{F}_d$ is *supported* by a (not necessarily locally complete) justification J in I under $\mathcal{B}$ if every locally complete extension of J supports x in I under $\mathcal{B}$.

This definition of support is compatible with the previous one since if J is a locally complete justification and K a locally complete extension of J, then any K-branch starting in $x \in \mathrm{dom}(J)$ is also a J-branch; hence K supports all facts that J supports. This alternative definition is useful in proofs, but also allows to give justifications that do not contain unnecessary information. For instance, under completion semantics, we do not need locally complete justifications, but just a single rule.

Support is invariant under equivalence of the underlying justification frame.

Proposition 2.15. *Let $\mathcal{JF}$ and $\mathcal{JF}'$ be two equivalent justification frames. Then $\mathcal{S}^{\mathcal{B}}_{\mathcal{JF}} = \mathcal{S}^{\mathcal{B}}_{\mathcal{JF}'}$ for any branch evaluation $\mathcal{B}$.*

[6] In (Denecker, Brewka, and Strass 2015), this is only defined for subsets containing $\mathbf{t}$ and $\mathbf{i}$ and not $\mathbf{f}$ and $\mathbf{u}$. The reason to allow more sets is explained later in subsection 4.

[7] In (Denecker, Brewka, and Strass 2015), a complete justification is required, that is, the domain is equal to $\mathcal{F}_d$. This, however, does not change the definition of support.

3 Complementary justification frames

In many applications, only one of x or $\sim x$ has explicit rules. For instance in logic programming (van Emden and Kowalski 1976; Marek and Truszczyński 1999), only rules for atoms are given, the rules for negative literals are left implicit. On the other hand, in Dungs argumentation frameworks (Dung 1995), only an attack relation between arguments is given, which can be seen as explicit rules for falsity of arguments, e.g., if a attacks b, then this is expressed by the rule $\sim b \leftarrow a$. Even in Example 2.2 we refrained from formulating when $\sim\mathrm{Path}(v,w)$ holds.

Under the assumption that x and $\sim x$ are complementary, rules for $\sim x$ can be constructed by using rules for x. As mentioned before, a case S of x can be seen as a sufficient condition for x to hold. Similarly, x only holds if a case of x holds, thus $\sim x$ holds if every case of x has a a fact that does not hold.

Definition 3.1. A *selection function* of $x \in \mathcal{F}_d$ in a justification frame $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \rangle$ is a function $\mathcal{S} : \mathcal{JF}(x) \to \mathcal{F}$ such that $\mathcal{S}(A) \in A$ for all $A \in \mathcal{JF}(x)$.

A selection function for x selects an element of each rule of x. In general, selection functions exists assuming the axiom of choice. Given a selection function $\mathcal{S}$ of x, the set $\sim\mathrm{Im}(\mathcal{S})$ is a potential case of $\sim x$, where $\mathrm{Im}(\mathcal{S})$ is the image of $\mathcal{S}$. This motivates the following.

Definition 3.2. Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \rangle$ be a justification frame. Define $R^* \subseteq \mathcal{F}_d \times 2^{\mathcal{F}}$ as the set of elements of the form

$$(\sim x, \sim \mathrm{Im}(\mathcal{S}))$$

for $x \in \mathcal{F}_d$ and $\mathcal{S}$ a selection function of x. The *complement* of $\mathcal{JF}$ is defined as $\mathrm{C}(\mathcal{JF}) \coloneqq \langle \mathcal{F}, \mathcal{F}_d, R^* \rangle$. The *complementation* of $\mathcal{JF}$ is defined as $\mathrm{CC}(\mathcal{JF}) \coloneqq \langle \mathcal{F}, \mathcal{F}_d, R \cup R^* \rangle$. A justification frame $\mathcal{JF}$ is *complementary* if it is equivalent with $\mathrm{C}(\mathcal{JF})$.

Remark 3.3. Denecker et al. (Denecker, Brewka, and Strass 2015) only defined complementation, which they called complement closure, which is a misleading name since complementation is not a closure, i.e. not idempotent. The essential properties of complementation are captured by complementary justification frames.

Complementation can also be applied to any set of rules. Most examples will start with a set or rules that does not form a justification frame.

Example 3.4. Take $\mathcal{F}_d = \{a, b, c, \sim a, \sim b, \sim c\}$, $\mathcal{F}_o = \mathcal{L}$ and $R = \{a \leftarrow b; a \leftarrow c; b \leftarrow \mathbf{t}; c \leftarrow \mathbf{t}\}$, then the complementation of R gives a complementary justification frame $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \cup R^* \rangle$, which adds the rules $\{\sim a \leftarrow \sim b, \sim c; \sim b \leftarrow \mathbf{f}; \sim c \leftarrow \mathbf{f}\}$. For instance the rule $\sim a \leftarrow \sim b, \sim c$ is in R since $a \leftarrow b$ and $a \leftarrow c$ are in R. ▲

Let us also check what complementation means in our running example.

Example 3.5. In Example 2.2, we did not state the rules for $\sim\mathrm{Path}(v,w)$. With complementation we get

$$\sim\mathrm{Path}(v,w) \leftarrow \{\sim\mathrm{Edge}(v,w)\} \cup S$$

with S a subset of $\mathcal{F}$ such that for every $x \in V$, at least $\sim\mathrm{Path}(v,x)$ or $\sim\mathrm{Path}(x,w)$ is in S. Intuitively it says that v and w are not connected if there is no edge between v and w and for all $x \in V$, there is no path from v to x or no path from x to w. ▲

We obtain the following characterisation for complementary frames.

Proposition 3.6. *Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ be a justification frame. Then $\mathcal{JF}$ is complementary if and only if for every $x \in \mathcal{F}_d$ the following hold*

1. *for every selection function $\mathcal{S}$ of x in $\mathcal{JF}$, there exists an $A \in \mathcal{JF}(\sim x)$ such that $A \subseteq \sim\mathrm{Im}(\mathcal{S})$;*
2. *for every $A \in \mathcal{JF}(x)$, there exists a selection function $\mathcal{S}$ of $\sim x$ in $\mathcal{JF}$ such that $\sim\mathrm{Im}(\mathcal{S}) \subseteq A$.*

Proof. Follows directly from writing out what equivalence means in this context. □

With complementation we can construct complementary justification frames.

Proposition 3.7. *Let $\mathcal{F}$ be a fact space and let $\mathcal{F}_d$ be a subset of $\mathcal{F}$ closed under $\sim$ that does not contain logical facts. Let $\{\mathcal{F}_1, \mathcal{F}_2\}$ be a partition of $\mathcal{F}_d$ such that $\sim\mathcal{F}_1 = \mathcal{F}_2$. Let R be a subset of $\mathcal{F}_1 \times (2^{\mathcal{F}} \setminus \emptyset)$ such that for each $x \in \mathcal{F}_1$, there is an $(x, A) \in R$. Then $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \cup R^*\rangle$ is complementary.*

In general, complementation of a justification frame is not always complementary as illustrated by the following example.

Example 3.8. Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ with $\mathcal{F}_d = \{a, \sim a\}$, $\mathcal{F}_o = \mathcal{L}$ and $R = \{a \leftarrow \mathbf{t}; \sim a \leftarrow \mathbf{t}\}$. Then the complementation of $\mathcal{JF}$ has the following rules $\{a \leftarrow \mathbf{t}; \sim a \leftarrow \mathbf{t}; a \leftarrow \mathbf{f}; \sim a \leftarrow \mathbf{f}\}$, while the complement of the complementation of $\mathcal{JF}$ has the rules $\{a \leftarrow \mathbf{t}, \mathbf{f}; \sim a \leftarrow \mathbf{t}, \mathbf{f}\}$. These two justification frames are not equivalent; hence the complementation of $\mathcal{JF}$ is not complementary. ▲

The following lemma illustrates that cases of a fact and its negation are intrinsically related in a complementary justification frame.

Lemma 3.9. *If a justification frame is complementary, then for every rule $x \leftarrow A$ and rule $\sim x \leftarrow B$, $A \cap \sim B \neq \emptyset$.*

Lemma 3.9 expands to justifications.

Lemma 3.10. *Let $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ be a complementary justification frame and $x \in \mathcal{F}_d$. If J and K are locally complete justifications with $x \in \mathrm{dom}(J)$ and $\sim x \in \mathrm{dom}(K)$, then there exists a J-branch $\mathbf{b}$ starting in x such that $\sim\mathbf{b}$ is a K-branch starting in $\sim x$.*

Even though, we have found branches that are each others negation, in general we cannot relate the evaluation of these branches. The following definition gives an obvious relation.

Definition 3.11. A branch evaluation $\mathcal{B}$ is *consistent* if for all justification frames $\mathcal{JF}$ and all $\mathcal{JF}$-branches $\mathbf{b}$ it holds that $\mathcal{B}(\sim\mathbf{b}) = \sim\mathcal{B}(\mathbf{b})$.

If a justification frame is complementary it means that rules for a fact and its negation are compatible. One should expect that in a such a frame, the justification status should also be compatible, that is, a fact is supported in I if and only if the negation is not supported in I. However, this is not always the case, as illustrated in the following example.

Example 3.12. Take $\mathcal{F}_d = \{a, \sim a, b, \sim b, c, \sim c\}$ and $\mathcal{F}_o = \mathcal{L}$, $\mathcal{F}_+ = \{a, b, c\}$, and $\mathcal{F}_- = \{\sim a, \sim b, \sim c\}$. The set of rules is the complementation of

$$\{a \leftarrow b; a \leftarrow c; b \leftarrow a; c \leftarrow a\}\,.$$

Define the branch evaluation $\mathcal{B}$ as follows for infinite branches:

- $\mathcal{B}(x_0 \to x_1 \to \cdots) = \mathbf{f}$ if $\exists i_0 \in \mathbb{N} : \forall i, j > i_0 : x_i \in \mathcal{F}_+$ and if $x_i = x_j$, then $x_{i+1} = x_{j+1}$;
- $\mathcal{B}(x_0 \to x_1 \to \cdots) = \mathbf{t}$ if $\exists i_0 \in \mathbb{N} : \forall i, j > i_0 : x_i \in \mathcal{F}_-$ and if $x_i = x_j$, then $x_{i+1} = x_{j+1}$;
- $\mathcal{B}(x_0 \to x_1 \to \cdots) = \mathbf{u}$ otherwise.

Finite branches are mapped to their last element. The only connected locally complete justifications with a or $\sim a$ in their domain are given below:

Therefore, it is easy to see that a is not supported by $\mathcal{JS}$ in any subset of $\mathcal{F}$ not containing $\mathbf{f}$, but also $\sim a$ is not supported by $\mathcal{JS}$ in any subset of $\mathcal{F}$ not containing both $\mathbf{t}$ and $\mathbf{u}$. This means that the semantics of the branch evaluation $\mathcal{B}$ is defective. The goal of the rest of the paper is to prove that this situation cannot occur for supported, Kripke-Kleene, stable and well-founded branch evaluations. ▲

4 Duality

Duality results are ubiquitous in mathematics and computer science. Duality results of points and hyperplanes in projective geometry are famous examples. In logic, the duality between conjunctive normal form (CNF) and disjunctive normal form (DNF) is another. This duality can be explained in terms of justification frames. Let ϕ be a CNF formula, i.e. it is a conjunction of clauses, of which each clause is a disjunction of literals. Let $c_1, \ldots, c_n$ be the different conjuncts of ϕ. For each conjunct c_i, let $d_{i,1}, \ldots, d_{i,m_i}$ be its disjuncts. We can construct the following set R of rules containing

$$\phi \leftarrow c_1, \ldots, c_n \quad \text{and} \quad c_i \leftarrow d_{i,j}$$

for all $1 \leq i \leq n$ and $1 \leq j \leq m_i$. The set R corresponds to the CNF formula ϕ. The dual of ϕ is defined as ϕ after swapping $\wedge$ with $\vee$ and changing literals to their negation. The complementation of the above set R forms a justification frame if taking $\mathcal{F}_d = \{\phi, \sim\phi, c_1, \sim c_1, \ldots, c_n, \sim c_n\}$ and $\mathcal{F}_o = \mathcal{L} \cup \{d_{i,j}, \sim d_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\}$. The rules for $\sim\phi$ and $\sim c_i$ correspond with the dual of the CNF formula ϕ. The duality result between CNF and DNF is as follows:

- ϕ is satisfiable if the dual of ϕ is not valid.

- ϕ is valid if the dual of ϕ is not satisfiable.

Let I be a subset of $\mathcal{F}$ such that exactly one of $d_{i,j}$ and $\sim d_{i,j}$ is in I for all i and j. Call I a $*$-set. It is not hard to see that ϕ is satisfiable if and only if ϕ is supported in some $*$-set I under $\mathcal{B}$ with $\mathcal{B}$ any branch evaluation that maps finite branches to their last element. For simplicity, suppose $\mathcal{B} = \mathcal{B}_{\mathrm{KK}}$. Similarly, the dual of ϕ is satisfiable if and only if $\sim\phi$ is supported in some $*$-set I under $\mathcal{B}$. This means that we have that

- ϕ is supported in some $*$-set I if and only if $\sim\phi$ is not supported in some $*$-set I.
- ϕ is supported in all $*$-sets I if and only if $\sim\phi$ is not supported in all $*$-sets I.

In what follows, we investigate how this result generalizes, i.e., we investigate the following question. If a fact x is supported in I, will $\sim x$ be supported in some I'? And conversely, if a fact x is not supported in I, will $\sim x$ be supported in some I'? We investigate this when I' is the dual of I, which we define below. The duality result of CNF and DNF formulas is also a consequence of our main theorem.

Interpretable sets

In this section we show how "subsets of $\mathcal{F}$" relate to interpretations as often defined in various branches of logic. In particular, given a propositional vocabulary Σ, a four-valued interpretation is often defined as a mapping $\Sigma \to \mathcal{L}$. Taking $\mathcal{F}$ the set of literals over Σ, extended with the logical facts, this induces a map $\mathcal{I} : \mathcal{F} \to \mathcal{L}$ such that $\mathcal{I}(\sim x) = \sim\mathcal{I}(x)$ for all $x \in \mathcal{F}$ and $\mathcal{I}(\ell) = \ell$ for all $\ell \in \mathcal{L}$. We call such a map an interpretation of $\mathcal{F}$.

As mentioned before, our set of logical facts corresponds to Belnap's four-valued logic (Belnap 1977). Each value can also be represented as a closed interval (in the truth order, represented as a tuple) of two-valued truth values: $\mathbf{t}$ corresponds to $(\mathbf{t},\mathbf{t})$, $\mathbf{f}$ to $(\mathbf{f},\mathbf{f})$, $\mathbf{u}$ to $(\mathbf{f},\mathbf{t})$, and $\mathbf{i}$ to $(\mathbf{t},\mathbf{f})$.

For each interpretation $\mathcal{I}$, we can associate two sets of facts: a *lower interpretable subset* I_ℓ and an *upper interpretable subset* I_u. The set I_ℓ consists of the true and inconsistent facts and I_u consists of the true and unknown facts. We can construct each type of fact. This is summarised below.

- The true facts are $I_\ell \cap I_u$.
- The false facts are $\mathcal{F} \setminus (I_\ell \cup I_u)$.
- The unknown facts are $I_u \setminus I_\ell$.
- The inconsistent facts are $I_\ell \setminus I_u$.

Remark 4.1. The terms lower and upper refer to which coordinate should be $\mathbf{t}$. For instance, the lower interpretable set consists of the true and inconsistent facts: the only logical facts for which the first coordinate is $\mathbf{t}$ in the above mentioned tuple notation are $\mathbf{t}$ and $\mathbf{i}$.

Now take an arbitrary subset I of $\mathcal{F}$. Let L be the set of logical facts in I. If $L = \{\mathbf{t},\mathbf{i}\}$, then I corresponds to the lower interpretable subset of a unique interpretation $\mathcal{I}_I$. Similar, if $L = \{\mathbf{t},\mathbf{u}\}$, then I corresponds to the upper interpretable subset of a unique interpretation $\mathcal{I}_I$. These are the only possibilities for interpretable subsets.

The set of interpretable[8] subsets of $\mathcal{F}$ is denoted as $\mathbb{I}_\mathcal{F}$ (or by $\mathbb{I}$ if $\mathcal{F}$ is clear from the context).

Definition 4.2. An interpretable subset I of $\mathcal{F}$ is *consistent* if $\mathcal{I}_I(x) = \mathbf{i}$ if and only if $x = \mathbf{i}$. An interpretable subset I of $\mathcal{F}$ is *co-consistent* if $\mathcal{I}_I(x) = \mathbf{u}$ if and only if $x = \mathbf{u}$. An interpretable subset I of $\mathcal{F}$ is *exact* if it is consistent and co-consistent.

For a given interpretation, the lower and upper interpretable subsets are related to each other by the following duality operator.

Definition 4.3. We define the *duality operator* on subsets of $\mathcal{F}$ as follows:

$$\mathrm{d}_\mathcal{F} : 2^\mathcal{F} \to 2^\mathcal{F} : I \mapsto \mathrm{d}_\mathcal{F}(I) \coloneqq \sim(\mathcal{F} \setminus I) = \mathcal{F} \setminus \sim I.$$

When $\mathcal{F}$ is clear from the context, we just write d for $\mathrm{d}_\mathcal{F}$. For an interpretation $\mathcal{I}$, we have that $\mathrm{d}(I_\ell) = I_u$ and $\mathrm{d}(I_u) = I_\ell$. In general the following holds.

Proposition 4.4. *1. The duality operator is involutive, that is* $\mathrm{d}(\mathrm{d}(I)) = I$ *for all* $I \subseteq \mathcal{F}$.

2. *The duality operator maps lower interpretable subsets to upper interpretable subsets and vice versa.*
3. *For I interpretable, it holds that $\mathcal{I}_{\mathrm{d}(I)} = \mathcal{I}_I$.*
4. *The duality operator preserves consistency and co-consistency of interpretable subsets.*
5. *The dual of an exact interpretable subset I differs from I only on the logical facts.*

Item (3) shows that the dual of an interpretable subset provides a different view on the same object (the same underlying interpretation), as is often the case with dualities in mathematics.

Remark 4.5. We effectively split an interpretation in a lower and upper interpretable subsets. Unfortunately, these concepts are less intuitive than interpretations. On the positive side, they simplify the arguments for proofs and enable us to state results that are not obvious in terms of interpretations. In order to relate to intuitions, we will rephrase most results on interpretable sets below in terms of interpretations.

Let $\mathcal{JS}$ be a justification system and $\mathcal{I}$ and interpretation of $\mathcal{F}$.

Definition 4.6. Let $\mathcal{JS}$ be a justification system, $x \in \mathcal{F}$ a fact, and $\mathcal{I}$ an interpretation of $\mathcal{F}$. If x is defined, the *supported value* of x in $\mathcal{I}$ by $\mathcal{JS}$ is defined as:

$$\mathrm{SV}_\mathcal{JS}(\mathcal{I}, x) \coloneqq \bigvee_{J \in \mathcal{J}_\mathcal{JS}(x)} \bigwedge_{\mathbf{b} \in B_J(x)} \mathcal{I}(\mathcal{B}(\mathbf{b})),$$

where $\mathcal{J}_\mathcal{JS}(x)$ is the set of locally complete justifications J in $\mathcal{JS}$ with $x \in \mathrm{dom}(J)$ and $B_J(x)$ is the set of J-branches starting in x. The meet and join are with respect to the truth order.

If x is open, the supported value of x in $\mathcal{I}$ by $\mathcal{JS}$ is defined as $\mathrm{SV}_\mathcal{JS}(\mathcal{I}, x) \coloneqq \mathcal{I}(x)$.

[8] In (Denecker, Brewka, and Strass 2015), Denecker et al. only considered lower interpretable subsets as input for the support operator. They called those sets interpretations. We chose for the terminology 'interpretable' to emphasise the difference with what is usually called an interpretation in logic.

This value can be seen as the value of the best justification for x. For a given justification J, $\bigwedge_{\mathbf{b}\in B_J(x)} \mathcal{I}(\mathcal{B}(\mathbf{b}))$ can intuitively be seen as the worst value of its branches.

It is not too difficult to see that the supported value is the same as the following tuple of two-valued truth values

$$(x \in \hat{\mathcal{S}}_{\mathcal{JS}}(I_\ell), x \in \hat{\mathcal{S}}_{\mathcal{JS}}(I_u)),$$

where the expression $x \in A$ is considered $\mathbf{t}$ if x is in A and $\mathbf{f}$ if not. As stated in the introduction, we want that the supported value commutes with negation, i.e. $\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, x) = \sim \mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \sim x)$. To prove this for all interpretations, this amounts to proving that x is supported in I if and only if $\sim x$ is not supported in $\mathrm{d}(I)$ for any interpretable set I. In algebraical terms, this is exactly

$$\mathrm{d}(\hat{\mathcal{S}}_{\mathcal{JS}}(I)) = \hat{\mathcal{S}}_{\mathcal{JS}}(\mathrm{d}(I))$$

for all interpretable subsets.

Remark 4.7. This rewording directly shows that our main theorem implies the duality between CNF and DNF formulas as mentioned in the introduction of this section.

If the supported value commutes with negation, then the supported value is the value of the interpretation corresponding to $\hat{\mathcal{S}}_{\mathcal{JS}}(I)$, which is well defined since $\hat{\mathcal{S}}_{\mathcal{JS}}(I)$ is interpretable since I is interpretable. In what follows we study this property and prove it for completion, Kripke-Kleene, stable and well-founded semantics. Moreover, we will prove it for co-stable and co-well-founded semantics, which we define in the next two subsections.

Signed justification frames

In many logical frameworks, there is an asymmetry between positive and negative literals, while in justification theory, facts and their negation are treated completely symmetrically. In this section, we show how a similar distinction can be introduced by means of a sign function. Consider for instance stable semantics of logic programs (Gelfond and Lifschitz 1988). There, the default value for atoms is false; as such, the default value for its negation is true. Thus, the set of defined literals is divided into two parts, those that are default and those that are not.[9] This idea is captured in signed justification frames.

Definition 4.8. Let $\mathcal{JF}$ be a justification frame. A *sign function* on $\mathcal{JF}$ is a map

$$\mathrm{sgn} : \mathcal{F}_d \to \{-, +\}$$

such that $\mathrm{sgn}(x) \neq \mathrm{sgn}(\sim x)$ for all $x \in \mathcal{F}_d$. We denote $\mathcal{F}_- := \mathrm{sgn}^{-1}(\{-\})$ and $\mathcal{F}_+ := \mathrm{sgn}^{-1}(\{+\})$. Remark that $\{\mathcal{F}_-, \mathcal{F}_+\}$ forms a partition of $\mathcal{F}_d$.

From now on, we fix a sign function on $\mathcal{JF}$. This structure is always present in a justification frame defined as in (Denecker, Brewka, and Strass 2015) and there it is a partition of whole $\mathcal{F}$. However, it raises the question which logical facts are positive and which are negative. To solve this, Denecker et al. introduced a positive and negative $\mathbf{u}$ and $\mathbf{i}$, which is an ad hoc solution and deviates from Belnap's four-valued logic (Belnap 1977). We abstain from deciding which logical fact is positive by only introducing it for defined facts.

The sign function is used to define new branch evaluations.

Definition 4.9. The *well-founded* (Van Gelder, Ross, and Schlipf 1991) branch evaluation $\mathcal{B}_{\mathrm{wf}}$ maps finite branches to their last element. Under $\mathcal{B}_{\mathrm{wf}}$, infinite branches are mapped to $\mathbf{t}$ if they have a negative tail, to $\mathbf{f}$ if they have a positive tail and to $\mathbf{u}$ otherwise.

The *stable* (Gelfond and Lifschitz 1988) (answer set) branch evaluation[10] $\mathcal{B}_{\mathrm{st}}$ maps a branch $x_0 \to x_1 \to \cdots$ to the first element that has a different sign than x_0 if it exists, otherwise to $\mathbf{t}$ if x_0 is negative and $\mathbf{f}$ if x_0 is positive. The only exception is that a finite branch $x_0 \to \cdots \to x_n$, with $x_0, \ldots, x_{n-1}$ having the same sign, is mapped to x_n.

The correspondence with the corresponding logic programming semantics is given in the following theorem from (Denecker, Brewka, and Strass 2015, Theorem 1).

Theorem 4.10 (Denecker et al.)**.** *To a propositional logic program* Π *we can associate a complementary justification frame* $\mathcal{JF}_\Pi$ *such that*

- *An exact lower interpretable subset* I *is an exact fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{sp}}}_{\mathcal{JF}_\Pi}$ *iff* I *is a supported model of* Π.
- *A lower interpretable subset* I *is a fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{KK}}}_{\mathcal{JF}_\Pi}$ *iff* I *is the Kripke-Kleene model of* Π.
- *A lower interpretable subset* I *is an exact fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{st}}}_{\mathcal{JF}_\Pi}$ *iff* I *is a stable model of* Π.
- *A lower interpretable subset* I *is a fixed point of* $\hat{\mathcal{S}}^{\mathcal{B}_{\mathrm{wf}}}_{\mathcal{JF}_\Pi}$ *iff* I *is the well-founded model of* Π.

Dual branch evaluations

Well-founded semantics accurately captures the semantics of inductive definitions (Denecker and Ternovska 2004; Denecker and Vennekens 2007; Denecker and Ternovska 2008; Denecker and Vennekens 2014). On the other hand, there are co-inductive definitions (Gupta et al. 2007). We define a semantics for co-induction by introducing a co-well-founded branch evaluation[11].

Definition 4.11. Let $\mathcal{JF}$ be a justification frame. Then the *co-well-founded* branch evaluation $\mathcal{B}_{\mathrm{cwf}}$ maps finite branches to their last element, branches with a positive tail to $\mathbf{t}$, branches with a negative tail to $\mathbf{f}$ and all other branches to $\mathbf{u}$.

After close inspection, this is exactly $\mathcal{B}_{\mathrm{wf}}$, but with an inverted sign function. This motivates the following. If $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R\rangle$ is a justification frame with sign function $\mathrm{sgn}_{\mathcal{JF}}$, then we denote $\overline{\mathcal{JF}}$ to be the justification frame $\langle \mathcal{F}, \mathcal{F}_d, R\rangle$ with sign function $\mathrm{sgn}_{\overline{\mathcal{JF}}} := -\mathrm{sgn}_{\mathcal{JF}}$. Each $\mathcal{JF}$-branch $\mathbf{b}$

[9] In some contexts, non-default literals are called *deviant* (Denecker, Bogaerts, and Vennekens 2018).

[10] This is different on finite branches than the stable branch evaluation introduced in (Denecker, Brewka, and Strass 2015); however, fixed points of the extended support operators are equal.

[11] Denecker et. al first described the co-well-founded branch evaluation in (Denecker, Brewka, and Strass 2015).

is a $\overline{\mathcal{JF}}$-branch. To make explicit the justification frame $\mathcal{JF}$ we regard a branch $\mathbf{b}$ in, we denote $\mathbf{b}$ as $\mathbf{b}_{\mathcal{JF}}$.

Definition 4.12. Let $\mathcal{B}$ be a branch evaluation. The *dual branch evaluation* $\overline{\mathcal{B}}$ of $\mathcal{B}$ is defined as follows:

$$\overline{\mathcal{B}}(\mathbf{b}_{\mathcal{JF}}) = \mathcal{B}(\mathbf{b}_{\overline{\mathcal{JF}}})$$

for any justification frame $\mathcal{JF}$ and $\mathcal{JF}$-branch $\mathbf{b}$.

We chose to call it dual semantics as induction and co-induction are dual concepts.

Example 4.13. The well-founded and co-well-founded branch evaluation are each others dual. ▲

Branch evaluations, such as $\mathcal{B}_{\mathrm{sp}}$ and $\mathcal{B}_{\mathrm{KK}}$, that do not depend on a sign function are self-dual. Apart from the co-well-founded branch evaluation, we constructed a new semantics: the *co-stable* branch evaluation $\mathcal{B}_{\mathrm{cst}} \coloneqq \overline{\mathcal{B}_{\mathrm{st}}}$. It differs with $\mathcal{B}_{\mathrm{st}}$ only on infinite branches having everywhere the same sign. In this case, it is the negation of $\mathcal{B}_{\mathrm{st}}$. Where stable semantics gives a default value of false to atoms, co-stable semantics gives a default value of true to atoms. Co-stable semantics could be useful in the context of stream reasoning (Beck, Eiter, and Folie 2017).

Justifications in dual semantics are related with justifications in the original semantics as illustrated in the following lemma.

Lemma 4.14. *Let $\mathcal{JS}$ be a justification system $\langle \mathcal{JF}, \mathcal{B} \rangle$ and let $\mathcal{JS}'$ be the justification system $\langle \overline{\mathcal{JF}}, \overline{\mathcal{B}} \rangle$. Then $\mathcal{S}_{\mathcal{JS}}(I) = \mathcal{S}_{\mathcal{JS}'}(I)$ for all subsets I of $\mathcal{F}$.*

While the proof of this lemma is (almost) trivial, it has a consequence that relates commutation of negation with the supported value between dual branch evaluations.

Corollary 4.15. *For a subset I of $\mathcal{F}$, the following are equivalent.*

1. *Under $\mathcal{B}$, x is supported in I if and only if $\sim x$ is not supported in $\mathrm{d}(I)$. ($\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(\mathrm{d}(I)) = \mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I))$)*
2. *Under $\overline{\mathcal{B}}$, x is supported in I if and only if $\sim x$ is not supported in $\mathrm{d}(I)$. ($\hat{\mathcal{S}}^{\overline{\mathcal{B}}}_{\mathcal{JF}}(\mathrm{d}(I)) = \mathrm{d}(\hat{\mathcal{S}}^{\overline{\mathcal{B}}}_{\mathcal{JF}}(I))$)*

Thus if we prove that the supported value commutes with negation for stable and well-founded semantics, we automatically have that the supported value of co-stable and co-well-founded semantics commutes with negation.

5 Branch evaluation classes

In this section, we study properties of various branch evaluations, in particular $\mathcal{B}_{\mathrm{sp}}$, $\mathcal{B}_{\mathrm{KK}}$, $\mathcal{B}_{\mathrm{wf}}$, $\mathcal{B}_{\mathrm{st}}$, and their duals. To do this systematically, we introduce novel classes of branch evaluations and study properties of them. In particular, we study so-called *splittable* branch evaluations. Such a branch evaluation can evaluate any branch either by only taking its start (first n elements) or its tail into account. Splittable branch evaluations allow us to "glue" justifications together and as such construct a single justification that supports $\mathcal{S}_{\mathcal{JS}}(I)$ in I.

For a finite branch $\mathbf{b} : x_0 \to x_1 \to \cdots \to x_n$, denote $\ell(\mathbf{b}) = n$. If $\mathbf{b}$ is infinite, we say $\ell(\mathbf{b}) = \infty$. Two branches $x_0 \to x_1 \to \cdots$ and $y_0 \to y_1 \to \cdots$ are *identical up to* n if for all $0 \leq i \leq n$, we have $x_i = y_i$. We say a $\mathcal{JF}$-branch $\mathbf{b} : x_0 \to x_1 \to \cdots$ is *decided* under $\mathcal{B}$ at $0 < n < \ell(\mathbf{b}) + 1$ if for every $\mathcal{JF}$-branch $\mathbf{b}'$ identical to $\mathbf{b}$ up to n, we have $\mathcal{B}(\mathbf{b}) = \mathcal{B}(\mathbf{b}')$. In this case, we call the $\mathcal{JF}$-path $x_0 \to \cdots \to x_n$ *decided* under $\mathcal{B}$. This means to determine the value of $\mathbf{b}$ under $\mathcal{B}$, we only need the $n+1$ first elements, so all relevant information is located at the beginning of the branch. This allows us to extend $\mathcal{B}$ to paths that are decided. On the other hand, we define $\mathbf{b}$ to be *transitive* under $\mathcal{B}$ at $0 \leq n < \ell(\mathbf{b})$ if

$$\mathcal{B}(\mathbf{b}) = \mathcal{B}(x_n \to x_{n+1} \to \cdots).$$

Intuitively, a branch is transitive if all information needed to evaluate it is located in the tail after n.

Definition 5.1. A branch $\mathbf{b}$ is called *splittable* under $\mathcal{B}$ at $0 \leq n < \ell(\mathbf{b}) + 1$ if it is either decided or transitive under $\mathcal{B}$ at n.

Intuitively, if a branch is splittable at n, then the information to evaluate the branch is either in tail or start, but not in both.

Remark 5.2. If $\mathcal{B}$ is clear from context, 'under $\mathcal{B}$' is left out.

Lemma 5.3. *If a branch $\mathbf{b}$ is decided at n, then it is also decided at m for $n \leq m < \ell(\mathbf{b}) + 1$.*

A branch $\mathbf{b}$ is called *first decided* at $0 < i < \ell(\mathbf{b}) + 1$ if $\mathbf{b}$ is decided at $i = 1$ or $\mathbf{b}$ is decided at $i > 1$ and $\mathbf{b}$ is not decided at $i - 1$.

Definition 5.4. We say that a branch $\mathbf{b}$ is *totally decided* if it is decided at i for every $0 < i < \ell(\mathbf{b}) + 1$. Similarly, $\mathbf{b}$ is *totally transitive* if it is transitive at i for every $0 \leq i < \ell(\mathbf{b})$. A branch $\mathbf{b}$ is *totally splittable* if it is splittable at i for every $0 \leq i < \ell(\mathbf{b}) + 1$.

Definition 5.5. A branch evaluation $\mathcal{B}$ is called

- *splittable*[12] if every $\mathcal{JF}$-branch is totally splittable;
- *transitive* if every $\mathcal{JF}$-branch is totally transitive;
- *decided* if every $\mathcal{JF}$-branch is totally decided;
- *parametric* if $\mathcal{B}(\mathbf{b}) \in \mathcal{F}_o$ for any $\mathcal{JF}$-branch $\mathbf{b}$;

for every justification frame $\mathcal{JF}$.

Lemma 5.6. *Any transitive or decided branch evaluation is splittable.*

Example 5.7. The branch evaluation $\mathcal{B}_{\mathrm{sp}}$ is decided, $\mathcal{B}_{\mathrm{KK}}$, $\mathcal{B}_{\mathrm{wf}}$ and $\mathcal{B}_{\mathrm{cwf}}$ are transitive and parametric, and $\mathcal{B}_{\mathrm{st}}$ and $\mathcal{B}_{\mathrm{cst}}$ are splittable. The branch evaluation $\mathcal{B}_{\mathrm{ex}}$ from Example 3.12 is transitive and parametric. ▲

Any branch $\mathbf{b} : x_0 \to x_1 \to \cdots$ has a least initial segment $\mathbf{b}^*$ that is decided. Indeed, if $\mathbf{b}$ is first decided at i, then $x_0 \to \cdots \to x_i$ is this segment. If $\mathbf{b}$ is nowhere decided, then $\mathbf{b}$ itself is this segment. Using this segment, we can give an equivalent characterisation of splittable.

Proposition 5.8. *A branch evaluation $\mathcal{B}$ is splittable if and only if for every branch $\mathbf{b}$, we have that the final segment of $\mathbf{b}^*$ is decided and has the same evaluation as $\mathbf{b}$.*

[12] The concept of splittable branch evaluations was first introduced in the master's thesis of the second author (Passchyn 2017).

Corollary 5.9. *Let $\mathcal{B}$ be a splittable branch evaluation. If a branch $\mathbf{b}$ is first decided at j. Then for all $0 \leq i < j$ the path $x_i \to \cdots \to x_j$ is decided and has the same evaluation as $\mathbf{b}$.*

Transitive justification systems over complementary justification frames are local in some sense.

Lemma 5.10. *Let $\mathcal{B}$ be a transitive branch evaluation such that finite branches are mapped to their last element. Let $\mathcal{JF}$ be a complementary justification frame Then for any subset I of $\mathcal{F}$ and any $x \in \mathcal{F}_d$*

- *$x \in \mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}(I)$ if and only if there is an $A \in \mathcal{JF}(x)$ such that $A \subseteq \hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I)$;*
- *$\sim x \notin \mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}(I)$ if and only if there is an $A \in \mathcal{JF}(x)$ such that $A \subseteq \mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I))$.*

The condition on finite branches in Lemma 5.10 is necessary as shown by the next example.

Example 5.11. Let $\mathcal{F}_d = \{a, \sim a\}$, $\mathcal{F}_o = \mathcal{L} \cup \{b, \sim b\}$ and $R = \{a \leftarrow b; \sim a \leftarrow \sim b\}$. Take the justification system $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}\rangle$ with $\mathcal{B}$ be a branch evaluation over $\mathcal{JF}$ such that $\mathcal{B} = \mathcal{B}_{\mathrm{KK}}$ on infinite branches and $\mathcal{B}(\mathbf{b}) = \sim\mathcal{B}_{\mathrm{KK}}(\mathbf{b})$ for finite branches $\mathbf{b}$. This branch evaluation is transitive and parametric. Take $I = \{\mathbf{t}, \mathbf{i}, b\}$. There exists an $A \in \mathcal{JF}(a)$ such that $A \subseteq \hat{\mathcal{S}}_{\mathcal{JS}}(I) = \{\mathbf{t}, \mathbf{i}, b, \sim a\}$, i.e. $A = \{b\}$. However, $a \notin \mathcal{S}_{\mathcal{JS}}(I) = \{\sim a\}$. ▲

6 Pasting justifications

In this section we define an operation that "glues" two justifications together. We investigate how branches are related after the gluing.

If a justification J is a subset of a justification K, then we say that K extends J or that J is a restriction of K. The union of two justifications is not necessarily a justification. However, the union of two justifications with disjoint domain is a justification.

Definition 6.1. The *extension* of J with K is defined as

$$(J \uparrow K)(x) \coloneqq \begin{cases} J(x) & \text{if } x \in \mathrm{dom}(J); \\ K(x) & \text{if } x \in \mathrm{dom}(K) \setminus \mathrm{dom}(J). \end{cases}$$

It is clear that $J \uparrow K$ is an extension of J.

Branches of the extension of two locally completely justifications behave nicely.

Lemma 6.2. *If J and K are two locally complete justifications, then $J \uparrow K$ is locally complete. Moreover, every $(J \uparrow K)$-branch either is a K-branch or is a concatenation of a K-path with a J-branch. (K-path means a path in G_K)*

The next lemma illustrates the power of splittable branch evaluations.

Lemma 6.3. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}\rangle$ be a splittable justification system. Take a subset I of $\mathcal{F}$. Let J and K be two locally complete justifications that support their domain in I under $\mathcal{JS}$, then $J \uparrow K$ also supports its domain in I under $\mathcal{JS}$.*

This result also holds for non locally complete justifications.

Lemma 6.4. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}\rangle$ be a splittable justification system. Take a subset I of $\mathcal{F}$. Let J and K be arbitrary justifications that support their domain in I under $\mathcal{JS}$, then $J \uparrow K$ also supports its domain in I under $\mathcal{JS}$.*

A *root* of a graph is a node such that all other nodes can be reached with a path starting in this root. We say a justification J is *root-supported* in I if it has a root that is supported by J in I. Transitive branch evaluations act well with root-supported justifications.

Lemma 6.5. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}\rangle$ be a justification system with $\mathcal{B}$ transitive. An I-root-supported justification supports its domain in I.*

The following proposition allows us to paste together an arbitrary number of justifications that support their domain into a justification that also supports its domain.

Proposition 6.6. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}\rangle$ be a splittable justification system. Take a subset I of $\mathcal{F}$. Let X be a subset of $\mathcal{F}_d$ such that for any $x \in X$, there exists a (locally complete) justification J_x that supports its domain in I. Then there exists a (locally complete) justification J that supports its domain $\mathrm{dom}(J) = \cup_{x \in X} \mathrm{dom}(J_x)$ in I and for any $y \in \mathrm{dom}(J)$, it holds that $J(y) = J_x(y)$ for some $x \in X$.*

Proof. The proposition is trivial if $X = \emptyset$, so assume $X \neq \emptyset$. We do a proof by transfinite induction. By the well-ordering theorem[13], we can fix a well-order on $X = \{x_i \mid i \leq \beta\}$ for some ordinal β. Define

- $K_0 = J_{x_0}$;
- $K_{i+1} = K_i \uparrow J_{x_{i+1}}$ for any ordinal $i < \beta$;
- $K_\alpha = \cup_{i<\alpha} K_i$ for any limit ordinal $\alpha \leq \beta$.

The zero case trivially supports its domain and the successor case by Lemma 6.4. For a limit ordinal $\alpha \leq \beta$, take a locally complete extension L of K_α and $x \in \mathrm{dom}(K_\alpha)$. There is an $i < \alpha$ such that $x \in \mathrm{dom}(K_i)$. Since x is supported by K_i in I and L is an extension of K_α and thus also of K_i, we have that x is supported by L in I. Because L is taken arbitrary, x is supported by K_α in I; hence K_α supports its domain in I.

The condition that for $y \in \mathrm{dom}(K_i)$, it holds that $K_i(y) = J_x(y)$ for some $x \in X$ is trivial for all cases by the definition of $\uparrow$ and $\cup$.

The zero case is trivially locally complete, the successor case by Lemma 6.2. Suppose by contradiction that the limit case does not preserve locally completeness. Thus, there exist defined facts x, y such that $x \in K_\alpha(y)$, but $x \notin \mathrm{dom}(K_\alpha)$. There is an $i < \alpha$ so that $K_\alpha(y) = K_i(y)$. Then this means that x is a leaf of K_i, which is a contradiction, that is, the limit case preserves locally completeness.

Finally, set $J = K_\beta$. □

This result enables to construct a justification that support $\mathcal{S}_{\mathcal{JS}}(I)$ in I under $\mathcal{JS}$.

[13] This is equivalent to the axiom of choice.

Corollary 6.7. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a splittable justification system. For any subset I of $\mathcal{F}$, there exists a justification J that supports its domain $\mathrm{dom}(J) = \mathcal{S}_{\mathcal{JS}}(I)$ in I. Moreover, if $\mathcal{B}$ is transitive, J can be taken to be locally complete.*

If $\mathcal{B}$ is not splittable, then such a justification as in Corollary 6.7 does not necessarily exist as shown by the following example.

Example 6.8. Let $\mathcal{B}$ be the branch evaluation that maps $x_0 \to x_1 \to x_2 \to \cdots$ to x_2 if it exists and to $\sim x_1$ otherwise. Let $\mathcal{F}_d = \{a, b, \sim a, \sim b\}$ and $\mathcal{F}_o = \mathcal{L} \cup \{c, d, \sim c, \sim d\}$. Let $R = \{a \leftarrow b; b \leftarrow c; b \leftarrow d\}$. Define $\mathcal{JF} = \langle \mathcal{F}, \mathcal{F}_d, R \cup R^* \rangle$ and take $I = \{\mathbf{t}, \mathbf{u}, c, \sim d\}$. Then $\mathcal{S}^{\mathcal{B}}_{\mathcal{JF}}(I) = \{a, b\}$. However, the only connected justifications that contain a and b in its domain are

$$\begin{array}{cc} a & a \\ \downarrow & \downarrow \\ b & b \\ \downarrow & \downarrow \\ c & d \end{array}$$

These justifications do not simultaneously support a and b in I. ▲

If $\mathcal{B}$ is not transitive, then there does not necessarily exist a locally complete justification that supports its domain $\mathrm{dom}(J) = \mathcal{S}_{\mathcal{JS}}(I)$ in I. That is, the moreover part of Corollary 6.7 does not hold for non-transitive branch evaluations. This is illustrated in the following example.

Example 6.9. Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B}_{\mathrm{sp}} \rangle$ be the justification system with $\mathcal{F}_d = \{a, \sim a, b, \sim b\}$, $\mathcal{F}_o = \mathcal{L} \cup \{c, \sim c\}$ $R = \{a \leftarrow b; b \leftarrow c; \sim a \leftarrow \sim b; \sim b \leftarrow \sim c\}$. Take $I = \{a, b, \sim c\}$, then $\mathcal{S}^{\mathcal{B}_{\mathrm{sp}}}_{\mathcal{JF}}(I) = \{a, \sim b\}$. Now if a locally complete justification has a in its domain, then it has b in its domain. Therefore there is no locally complete justification with domain $\mathcal{S}^{\mathcal{B}_{\mathrm{sp}}}_{\mathcal{JF}}(I)$ that supports its domain. ▲

However, we can still find a locally complete justification J that supports $\mathcal{S}_{\mathcal{JS}}(I)$ in I, i.e. any locally complete extension of a justification from Corollary 6.7.

The justification found in Corollary 6.7 is a conglomerated explanation for all supported facts and will prove to be useful later on; particularly in the proof of our main theorem.

7 Consistency of supported value

A justification that supports x in I gives an explanation why x is valid. If we have a justification for x, we expect not to find a justification for $\sim x$. Under reasonable assumptions this holds.

Proposition 7.1. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a complementary justification system with $\mathcal{B}$ consistent. Take a subset I of F. If $x \in \mathcal{F}_d$ is supported by $\mathcal{JS}$ in I, then $\sim x$ is not supported by $\mathcal{JS}$ in $\mathrm{d}(I)$.*

If I is a lower interpretable set, then Proposition 7.1 states that if the supported value of x is $\mathbf{t}$ or $\mathbf{i}$, then the supported value of $\sim x$ is not equal to $\mathbf{t}$ or $\mathbf{u}$. These two statements do not conflict each other. If I is an upper interpretable set, this is the same but $\mathbf{u}$ and $\mathbf{i}$ swapped.

Proposition 7.1 is equivalent with

$$\mathcal{S}_{\mathcal{JS}}(I) \subseteq \mathrm{d}(\mathcal{S}_{\mathcal{JS}}(\mathrm{d}(I))).$$

This in turn implies that

$$\hat{\mathcal{S}}_{\mathcal{JS}}(\mathrm{d}(I)) \subseteq \mathrm{d}(\hat{\mathcal{S}}_{\mathcal{JS}}(I)).$$

As a consequence we get a partial result concerning supported values.

Corollary 7.2. *If $\mathcal{JS}$ is a complementary justification system with a consistent branch evaluation, then*

$$\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, x) \leq_t \sim \mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \sim x)$$

for any $x \in \mathcal{F}$ and interpretation $\mathcal{I}$ of $\mathcal{F}$, and where $\leq_t$ denotes the truth order.

The converse of Proposition 7.1 does not always hold as illustrated in Example 3.12. For some branch evaluations the converse does hold.

Proposition 7.3. *Let $\mathcal{JF}$ be a complementary justification frame. Then for $\mathcal{B} \in \{\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{st}}, \mathcal{B}_{\mathrm{KK}}, \mathcal{B}_{\mathrm{wf}}\}$ it holds that if $\sim x$ is not supported in I, then x is supported in $\mathrm{d}(I)$ for all interpretable subsets I of $\mathcal{F}$.*

The proof heavily depends on Corollary 6.7 and can be found in the extended version of this text. Intuitively, if I is a lower interpretable set, Proposition 7.3 states that if the supported value of x is $\mathbf{f}$ or $\mathbf{u}$, then the supported value of $\sim x$ is $\mathbf{t}$ or $\mathbf{i}$. If I is an upper interpretable set, Proposition 7.3 states that if the supported value of x is $\mathbf{f}$ or $\mathbf{i}$, then the supported values of $\sim x$ is $\mathbf{t}$ or $\mathbf{u}$. Proposition 7.3 is equivalent with

$$\mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I)) \subseteq \hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(\mathrm{d}(I))$$

Combining Proposition 7.1 and Proposition 7.3, we get that the following theorem.

Theorem 7.4. *Let $\mathcal{JF}$ be a complementary justification frame. Let I be an interpretable subset of $\mathcal{F}$. Then $x \in \mathcal{F}_d$ is supported in I under $\mathcal{B}$ if and only if $\sim x$ is not supported in $\mathrm{d}(I)$ under $\mathcal{B}$ for $\mathcal{B} \in \{\mathcal{B}_{\mathrm{sp}}, \mathcal{B}_{\mathrm{KK}}, \mathcal{B}_{\mathrm{st}}, \mathcal{B}_{\mathrm{wf}}\}$. This is equivalent with $\mathrm{d}(\hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(I)) = \hat{\mathcal{S}}^{\mathcal{B}}_{\mathcal{JF}}(\mathrm{d}(I))$.*

Corollary 7.5. *For the justification systems above, the supported value commutes with negation, i.e.*

$$\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \sim x) = \sim \mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, x)$$

for all interpretations $\mathcal{I}$ of $\mathcal{F}$.

This means that $\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \cdot)$ is an interpretation of $\mathcal{F}$. This result is what one intuitively expects. However, it is not too difficult to come up with an example in which this intuition does not hold.

Example 7.6. *Continuation of Example 3.12.* For any interpretation $\mathcal{I}$ of $\mathcal{F}$, we have that $\mathrm{SV}(\mathcal{I}, a) = \mathbf{f}$, while $\mathrm{SV}(\mathcal{I}, \sim a) = \mathbf{u}$. Therefore, $\mathrm{SV}(\mathcal{I}, \cdot)$ is not an interpretation of $\mathcal{F}$. ▲

The branch evaluation in this example induces a defective semantics since it has a counter-intuitive notion of support. Our expectation is that the supported value is always an interpretation and equal to the one corresponding to $\hat{\mathcal{S}}_{\mathcal{JS}}(I)$ as lower bound.

A similar result is already known for tree-like justifications for three-valued completion, stable and well-founded semantics, see (Denecker and De Schreye 1993).

By Corollary 4.15, we have that Theorem 7.4 also holds for $\mathcal{B}_{\text{cwf}}$ and $\mathcal{B}_{\text{cst}}$. In justification systems where the supported value commutes with negation, we have in most cases that the extended support operator preserves consistency of interpretable subsets.

Proposition 7.7. *Let $\mathcal{JS} = \langle \mathcal{F}, \mathcal{F}_d, R, \mathcal{B} \rangle$ be a justification system such that $\mathrm{d}(\hat{\mathcal{S}}_{\mathcal{JS}}(I)) = \hat{\mathcal{S}}_{\mathcal{JS}}(\mathrm{d}(I))$ for all interpretable subsets I of $\mathcal{F}$. Suppose $\mathcal{B}$ does not map $\mathcal{JF}$-branches to* **i**. *Then $\hat{\mathcal{S}}_{\mathcal{JS}}$ preserves consistency of interpretable subsets.*

Remark 7.8. For a branch evaluation $\mathcal{B}$ that maps finite branches to their last element, $\mathcal{B}$ not mapping $\mathcal{JF}$-branches to **i** implies that rules cannot have **i** in their body.

Proposition 7.7 states that if an interpretation $\mathcal{I}$ is three-valued, then $\mathrm{SV}_{\mathcal{JS}}(\mathcal{I}, \cdot)$ is a three-valued interpretation.

8 Conclusion

Justification theory is an abstract framework to define and study semantics of non-monotonic logics. This includes various types of logic programs, abstract argumentation, inductive and co-inductive definitions, and nested definitions. Next to being useful to define semantics of logics, the framework can also be used when solving practical problems, for instance for computing unfounded sets in modern ASP solvers (Gebser, Kaufmann, and Schaub 2012; Mariën et al. 2008), relevance checking (Jansen et al. 2016), lazy grounding algorithms (De Cat et al. 2015; Bogaerts and Weinzierl 2018), and provenance systems for databases (Damásio, Analyti, and Antoniou 2013).

Complementary justification frames arise naturally for a range of formalisms, such as, abstract argumentation and the duality between CNF and DNF formulas. In many systems, rules are only defined for atoms, such as, in logic programming, while in abstract argumentation rules are only defined for negative literals. These rules can be transformed into a complementary justification frame.

The introduced dual branch evaluations provide a systematic way to define a dual semantics in justification theory. To the extent of our knowledge, we are the first to define such duality. In particular, it shows the relation between inductive and co-inductive logic programs.

By introducing splittable branch evaluations, we have found a general class in which supporting justifications can be pasted. An interesting question is if there is a more general class of branch evaluations that satisfies Proposition 6.6 and Corollary 6.7.

This pasting is then used to prove that the supported value commutes with negation for completion, Kripke-Kleene, stable and well-founded semantics. Using our duality result, we also get it for co-stable and co-well-founded semantics. A similar result was proven for tree-like justifications in a three-valued setting in (Denecker and De Schreye 1993). An interesting question for future work is to find a complete characterisation for when the supported value commutes with negation. This would give a characterisation which branch evaluations do not induce a defective semantics.

References

Beck, H.; Eiter, T.; and Folie, C. 2017. Ticker: A system for incremental asp-based stream reasoning. *TPLP* 17(5-6):744–763.

Belnap, N. D. 1977. A useful four-valued logic. In *Modern Uses of Multiple-Valued Logic*. Reidel, Dordrecht. 8–37. Invited papers from ISMVL.

Bogaerts, B., and Weinzierl, A. 2018. Exploiting justifications for lazy grounding of answer set programs. In *Proceedings of IJCAI*, 1737–1745.

Cabalar, P.; Fandinno, J.; and Fink, M. 2014. Causal graph justifications of logic programs. *TPLP* 14(4–5):603–618.

Clark, K. L. 1978. Negation as failure. In *Logic and Data Bases*, 293–322. Plenum Press.

Damásio, C. V.; Analyti, A.; and Antoniou, G. 2013. Justifications for logic programming. In Cabalar, P., and Son, T. C., eds., *Logic Programming and Nonmonotonic Reasoning, 12th International Conference, LPNMR 2013, Corunna, Spain, September 15-19, 2013. Proceedings*, volume 8148 of *LNCS*, 530–542. Springer.

De Cat, B.; Bogaerts, B.; Devriendt, J.; and Denecker, M. 2013. Model expansion in the presence of function symbols using constraint programming. In *Proceedings of ICTAI*, 1068–1075.

De Cat, B.; Denecker, M.; Bruynooghe, M.; and Stuckey, P. J. 2015. Lazy model expansion: Interleaving grounding with search. *J. Artif. Intell. Res. (JAIR)* 52:235–286.

Denecker, M., and De Schreye, D. 1993. Justification semantics: A unifying framework for the semantics of logic programs. In *Proceedings of LPNMR*, 365–379.

Denecker, M., and Ternovska, E. 2004. A logic of nonmonotone inductive definitions and its modularity properties. In Lifschitz, V., and Niemelä, I., eds., *LPNMR*, volume 2923 of *LNCS*, 47–60. Springer.

Denecker, M., and Ternovska, E. 2008. A logic of nonmonotone inductive definitions. *ACM Trans. Comput. Log.* 9(2):14:1–14:52.

Denecker, M., and Vennekens, J. 2007. Well-founded semantics and the algebraic theory of non-monotone inductive definitions. In *Proceedings of LPNMR*, 84–96.

Denecker, M., and Vennekens, J. 2014. The well-founded semantics is the principle of inductive definition, revisited. In *Proceedings of KR*, 22–31.

Denecker, M.; Bogaerts, B.; and Vennekens, J. 2018. Causal reasoning in a logic with possible causal process semantics. under review.

Denecker, M.; Brewka, G.; and Strass, H. 2015. A formal theory of justifications. In *Proceedings of LPNMR*, 250–264.

Denecker, M.; Marek, V.; and Truszczyński, M. 2000. Approximations, stable operators, well-founded fixpoints and

applications in nonmonotonic reasoning. In *Logic-Based Artificial Intelligence, Springer*, volume 597, 127–144.

Denecker, M.; Marek, V.; and Truszczyński, M. 2011. Reiter's default logic is a logic of autoepistemic reasoning and a good one, too. In *Nonmonotonic Reasoning – Essays Celebrating Its 30th Anniversary*. College Publications. 111–144.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *AIJ* 77(2):321 – 357.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *AIJ* 187:52–89.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of ICLP/SLP*, 1070–1080.

Gupta, G.; Bansal, A.; Min, R.; Simon, L.; and Mallya, A. 2007. Coinductive logic programming and its applications. In Dahl, V., and Niemelä, I., eds., *ICLP*, volume 4670 of *LNCS*, 27–44. Springer.

Harrison, A.; Lifschitz, V.; and Yang, F. 2013. On the semantics of gringo. *CoRR* abs/1312.6149.

Jansen, J.; Bogaerts, B.; Devriendt, J.; Janssens, G.; and Denecker, M. 2016. Relevance for SAT(ID). In *Proceedings of IJCAI*, 596–603.

Marek, V., and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*. Springer-Verlag. 375–398.

Mariën, M.; Wittocx, J.; Denecker, M.; and Bruynooghe, M. 2008. SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In *Proceedings of SAT*, 211–224.

Passchyn, N. 2017. Justification frames. introducing splittable branch evaluations. Master Thesis; Denecker, Marc (supervisor).

van Emden, M. H., and Kowalski, R. A. 1976. The semantics of predicate logic as a programming language. *J. ACM* 23(4):733–742.

Van Gelder, A.; Ross, K. A.; and Schlipf, J. S. 1991. The well-founded semantics for general logic programs. *J. ACM* 38(3):620–650.

Towards a boolean dynamical system representation in a monmonotonic modal logic

Pierre Siegel[1], **Andrei Doncescu**[2], **Vincent Risch**[1], **Sylvain Sené**[1]
[1] Aix-Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
[2] Université Paul Sabatier CNRS, LAAS, Toulouse, France
{pierre.siegel,vincent.risch,sylvain.sene}@univ-amu.fr, andrei.doncescu@laas.fr

Abstract

Boolean dynamical systems (BDSs) represent the evolution of interactions inside a finite network of entities taking Boolean states over discrete time. These networks are classically used to model interactions of biological networks. In this context, a genetic network can be represented by both a Transition Graph (TG) and an Interaction Graph (IG). The precise relationship between IG and TG has been studied for many years in dynamical systems theory while still an open question. The global purpose of this article is to further study this relationship *via* a logical representation of BDSs into a nonmonotonic modal logic called Hypothesis Logic ($\mathcal{H}$). While the *dynamics* of a BDS are characterized by a function f, an important part of the studies focused on the analysis of both stable configurations (i.e. fixed points of f), and stable/unstable cycles of f. For the representation of some genetic networks with no negative feedback circuits, results were previously obtained with some well known nonmonotonic formalisms. So far however, BDSs representation by most of these formalisms does not permit to capture cyclic dynamical behaviors. Notably, the equivalent of a negative circuit has no extension in default logic (DL). This is embarrassing because these cycles may represent real interactions in living organisms like the cell cycle. This possible lack of extensions in DL was studied in $\mathcal{H}$, for which theories always have extensions while some of these, called *ghost extensions*, are actually not extensions of the corresponding theories in DL. This paper addresses to the question of a first representation of the dynamics of BDSs with $\mathcal{H}$, and ghost extensions appear to be a powerful tool in this respect. As we are especially concerned with cycles, it provides us with hints of simple algorithms for computing exhaustively both stable/unstable cycles and fixed points: distinguishing between stable/unstable as well as enumerating all the solutions in practice would be a major advance that would lead to apprehend better inner fundamental aspects in biology.

1 Introduction

Biological networks are representations of the bioprocesses on three levels of interactions into the biological cell: genomic, protein and metabolic level. The techniques which are the most used in Network Modeling are boolean networks, bayesian belief network and metabolic network modeling methods. The boolean network are well adapted for Gene Regulatory Networks. At this level, it is considered the genes are on-off switches which do not act independently. Two genes are connected if the expression of one gene modulates the expression of another one by activation or inhibition. It is interesting to remark some genes control the response of the cell to changes in the environment by regulating other genes.

From a logical point of view, a biological system can be considered as a set of interacting elements changing along discrete time. Genetic networks are specific biological systems that represent how the proteins/genes in a cell interact for the survival, reproduction, or death of this cell. These interactions can be studied in the context of automata networks and Boolean dynamical systems (BDSs) as a set of entities taking Boolean states. Founding theorems (Demongeot, Noual, and Sené 2012; Melliti et al. 2015; 2013; Remy et al. 2003; Remy, Ruet, and Thieffry 2008; Richard 2010; Richard and Comet 2007; Robert 1986; Thomas 1981) have focused on feedback circuits (simply called circuits hereafter).

A genetic network can be represented by both a *transition graph* (TG) and an *interaction graph* (IG). The relationship between TG and IG has been studied for many years in dynamical systems theory, but remains an important open question. This paper adresses this question with help of a representation of BDSs *via* a nonmonotonic modal logic called Hypothesis Logic ($\mathcal{H}$): our logic-based approach seems to be a step toward a global clarification of this relationship. Preliminary results were already given in (Siegel et al. 2017).

The study of genetic networks is a source of relevant questions regarding knowledge representation. First, interactions appear as a form of causality; as such, we expect to model it thanks to logical inferences, but of which kind? The use of classical logic is inadequate in this context because it cannot deal with inconsistencies. Moreover, what we learn arises largely from long and expensive experiments. Hence, we know only a small part of the interactions while this knowledge can be revisable, uncertain, contradictory and even false. Eventually, algorithmic complexity is a crucial issue regarding the need to provide algorithms with reasonable calculation times in practice. These questions have been studied in artificial intelligence since the late 1970s, especially by the use of both particular nonmonotonic logics and techniques derived from constraint programming. In particular it is possible to use default logic (DL) (Reiter 1980) and

answer set programming formalism (ASP) (Lifschitz 1999).

The *dynamics* of a BDS are characterized by a function f and an updating mode. An important part of the studies done on BDSs focused on the analysis of both stable configurations (i.e. fixed points of f), and stable/unstable cycles of f. For the representation of some genetic networks with no negative feedback circuits, results have been obtained with DL (Doncescu, Siegel, and Le 2014),(Doncescu and Siegel 2015). Representation of a BDS by whatever DL, ASP or other nonmonotonic formalisms, enables to find fixed points. But, these representations are not suitable for finding cycles. Notably, the equivalent of a negative circuit has no extensions in default logic. This is embarrassing because these cycles may represent real interactions in living organisms like the cell cycle (Davidich and Bornholdt 2008; Li et al. 2004), or the circadian cycle (Akman et al. 2012; Roenneberg and Merrow 2003). This possible lack of extensions in default logic has been fully studied in the context of *hypothesis logic* ($\mathcal{H}$) (Schwind and Siegel 1994; Siegel and Schwind 1993). In this logic, theories always have extensions while some of these, called *ghost extensions*, are actually not extensions of the corresponding theories in default logic. Moreover, very simple and efficient algorithms, used for solving SAT problems, can be applied to the computation of extensions, fixed points and cycles.

The representation of the dynamics of BDSs in $\mathcal{H}$ aims at making possible to discriminate between stable configurations (fixed points), limit cycles and unstable cycles. We introduce representations for both Interaction Graphs and Asynchronous Transition Graphs in $\mathcal{H}$, which allows us to exhibit new formal results. Ghost extensions play here a key role. This approach provides us with hints of simple algorithms for distinguishing between stable/unstable as well as enumerating all the solutions in practice. This would be a major advance regarding better inner fundamental aspects in biology.

In what follows, Section 2 presents basic definitions for BDSs and Section 3 reminds the basics of nonmonotonic and hypothesis logic. Section 4 gives a representation of IGs into $\mathcal{H}$ and prove some properties related to this representation. Section 5 studies relationships between ATGs of a BDS and $\mathcal{H}$: Theorem 3 states that there exists an isomorphism between stable configurations and stable extensions, and Theorem 4 states that every negative feedback circuit admits a set of ghost extensions whose semantics is analogous to BDS dynamical cycles.

2 Finite Boolean dynamical systems

A finite discrete dynamical system (DDS) describes the evolution of the interactions in a network of n entities numbered from 1 to n, over discrete time. This evolution is the *dynamics* of the system. An example of such a system is the representation of genetic networks, namely networks representing interactions between the genes or the proteins of a cell (Aracena et al. 2006; Demongeot et al. 2011; 2010; Fauré et al. 2006; Kauffman et al. 2003; Mendoza, Thieffry, and Alvarez-Buylla 1999). In the context of genetic networks, an entity $i \in \{1, ..., n\}$ depicts a protein whose concentration is denoted by x_i so that there is only a finite number of possible concentrations. In such networks, given a protein i, a set of interactions for proteins on i gives the conditions for this set to increase or decrease the concentration of i.

Regarding *Boolean dynamical systems* (BDSs) studied in this paper, the concentrations x_i are in $\{0, 1\}$. In this case, $x_i = 1$ denotes the presence of i and $x_i = 0$ its absence. Yet $x_i = 1$ (resp. $x_i = 0$) is the $activation$ (resp. $inhibition$) or the $production$ (resp. $destruction$) of the protein.

Note that what is written $x_i = 0$ (resp. $x_i = 1$) in the context of BDS corresponds logically to $v(i) = 0$ (resp. $v(i) = 1$), where v is the standard valuation function of propositional logic. Hence, a protein i is nothing else than a propositional variable. Abusing the notations we authorize ourselves to write either $i, \neg i$ as well as $x_i, \neg x_i$ depending on the context. For instance, $x = (\neg x_1, x_2, \neg x_3)$ corresponds to $x = (\neg 1, 2, \neg 3)$ or even simply $(\overline{1}, 2, \overline{3})$.

Consider $V = \{1, ..., n\}$ a set of n entities. A *configuration* $x = (x_1, ..., x_n)$ of the system is an assignment of a truth value $x_i \in \{0, 1\}$ to each element i of V. The set of all configurations (Jacob and Monod 1978), also called the *space of configurations*, is denoted by $X = \{0, 1\}^n$. The *dynamics* of such a system is modeled via both a function f, called the *global transition function*, and an *updating mode* that defines how the elements of V are updated along time. More formally, $f : X \to X$ is such that $x = (x_1, ..., x_n) \mapsto f(x) = (f_1(x), ..., f_n(x))$, where each function $f_i : X \to \{0, 1\}$ is *a local transition function* that gives the evolution of the state x_i over time. There exists an infinite number[1] of updating modes among which the *parallel* and the *asynchonous* ones remain the most used. The *parallel* (or *perfectly synchronous*) updating mode is such that all the entities of the network are updated at each time step. Conversely, the *asynchronous* updating mode is a non-deterministic variation in which only one entity is updated at a time. In the sequel, we restrict our study to asynchronous dynamics (Melliti et al. 2015; Remy, Ruet, and Thieffry 2008; Richard and Comet 2007).

2.1 Asynchronous transition graphs

Every dynamic being characterized by a function f and an updating mode, an important part of studies done on BDSs focused on the analysis of both stable configurations (i.e. fixed points of f), and stable/unstable cycles of f. Regarding the asynchronous case, this study is fulfilled via the notion of *Asynchronous Transition Graph* (ATG) associated with f.

Let $X = \{0, 1\}^n$ be the configuration space and consider a function $f : X \to X$. The *asynchronous dynamics* of f is given by its ATG $\mathscr{G}(f) = (X, T(f))$, the digraph whose vertex set is the configuration space and arc set is the set of transitions: $T(f) = \{(x, y) \in X^2 \mid x \neq y, x = (x_1, ..., x_i, ..., x_n), y = (x_1, ..., x_{i-1}, f_i(x), x_{i+1}, ..., x_n)\}$

If $(x, y) \in T(f)$, then the Hamming distance between x and y equals 1 (the transition is *unitary*). An *orbit* in $\mathscr{G}(f)$ is a sequence of configurations $(x^0, x^1, x^2, ...)$ such that either $(x^t, x^{t+1}) \in T(f)$ or $x^{t+1} = x^t$, if $x^t = f(x^t)$ (*i.e.*, x^t has no successors). A *cycle* of length r is a sequence of configurations $(x^1, ..., x^r, x^1)$ with $r \geq 2$ whose configura-

[1] Infinite, because deterministic updating modes are basically defined as infinite sequences of subsets of nodes of the network.

tions $x^1, ..., x^r$ are all different. From this, we derive what is classically called an asynchronous *attractor* in dynamical systems, namely a terminal *strongly connected component* (SCC) of $\mathscr{G}(f)$, *i.e.* a SCC with no outward transitions. Among attractors, in the sequel, we will pay particular attention to stable configurations (fixed points) and cycles. A *stable configuration* is a trivial attractor, *i.e.* a configuration x such that $\forall i \in V, x_i = f_i(x)$, which implies that $x = f(x)$. A *stable cycle* is a cyclic attractor such that, on $\mathscr{G}(f)$, $\forall t < r$, x^{t+1} is the unique successor of x^t and x^1 is the unique successor of x^r. If an attractor is neither trivial nor cyclic, it is called a *stable oscillation*. When it is possible to get out from a SCC, this SCC is called an unstable cycle or oscillation depending on which it is cyclic or not.

An orbit that reaches a stable configuration stays there endlessly. Similarly, if it reaches a stable cycle, it adopts endlessly a stable oscillating behavior.

Example 1 (Boolean positive and negative circuit of size 3) Consider $V = \{1, 2, 3\}$, $X = \{0, 1\}^3$ and the two following functions f and g such that $f(x_1, x_2, x_3) = (\neg x_2, \neg x_3, x_1)$ and $g(x_1, x_2, x_3) = (\neg x_3, x_1, x_2)$. From the functions f and g, it is easy to derive their related ATGs, $\mathscr{G}(f)$ and $\mathscr{G}(g)$, pictured in Figure 1. For each arc (x, y) in $\mathscr{G}(f)$ and $\mathscr{G}(g)$, if $x \neq y$ then x differs from y by a single component. There are up to 3 transitions leaving each configuration. Here, $\mathscr{G}(f)$ has two stable configurations, $(\neg 1, 2, \neg 3)$ and $(1, \neg 2, 3)$ while all the other configurations belong to an unstable cycle pictured in bold. $\mathscr{G}(g)$ has a stable cycle pictured in bold. This cycle is stable because there is only one transition (corresponding to one arrow in the picture) leaving from each configuration, which is not the case for the unstable cycle of $\mathscr{G}(f)$.

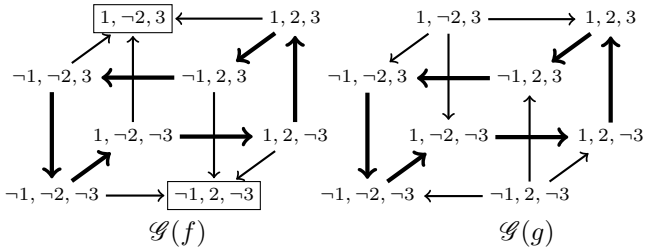


Figure 1: ATGs of functions f and g given in Example 1.

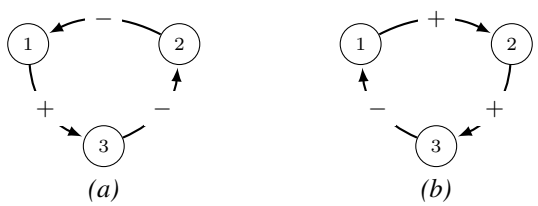


Figure 2: *(a)* IG associated with ATG $\mathscr{G}(f)$ and *(b)* IG associated with ATG $\mathscr{G}(g)$ of BDSs defined in Example 1.

2.2 Interaction graphs and circuits

A TG is an excellent tool for studying the behavior of a function. But in practice biological data come from experiments that generally yield correlations among gene expressions. This information is classically modeled by interaction graphs, exponentially more compact and more "readable" than TGs. Contrary to TGs, these graphs only give static information about how entities act on each other.

The IG of a BDS of function f is induced by its local transition functions f_i. An important line of research on BDSs concerns what we can say about the TG of a BDS by knowing only its static specification, that is its function, and thus its IG.

An IG is a signed digraph $G = (V, I)$, where $V = \{1, ..., n\}$ is the set of vertices and $I \subseteq V \times S \times V$, with $S = \{-, +\}$. An arc $(i, +, j)$ (resp. $(i, -, j)) \in I$ is said to be *positive* (resp. *negative*). A *circuit* $C = \{(i_1, s_{(1,2)}, i_2), ..., (i_k, s_{(k,1)}, i_1)\}$ of size k in terms of graph theory is *elementary* if all i_s that compose it are distinct. A circuit is *positive* (resp. *negative*) when it contains an even (resp. an odd) number of negative arcs. From the BDS point of view, the presence of an arc (i, s, j) in an IG G means that the value of i affects that of j: we say that i regulates j.

Consider the toy example where j has only one incoming arc, from i. In this case, the effect of the regulation is very simple: if the arc is positive (resp. negative), the state of j will take the value (resp. the opposite value) of that of i after one update, such that $f_j(x) = x_i$ (resp. $f_j(x) = \neg x_i$). Notice that elementary circuits are regulated this way. For example Figure 2 pictures the IGs associated with the ATGs of the BDSs defined from f and g in Example 1.

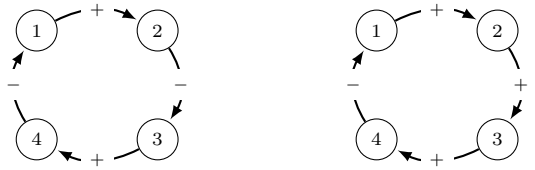
More generally, an IG $G = (V, I)$ represents the existence of the interactions involved between its entities in V. Specifying the nature of these interactions and the conditions under which they occur effectively leads to relate G to a BDS of function f, so that G becomes the IG of f and is then denoted by $G(f) = (V, I(f))$. This is done by assigning a local transition function f_i to every $i \in V$ so that $\forall j \in V, \exists x \in \{0, 1\}^n,\ f_i(x) \neq f_i(\overline{x}^j) \iff (j, s, i) \in I(f)$, where, given $x = (x_1, ..., x_n)$, $\overline{x}^j = (x_1, ..., x_{j-1}, \neg x_j, x_{j+1}, ..., x_n)$. We generalize this notation by $\overline{x} = (\neg x_1, ..., \neg x_n)$. Such a specification induces the minimality of $G(f)$ because each arc represents an effective interaction.

Note 1 Consider a BDS and its associated IG $G(f)$, such that arc (i, s, i) belongs to $I(f)$. If $s = +$ (resp $s = -$), this arc makes i tending to maintain (resp. negate) its state. It depends of course on whether i admits other in-neighbors than itself or not and on the positive or negative influence of these neighbors. In the case i admits no other in-neighbors, it is trivial that i endlessly maintains (resp. negate) its state if $s = +$ (resp. $s = -$).

Let us present now the asynchronous dynamical behaviors of a Boolean positive circuit and of a Boolean negative circuit of size 4 in Examples 2 and 3 below.

Example 2 (Boolean positive circuit of size 4) *Consider the BDS of function $f(x_1, x_2, x_3, x_4) = (\neg x_4, x_1, \neg x_2, x_3)$. Figure 3-a depicts the corresponding IG. This BDS admits two stable configurations, $(1, 2\neg 3, \neg 4)$ and $(\neg 1, \neg 2, 3, 4)$, and an unstable oscillation.*

Example 3 (Boolean negative circuit of size 4**)** *Consider the BDS of function* $g(x_1, x_2, x_3, x_4) = (\neg x_4, x_1, x_2, x_3)$. *Figure 3-b depicts the correponding IG. This BDS admits one stable cycle of length and one unstable cycle of length* 8



(a) *(b)*

Figure 3: *(a)* IG of Boolean positive circuit of size 4 (Example 2) and *(b)* negative circuit of size 4 (Example 3).

2.3 General central results

By considering that BDSs are good candidates for qualitatively modeling genetic networks (since established by the seminal papers of (Kauffman 1969) and (Thomas 1973)), the presence of several attractors in their dynamical behaviors allows to model the cellular specialization at the level of cells. Indeed, if a genetic network controls a phenomenon of specialization, the cell will specialize (i.e. will acquire a particular phenotype or a specific physiological function) according to the attractor towards which its underlying BDS evolves. These works and the numerous other ones using BDSs (or, more generally, DDSs) highlighted the essential role of studies aiming at understanding the formal relations between IG and TG. They also clearly underlined the essential role of circuits, nowadays known as the behavioral complexity engines in dynamical systems. This comes in particular from Robert who established that, if the IG $G(f)$ of a DDS f is acyclic, then f converges towards a unique stable configuration (Robert 1986). Moreover, in (Thomas 1981), Thomas conjectured that $G(f)$ of a asynchronous DDS f must contain a positive (resp. negative) circuit for the latter to admit several stable configurations (resp. a non-trivial attractor such as a stable cycle or a more complex one). These two conjectures have been proven to be true under the hypothesis of asynchronous updating mode (Remy et al. 2003; Remy, Ruet, and Thieffry 2008; Richard 2010; Richard and Comet 2007). Furthermore, notice that in (Remy et al. 2003), the authors showed that an asynchronous positive circuit of size n admits two attractors, namely two stable configurations x and $\overline{x}$, and that an asynchronous negative circuit admits only one attractor, namely a stable cycle of length $2n$.

Note 2 *While in the following we use positive and negative feedback circuits because of their central role in dynamic, our definitions and results hold for the full general framework. For instance the example proposed in Figure 4 is tractable inside our framework. Due to lack of space we do not treat here.*

3 Nonmonotonic, default, hypothesis logics

Representing IGs with a logical formalism seems natural because the way an arc (i, s, j) is interpreted suggests a close relation with what is called *material implication* in logic.

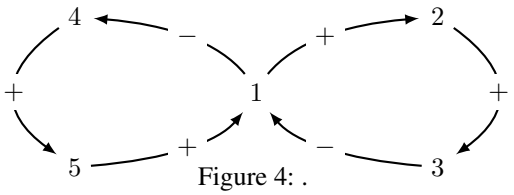


Figure 4: .

Such a representation from classical logic is not adapted because it leads to inconsistencies in most cases. A way to manage these inconsistencies is provided by nonmonotonic formalisms, among which default logic or ASPs (a more tractable restriction of default logic). *Default logic* (DL) (Reiter 1980) concerns standard formulas to which contextual inference rules called defaults are added in order to deal with revisable informations: a *default* is a local inference rule $d = \frac{A:B}{C}$, whose application specifically depends on the formulas A, B, C that compound it. The intuitive meaning is: "If A holds, if B is coherent/consistent with what is known, then C holds". The fact that a default can be triggered or not depending on the context, further leads to a notion of *extensions* as max-consistent sets of formulas with respect to the trigger of the defaults used to get it. The underlying reasoning is nonmonotonic because adding here a new information may invalidate previously triggered defaults.

There is clearly a connection between default extensions and stable configurations of a BDS (Doncescu, Siegel, and Le 2014; Doncescu and Siegel 2015). Also, recent works have studied the connection between ASP stable models and stable configurations. A drawback is that these connections are limited to the representation of stable configurations only (included odd circuits)while it tells nothing about unstable cycles. The problem arises from the way to capture all the dynamics.

Another drawback, actually linked to the first one, is that in DL, some theories may simply have no extensions at all, thus depicting a form of deep inconsistency which renders computation more difficult. By definition, DL only computes stable extensions. Such a type of extension is limited since it appears too cheap to handle with more than stability while we expect to capture also unstable cycles, e.g. the simple default theory ($d = \frac{:A}{\neg B}, \frac{:B}{\neg C}, \frac{:C}{\neg B}$), linked to the representation of a negative circuit, has no stable extension.

This drawback is overcome by justified DL (Lukaszewicz 1988) as well as by hypothesis logic. While justified DL exclusively consists in a reformulation of the conditions by which extensions are obtained, hypothesis logic is both a reformulation and a generalization of DL inside the framework of a normal bimodal logic. It actually generalizes justified (and hence classical) DL in the following way: while stable extensions are a special case of justified extensions, justified extensions are in turn a special case of the extensions obtained by hypothesis logic. Besides, the specific way hypothesis logic answers the question of the lack of extensions in DL sheds a new light on the representation of the dynamics in BDSs. In the context of a *theoretical* study concerning BDSs, we will find in Logic the formal criterias that help delimiting the "good" properties of a *langage* devoted to such a study. This is indeed the role of Logic to state rep-

resentation theorems between a langage and the objects this langage concerns. We expect hypothesis logic to be a right candidate as such a langage because it overtakes some of the limitations of DL (and hence ASP also).

3.1 Hypothesis Logic

Hypothesis logic $\mathcal{H}$ is a bi-modal logic[2] with two modal operators L and [H]. If f is a formula, the intuitive meaning of $\mathrm{L}f$ is *f is proved/stated*. The dual H of [H] is defined as $\mathrm{H}f = \neg[\mathrm{H}]\neg f$. It was introduced in (Schwind and Siegel 1994; Siegel and Schwind 1993). The intuitive meaning of $\mathrm{H}f$ is *f is a hypothesis*, and hence $[\mathrm{H}]f$ means *$\neg f$ is not a hypothesis*.

A default $\frac{A:B}{C}$ is interpreted in $\mathcal{H}$ by the modal formula $\mathrm{L}A \wedge \mathrm{H}B \rightarrow \mathrm{L}C$ whose intuitive meaning is: If A is stated and B is a valid hypothesis then C is stated. The language of $\mathcal{H}$, denoted by $\mathscr{L}(\mathcal{H})$, is defined by the following rules:

- Any formula of first-order logic is in $\mathscr{L}(\mathcal{H})$.
- If f and g are in $\mathscr{L}(\mathcal{H})$ then formulas $\neg f$, $(f \wedge g)$, $(f \vee g)$, $(f \rightarrow g)$, $\mathrm{L}f$, $[\mathrm{H}]f$, $\mathrm{H}f$ are too.

L has the properties of the modal system T and [H] has those of the modal system K. As a consequence, the inference rules and axiom schemata of $\mathcal{H}$ are:

- All inference rules and axiom schemata of first-order logic.
- (N[H]): $\vdash f \implies \vdash [\mathrm{H}]f$, the *necessitation rule* for [H].
- (NL): $\vdash f \implies \vdash \mathrm{L}f$, the *necessitation rule* for L.
- (K[H]): $\vdash [\mathrm{H}](f \rightarrow g)) \rightarrow ([\mathrm{H}]f \rightarrow [\mathrm{H}]g)$, the *distribution axiom schema* for [H].
- (K[L]): $\vdash \mathrm{L}(f \rightarrow g) \rightarrow (\mathrm{L}f \rightarrow \mathrm{L}g)$, the *distribution axiom schema* for L.
- (TL): $\vdash \mathrm{L}f \rightarrow f$, the *reflexivity axiom schema* for T.

Unlike L, [H] has no reflexivity axiom schema and, as it stands, there is so far no connections between L and [H]. We make this connection by adding the following *link axiom schema*:

$$(\mathrm{LI}) : \vdash \neg(\mathrm{L}f \wedge H\neg f).$$

This very weak axiom is one of the bases of $\mathcal{H}$. It means that it is impossible to prove f and to assume the hypothesis $\neg f$ at the same time. Note the following equivalences: $\neg(\mathrm{L}f \wedge \mathrm{H}\neg f) \iff \mathrm{L}f \rightarrow \neg\mathrm{H}\neg f \iff \mathrm{H}\neg f \rightarrow \neg\mathrm{L}f$. The first one means that *if we prove f, we cannot assume the hypothesis $\neg f$*, the second that *if we assume the hypothesis $\neg f$, we cannot prove f*.

3.2 Hypothesis theories and extensions

$\mathcal{H}$ is a monotonic logic. In order to catch nonmonotonicity, a notion of extension is added similarly to default logic. However, contrary to the latter, two kinds of extensions are considered here, namely stable extensions and ghost extensions. More formally :

[2]For a classical lecture on modal logics, see for instance (Chellas 1980), among others.

Definition 1 *Let $\mathcal{H}$ be the hypothesis logic:*

- *A* hypothesis theory *is a pair $\mathcal{T} = \{\mathrm{HY}, \mathrm{F}\}$, where* F *is a set of formulas of $\mathcal{H}$ and* HY *is a set of hypotheses.*
- *An* extension *E of $\mathcal{T}$ is a set $E = \mathrm{Th}(\mathrm{F} \cup \mathrm{HY}')$, such that* HY' *is a maximal subset of* HY *consistent with* F.
- *E is a* stable extension *if it satisfies the* coherence property*: $\forall \mathrm{H}f,\ \neg\mathrm{H}f \in E \implies \mathrm{L}\neg f \in E$. Hence, given the link axiom schema, we get: $\forall f, \mathrm{L}\neg f \in E \Leftrightarrow \neg\mathrm{H}f \in E$*
- *E is a* ghost extension *otherwise,* i.e. *if it satisfies: $\exists \mathrm{H}f,\ \neg\mathrm{H}f \in E$ and $\mathrm{L}\neg f \notin E$*

Hence for a ghost extension, we only get: $\forall f, \mathrm{L}\neg f \in E \Rightarrow \neg\mathrm{H}f \in E$

In other words, ghost extensions are "pre"–stable extensions.

Theorems 1 and 2 below give fundamental properties of $\mathcal{H}$. Their proof are given in (Siegel and Schwind 1993; Schwind and Siegel 1994).

Theorem 1 *If* F *is consistent then $\mathcal{T} = \{\mathrm{HY}, \mathrm{F}\}$ has at least one extension.*

Theorem 2 *Let $\Delta = \{D, W\}$ be an arbitrary default theory. Δ can be translated into a hypothesis theory $\mathcal{T}(\Delta)$ such that:*

1. W is consistent, $\mathcal{T}(\Delta)$ admits at least one extension;

2. The set of standard extensions of Δ is isomorphic to the set of stable extensions of $\mathcal{T}(\Delta)$.

Thus, an extension is obtained by adding one of the largest sets of hypotheses to F while remaining consistent. Note that if F is consistent, then there is always an extension, which is not the case in DL. Intuitively, E is stable if each time it is forbidden to assume the hypothesis f, $\neg f$ is proven. It is a ghost extension otherwise. Stable extensions correspond to the standard extensions of DL (and to stable models of ASP). They represent stable configurations of BDSs. Ghost extensions do not have any correspondence in DL. While default theories may have no extensions, this is not the case in hypothesis theories. Nevertheless, default theories that have no extensions seem to contain what looks very much like stable or unstable cycles. In the sequel, by using $\mathcal{H}$, we show that stable cycles are characterized by ghost extensions. Moreover, specific unstable cycles can also be characterized by ghost extensions, but the generalization of this result is a conjecture yet.

4 Representing Interaction Graphs into $\mathcal{H}$

We saw previously that a genetic network modeled by a BDS can be represented by both an ATG and an IG. As told, an important and open question concerns the formal links between these two representations. This section deals only with IGs. Let G be an arbitrary IG. Such a graph is translated into a hypothesis theory $\mathcal{T}(G)$ and properties related to extensions of $\mathcal{T}(G)$ are then proved. These properties will be used in the next section, devoted to ATG: G will be the IG related to an ATG, while both stable configurations and stable cycles of the ATG are studied thanks to $\mathcal{T}(G)$. Theorem 3 states that there exists an isomorphism between stable configurations of $\mathscr{G}$ and stable extensions of $\mathcal{T}(G)$ and Theorem 4 states that every negative feedback circuit corresponds to a set of equivalent ghost extensions.

4.1 Representation

One of the interests of hypothesis logic is that this bimodal logic enables us to use three kinds of information: i, $\mathrm{L}i$ and $\mathrm{H}i$. This increasing of expressive power allows a more precise representation of biological networks. Hence, by combining modalities with negations, we can use $\{i, \mathrm{H}i, \mathrm{H}\neg i, \mathrm{L}i, \mathrm{L}\neg i\}$.

Given an IG G of a BDS modeling the genetic network of a cell, and i a protein, using hypothesis logic, we define that:

- i means that *i is present in the cell and $\neg i$ means that it is absent*.
- $\mathrm{L}i$ means that *i is produced by the cell (i is being activated) and $\neg\mathrm{L}i$ means that i is not produced (i is not being activated)*.
- $\mathrm{L}\neg i$ means that *i is destroyed by the cell (i is being inhibited) and $\neg\mathrm{L}\neg i$ means that i is not destroyed (i is not being inhibited)*.
- $\mathrm{H}i$ (resp. $\neg\mathrm{H}i$) means that *the cell gives (resp. does not give) the permission for attempting to produce i. In other words, the cell has (resp. has not) the ability to activate i*.
- $\mathrm{H}\neg i$ (resp. $\neg\mathrm{H}\neg i$) means that *the cell gives (resp. does not give) the permission for attempting to destroy i. In other words, the cell has (resp. has not) the ability to inhibit i*.

Regarding the use of $\mathcal{H}$ in this context, the role of an extension appears to gather a maximum of consistent permissions. Note that even if $\mathrm{H}i$ stands for the cell giving permission to attempt the production of i, this production is not mandatory. It can be carried out or not, according to the context (i.e. the set of all interactions in the cell). Similarly $\mathrm{H}\neg i$ gives the authorization to destroy i.

Meanwhile, it is important to note that $\mathrm{L}i$ and $\mathrm{L}\neg i$ are actually actions (production or destruction of a protein). So there is a difference between $\mathrm{L}\neg i$ which says that i is destroyed, and $\neg\mathrm{L}i$ which says that i is not produced, and hence is weaker. Likewise, there is a similar distinction between $\mathrm{H}\neg i$ and $\neg\mathrm{H}i$.

We first focus on some important properties of our translation with respect to genetic networks[3].

Proposition 1 *If G is the IG of a BDS modeling a genetic network and if i is a protein, the following holds in $\mathcal{H}$:*

(1) $\mathrm{L}i \rightarrow i$ *and* $\mathrm{L}\neg i \rightarrow \neg i$ *(i.e. if i is produced (resp. destroyed), then i is present (resp. absent).)*

(2) $\neg(\mathrm{L}i \wedge \mathrm{H}\neg i)$ *and* $\neg(\mathrm{L}\neg i \wedge \mathrm{H}i)$ *(i.e. it is impossible to produce (resp. destroy) i and to give the permission to destroy (resp. produce) i it at the same time.)*

(3) $\neg(\mathrm{L}i \wedge L\neg i)$ *(i.e. it is impossible to produce and destroy i at the same time.)*

4.2 Translation of an interaction graph.

An IG $G = (V, I)$ is translated into a hypothesis theory $\mathcal{T}(G) = \{\mathrm{HY}(G), \mathrm{F}(G)\}$ so that every arc $(i, s, j) \in I$ is translated into a pair of implications of $\mathcal{H}$. More precisely:

- A positive arc $(i, +, j)$ is translated into: $\{\mathrm{H}i \rightarrow \mathrm{L}j, \mathrm{H}\neg i \rightarrow \mathrm{L}\neg j\}$.
- A negative arc $(i, -, j)$ is translated into: $\{\mathrm{H}i \rightarrow \mathrm{L}\neg j, \mathrm{H}\neg i \rightarrow \mathrm{L}j\}$.
- $\mathrm{F}(G)$ is the union of the translations of all elements of I.
- $\mathrm{HY}(G)$ is the set of all $\mathrm{H}i$ and $\mathrm{H}\neg i$ appearing in $\mathrm{F}(G)$.

Note 3 *This translation only uses implications between two atomic formulas. These implications could be considered as binary clauses. Therefore only a fragment of the plain formalism $\mathcal{H}$ is used in this paper, which is enough for the description of "conventional" BDSs. Note that $\mathcal{H}$ formulas may contain all the logical connectors $(\wedge, \vee, \neg, \rightarrow, \leftrightarrow$ ...$)$ hence full $\mathcal{H}$ can be used to describe other properties of biological networks, e.g. the binding (two proteins bind to give a new protein). It is also possible to assert the proposition i alone or $\mathrm{L}i$ alone or $\mathrm{H}i$ alone. We can even avoid the double implication $\{Hx \rightarrow Ly, H\neg x \rightarrow L\neg y\}$ given for the translation of a BDS: for some functions only one involvement of the two can be enough. This increases the expressive power of the formalism, which in turn should increase the algorithmic complexity, but $\mathcal{H}$ is still usable however.*

The following definitions and propositions, are needed for understanding the intuition behind this representation of IGs by $\mathcal{H}$. They will especially allow us to state properties and theorems 3 and 4 which make links between IGs and ATGs. In $\mathcal{H}$, it is usually allowed to have both $\mathrm{H}i$ and $\mathrm{H}\neg i$. Regarding the fragment of $\mathcal{H}$ used here, Proposition 2 below shows that this no longer holds, because of the double logical implication obtained from our translation of an arc:

Proposition 2 *Let $G = (V, I)$ an IG and $i \in V$. For every $\mathrm{H}i$ of $\mathcal{T}(G)$, $\neg(\mathrm{H}i \wedge \mathrm{H}\neg i)$ holds.*

Definition 2 *Let $G = (V, I)$ an IG such that $V = \{1, .., n\}$. Let $\mathcal{T}(G) = \{\mathrm{HY}(G), \mathrm{F}(G)\}$ the translation of G into a hypothesis theory. Let $E = \{\mathrm{Th}(\mathrm{F}(G)) \cup \{\mathrm{H}y_k\}\}$ be an extension of $\mathcal{T}(G)$ obtained by adding to $\mathrm{F}(G)$ a maximal consistent set $\{\mathrm{H}y_k\}$ of hypotheses, with $y_k \in \{1, ..., n, \neg 1, ..., \neg n\}$. (For lightening the reading, we simply write that E is an extension of G). We have:*

1. E is complete *if, for all $i \in V$, $\mathrm{H}i \in E$ or $\mathrm{H}\neg i \in E$.*

2. A vertex $i \in V$ is free *in E if $\mathrm{L}i \notin E$ and $\mathrm{L}\neg i \notin E$. It is* fixed *otherwise.*

3. The degree of freedom of E, denoted $\deg(E)$, is the number of free vertices that compose it.

4. The mirror *of E, denoted $\mathrm{mir}(E)$, is defined as $\mathrm{mir}(E) = \mathrm{Th}(\mathrm{F}(G) \cup \{\mathrm{H}\neg y_k\})$.*

5. The generating set *of E, denoted $\mathrm{Gen}(E)$ is the set of formulas $(\mathrm{H}y \rightarrow \mathrm{L}z) \in \mathrm{F}(G)$ such that $\mathrm{H}y \in \{Hy_k\}$.*

6. The graph of E*, denoted by $G(E)$, is the unsigned digraph of vertices $\{y_1, ..., y_m\}$ such that:*

$$(y_i, y_j) \in G(E) \iff (\mathrm{H}y_i \rightarrow \mathrm{L}y_j) \in \mathrm{Gen}(E).$$

Proposition 3 *Let G be an IG and E one of its extensions. The mirror of E is also an extension of G.*

Intuition might suggest that the notions of stable extension, complete extension, and extension of degree 0 are equivalent. In fact this is wrong in the general case and it is only possible to prove that a stable extension is complete.

[3]Note to the rewievers: all the proofs of the original propositions and theorems given in the paper can be found at the url `https://amubox.univ-amu.fr/index.php/s/nhwsZ5eqV8BYVv1`

But we prove that, when any vertex of the IG G has an incoming arc, if E is complete then E is both stable and of degree 0; this is especially the case for circuits.

Proposition 4 *Let $G = (V, I)$ be an IG of a given BDS f and let E be an extension of G. The following holds:*

1. If E is stable, then E is complete.

Moreover, if each of the vertices of G has at least one incoming arc:

2. If E is complete, then $\deg(E) = 0$.

3. If E is complete, then E is stable.

4. If E is stable then $\deg(E) = 0$.

Proposition 5 *Let $E = \{\mathrm{Th}(\mathrm{F}(G)) \cup \{\mathrm{H}y_k\}\}$ be an extension and consider* $\mathrm{Gen}(E)$ *its generating set:*

1. $E = \{\mathrm{Th}(\mathrm{Gen}(E)) \cup \{\mathrm{H}y_k\}\}$.

2. If $\mathrm{H}x \in \{\mathrm{H}y_k\}$ then $\mathrm{Gen}(E)$ cannot contain both $\mathrm{H}x \to \mathrm{L}y$ and $\mathrm{H}\neg x \to \mathrm{L}\neg y$ at the same time.

3. If $\deg(E) = 1$ *then an order can be chosen among the y_k with a circular permutation such that:* $\mathrm{Gen}(E) = \{\mathrm{H}i \to \mathrm{L}(i+1), \mathrm{H}(i+1) \to \mathrm{L}(i+2), ...\mathrm{H}(i-2) \to \mathrm{L}(i-1)\}$.

Proposition 6 *Given $G = (V, I)$, with V = {1, ..., n}, a negative circuit of size n, then:*

1. $\mathcal{T}(G)$ has no extensions of degree 0.

2. $\mathcal{T}(G)$ has $2n$ extensions of degree 1. *We will say that these 2n extensions are* equivalent.

The examples below serve as an illustration of the notions introduced here.

Example 2 (continued) Consider the BDS of the function $f(x_1, x_2, x_3, x_4) = (\neg x_4, x_1, \neg x_2, x_3)$ studied in Example 2 and Figure 3. This BDS corresponds to a positive circuit. Its admits two stable configurations, $(x_1, x_2, \neg x_3, \neg x_4)$ and $(\neg x_1, \neg x_2, x_3, x_4)$ and an unstable oscillation. Let $G(f)$ the IG of f, depicted in Figure 3-b. By construction $G(f)$ is a set of four arcs:

• $G(f) = \{(1, +, 2), (2, -, 3), (3, +, 4), (4, -, 1)\}$

The positive arc $(1, +, 2)$ is translated into $\mathcal{H}$ by the pair of formulas $\{\mathrm{H1} \to \mathrm{L2}, \mathrm{H}\neg 1 \to \mathrm{L}\neg 2\}$ and the negative arc $(2, -, 3)$ is translated by the pair $\{\mathrm{H2} \to \mathrm{L}\neg 3, \mathrm{H}\neg 2 \to \mathrm{L3}\}$. The other arcs are translated in the same way, therefore the translation of $G(f)$ into a hypothesis theory is $\mathcal{T}(G(f)) = \{\mathrm{HY}(G(f)), \mathrm{F}(G(f))\}$, where:

• $\mathrm{HY}(G(f)) = \{\mathrm{H1}, \mathrm{H2}, \mathrm{H3}, \mathrm{H4}, \mathrm{H}\neg 1, \mathrm{H}\neg 2, \mathrm{H}\neg 1, \mathrm{H}\neg 4\}$

• $\mathrm{F}(G(f)) = \{\mathrm{H1} \to \mathrm{L2}, \mathrm{H}\neg 1 \to \mathrm{L}\neg 2, \mathrm{H2} \to \mathrm{L}\neg 3,$
$\mathrm{H}\neg 2 \to \mathrm{L3}, \mathrm{H3} \to \mathrm{L4}, \mathrm{H}\neg 3 \to \mathrm{L}\neg 4, \mathrm{H4} \to \mathrm{L}\neg 1, \mathrm{H}\neg 4 \to \mathrm{L1}\}$

We can show that $\mathcal{T}(G(f))$ has two stable extensions: $E1$ obtained by adding to $F(G(f))$ the set of hypothesis $\{\mathrm{H1}, \mathrm{H2}, \mathrm{H}\neg 3, \mathrm{H}\neg 4\}$ and $E2$ obtained by adding to $F(G(f))$ the set of hypotheses $\{\mathrm{H}\neg 1, \mathrm{H}\neg 2, \mathrm{H3}, \mathrm{H4}\}$[4]. These extensions correspond to the two stables configurations of the related BDS.

• $E1 = \mathrm{Th}(\mathrm{F}(G(f)) \cup \{\mathrm{H1}, \mathrm{H2}, \mathrm{H}\neg 3, \mathrm{H}\neg 4\})$,

• $E2 = \mathrm{Th}(\mathrm{F}(G(f)) \cup \{\mathrm{H}\neg 1, \mathrm{H}\neg 2, \mathrm{H3}, \mathrm{H4}\})$.

In the sequel, given i a proposition of propositional calculus, we will consider that i, Hi,Li, $\neg i$, $\neg Hi$ et $\neg Li$ are literals of $\mathcal{H}$, and that $\mathrm{H}i \to \mathrm{L}j$ is a clause. From the deductive closure of $E1$ with subsumption, we obtain that these extensions are logically equivalent to sets of literals:

• $E1 = \mathrm{Th}\{\mathrm{H1}, \mathrm{H2}, \mathrm{H}\neg 3, \mathrm{H}\neg 4, \neg\mathrm{H}\neg 1, \neg\mathrm{H}\neg 2, \neg\mathrm{H3}, \neg\mathrm{H4}, \mathrm{L1}, \mathrm{L2}, \mathrm{L}\neg 3, \mathrm{L}\neg 4, \neg\mathrm{L}\neg 1, \neg\mathrm{L}\neg 2, \neg\mathrm{L3}, \neg\mathrm{L4}, 1, 2, \neg 3, \neg 4\}$

• $E2 = \mathrm{Th}\{\mathrm{H}\neg 1, \mathrm{H}\neg 2, \mathrm{H3}, \mathrm{H4}, \neg\mathrm{H1}, \neg\mathrm{H2}, \neg\mathrm{H}\neg 3, \neg\mathrm{H}\neg 4, \mathrm{L}\neg 1, \mathrm{L}\neg 2, \mathrm{L3}, \mathrm{L4}, \neg\mathrm{L1}, \neg\mathrm{L2}, \neg\mathrm{L}\neg 3, \neg\mathrm{L}\neg 4, \neg 1, \neg 2, 3, 4\}$

For the sake of simplicity, let us assimilate any extension $\mathrm{Th}(\mathrm{F}(G(f)) \cup \{\mathrm{H1}, ..., \mathrm{H}n\})$ with the set of hypotheses $\{\mathrm{H1}, ..., \mathrm{H}n\}$ associated with it. Looking at $E1$ we notice that, in accordance with Definitions 1, and 2, we have:

• $E1$ and $E2$ are stable extensions because for all i, $\neg\mathrm{H}i \in E1$ (resp $E2$) $\Rightarrow L\neg i \in E1$ (resp $E2$).

• $E1$ is complete because for all i, either Hi belongs to $E1$ or $H\neg i$ belongs to $E1$.

• For all i, $Li \in E1$ or $L\neg i \in E1$. So all vertices are fixed, and the degree of freedom of $E1$ is 0.

• $E2$ is the mirror of $E1$.

• The generating set of $E1$, is $Gen(E1) = \mathrm{H1} \to \mathrm{L2}, \mathrm{H2} \to \mathrm{L}\neg 3, \mathrm{H}\neg 3 \to \mathrm{L}\neg 4, \mathrm{H}\neg 4 \to \mathrm{L1}\}$.

• The graph of $E1$, depicted in Figure 5-c, is: $G(E) = (1, 2), (2, \neg 3), (\neg 3, \neg 4), (\neg 4, 1)$.

From the biological side, the subset $\{\mathrm{L1}, \mathrm{L2}, \mathrm{L}\neg 3, \mathrm{L}\neg 4\}$ of $E1$ represents the expression pattern of each protein in $E1$: 1 and 2 are produced by the cell and 3 and 4 are destroyed. Moreover the subset $\{1, 2, \neg 3, \neg 4\}$ of $E1$ (in fact the vertices of $G(E)$) gives the status of each proteins: 1 and 2 are present in the cell and $\neg 3$ and $\neg 4$ are absent. Similarly in $E2$, the subsets $\{\neg\mathrm{L1}, \neg\mathrm{L2}, \mathrm{L3}, \mathrm{L4}\}$ and $\{\neg 1, \neg 2, 3, 4\}$ represent the expression pattern and the status of proteins.

An intuition of the computation of $E1$ is given by the construction process described by Figure 5. Figure 5-a is the IG of f. Figure 5-b gives the construction of $E1$. At first $E1$ is empty. We add to $E1$ the hypothesis $\mathrm{H1} \in HY(G(f))$. Since $(\mathrm{H1} \to \mathrm{L2}) \in F(G(f))$ we get L2. The instance $(\mathrm{L2} \to \neg H\neg 2)$ of the axiom LI of $\mathcal{H}$ then tells that it is impossible to have $H\neg 2$. Construction of $E1$ goes on by adding the hypothesis H2 to $E1$. We get $\mathrm{L}\neg 3$ with $\mathrm{H2} \to \mathrm{L}\neg 3 \in F(G(f)$. The axiom LI gives $\neg\mathrm{H3}$ and we add the hypotheses $\mathrm{H}\neg 3$ which gives $\mathrm{L}\neg 4$. We end up by adding $\mathrm{H}\neg 4$ which gives L1.

By only looking at i we have $G(E1)$, the graph of $E1$ (Figure 5-c) which represents the first stable configuration of f. In this final graph, each arc between 2 vertices denotes the relation of causality that links the corresponding proteins: for 1 to be present we need 4 to be absent, that is 3 to be absent in its turn, which is involved by the presence of 2, itself caused by the presence of 1 and so on... In a similar way, we build the extension $E2$ starting from $H\neg 1$.

Example 3 (continued) Consider the BDS of the function $g(x_1, x_2, x_3, x_4) = (\neg x_4, x_1, x_2, x_3)$ studied in Example 3. It admits one attractor (a stable cycle of length 8). It admits also an unstable cycle of length 8. Let be $G(g)$ the IG of g. By construction it is a set of four arcs:

• $G(g) = \{(1, +, 2), (2, +, 3), (3, +, 4), (4, -, 1)\}$

Following the same guidelines as above, the translation of $G(g)$ into a hypothesis theory is $\mathcal{T}(G(g))$ gives us here eight ghost extensions. The first one is:

[4]This is shown by attempting to add to $\mathrm{F}(G(f))$ each subset of $\mathrm{HY}(G(f))$ and keeping only those which are the maximals ones consistent with $\mathrm{F}(G(f))$.

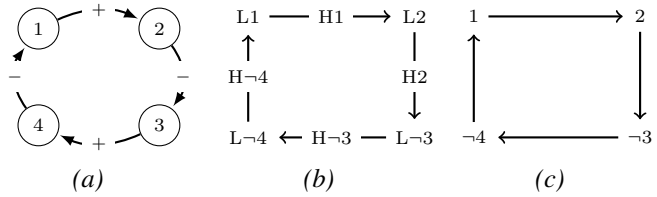


Figure 5: *(a)* IG $G(f)$, *(b)* Construction of extension $E1$ and *(c)* $G(E1)$ (the graph of $E1$) of the positive circuit defined in Example 2.

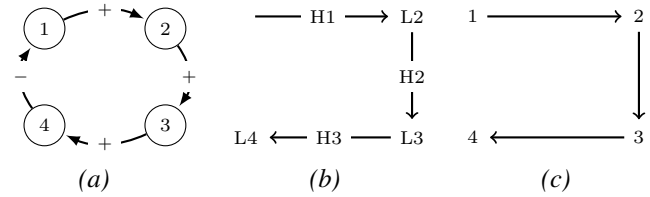


Figure 6: *(a)* Interaction graph $G(g)$, *(b)* Construction of extension $E1$ and *(c)* $G(E1)$ (graph of $E1$) of the negative circuit defined in Example 3 (continued).

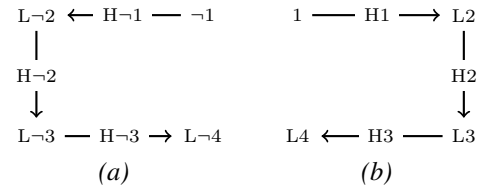


Figure 7: *(a)* $E1$ and *(b)* his mirror $E2$ of the negative circuit defined in Example 3 (continued)

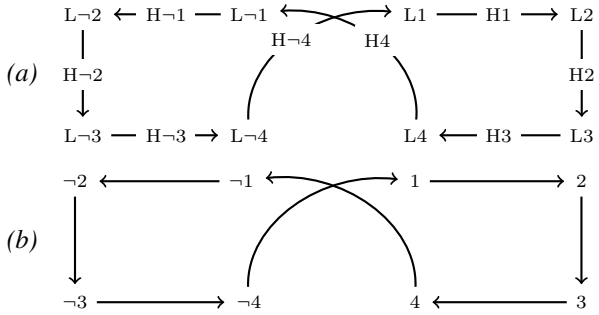


Figure 8: *(a)* Construction of $E1 \rightleftharpoons E2$ and *(b)* its graph $G(E1 \rightleftharpoons E2)$ of the negative circuit defined in Example 3

- $E1 = \mathrm{Th}(\mathrm{F}(G(g)) \cup \{\mathrm{H1}, \mathrm{H2}, \mathrm{H3}\})$
  $= \mathrm{Th}(\{\mathrm{H1}, \mathrm{H2}, \mathrm{H3}, \neg\mathrm{H}\neg1, \neg\mathrm{H}\neg2, \neg\mathrm{H}\neg3, \neg\mathrm{H}\neg4,$
  $\neg\mathrm{H4}, \mathrm{L2}, \mathrm{L3}, \mathrm{L4}, \neg\mathrm{L}\neg1, \neg\mathrm{L}\neg2, \neg\mathrm{L3}, \neg\mathrm{L}\neg4, 2, 3, 4\})$.

The generator set of $E1$ is $\{\mathrm{H1}\rightarrow\mathrm{L2}, \mathrm{H2}\rightarrow\mathrm{L3}, \mathrm{H3}\rightarrow\mathrm{L4}\}$. The literal $\neg\mathrm{H}\neg1 \in E1$ characterizes $E1$ as a ghost extension because $L1 \notin E1$. The extension is not complete because it contains neither H4 nor H$\neg$4. Vertex 1 is free because $E1$ contains neither L1 nor L$\neg$1. This is the only free vertex, hence $E1$ is a ghost extension of degree 1. For this extension, vertices 2, 3 and 4 are fixed while nothing is known about 1. The drawing of $E1$ (Figure 6-b) shows what happens. Similary to Example 2, $E1$ is constructed by adding {H1, H2, H3} to $\mathrm{F}(G(g))$. The set $\{\mathrm{L2}, \mathrm{L3}, \mathrm{L4}\}$ is obtained, which yields $\{2, 3, 4\}$ from axiom (T). Hence it is impossible to add H$\neg$4 because $(\mathrm{H}\neg4 \rightarrow \neg\mathrm{L4})$ from the axiom of coherence and $\mathrm{L4} \in E1$. Also, one can not add H4 because $(\mathrm{H4} \rightarrow \mathrm{L}\neg1) \in \mathrm{F}(G(g))$ and $\mathrm{L1} \in E1$, which will imply 1 and $\neg1$ with axiom (T). In Figure 6-b, the place for 1 is empty because the extension cannot contain both L1 and L$\neg$1, and 1 is free. Indeed, in order to get L1, we should use H$\neg$4 $\rightarrow$ L1, which is impossible because L4 is true and, because from the axiom of coherence, L4 $\rightarrow$ $\neg$H$\neg$4. We cannot have L$\neg$1 because H1 is true and H1 $\rightarrow$ $\neg$L$\neg$1. The graph of $E1$, $G(E1)$ is depicted in Figure 6-c.

It is important to see that the notion of degree of freedom plays a key role here. Noting that Figure 6-b (resp. Figure 6-c) are not circuits because there is no arc between $L4$ (resp. 4) and the place of 1. From $L4$ we can then *escape* the incomplete circuit. Let us indeed show how to do it. Since $E1$ is an extension, $E2$ the mirror of $E1$ is also an extension from Proposition 3 (see Figure 7). Taking into account that 1 is free in $E1$ and also in $E2$, $E1$ and in $E2$ can be connected ($E1 \rightleftharpoons E2$) by binding L4 to L$\neg$1 using H$\neg$4 on one hand, and L$\neg$4 to L1 using H4 on the other hand (Figure 8-a). By linking $G(E1)$ and $G(E2)$ (Figure 8-b), we then obtain a cycle $(1, 2, 3, 4, \neg1, \neg2, \neg3, \neg4)$ of length 8 such as depicted. This cycle represents the expression pattern of each protein over time: we turn two times in the IG until returning to the initial state.

In this example, there are actually eight equivalent extensions $E1 = \{\mathrm{L2}, \mathrm{L3}, \mathrm{L4}\}$, its miror $E2 = \{\mathrm{L}\neg2, \mathrm{L}\neg3, \mathrm{L}\neg4\}$ and 6 other extensions which come from permutations on i, namely:

$\{\mathrm{L3}, \mathrm{L4}, \mathrm{L}\neg1\}, \{\mathrm{L4}, \mathrm{L}\neg1, \mathrm{L}\neg2\}, \{\mathrm{L}\neg1, \mathrm{L}\neg2, \mathrm{L}\neg3\},$
$\{\mathrm{L}\neg2, \mathrm{L}\neg3, \mathrm{L}\neg4\}, \{\mathrm{L}\neg3, \mathrm{L}\neg4, \mathrm{L1}\}, \{\mathrm{L}\neg4, \mathrm{L1}, \mathrm{L2}\}$.

Note that each of these extensions corresponds to 3 successive entities of the cycle above. The latter cycles contains implicitly each of the 8 possible extensions. There is a trick here since we actually used an update that is not asynchronous, but this trick, aiming at authorizing synchronous changes appears to be virtuous. It will be seen that these 8 extensions correspond to the stable cycle of the ATG $\mathscr{G}(g)$.

5 Representing Asynchronous Transition Graphs into $\mathcal{H}$

Consider an asynchronous BDS, whose its IG is $G = (V, I)$ and its ATG is $\mathscr{G}$. Let $\mathcal{T}(G) = \{\mathrm{HY}(G), \mathrm{F}(G)\}$ be the hypothesis theory associated with G. Up to now, we have studied the representation of IGs into hypothesis logic.

This section studies the relationship between this representation and the ATG. It uses *Kripke semantics* (Kripke 1963) that has been defined for normal modal logics, *i.e.*, the logics that contain at least axiom (K). We only provide here the bases needed for our developments. A *Kripke structure* is a digraph $K = (W, R)$ where W (the universe) is a set $\{w_k\}$ of worlds and $R \subseteq W \times W$ is a binary relation among worlds: the *accessibility relation*. When $w_k \, R \, w'_k$, w'_k is *accessible* from w_k. A *Kripke model* is obtained by

assigning in every world a truth value to every proposition i. A world is then mapped to a logical interpretation, and hence implicitely to a state of a BDS. Modal formulas other than i are assigned to worlds with the following condition: for all f, Lf is true in a world w_k if and only if f is true in all reachable worlds from w. The different axioms that hold in different modal logics depend on the properties of the accessibility relations R. It is known that:

1. For the modal system K, R is any relation.
2. Axiom (T) holds if and only if R is reflexive.

Example 4 *Consider the Kripke structure K such that $K = (W, R)$, where $W = \{w_1, w_2, w_3\}$ and $R = \{(w_1, w_2), (w_1, w_1), (w_2, w_2), (w_3, w_3)\}$.*

Consider now that a truth value is assigned to three variables in each world of the universe W such that: $w_1 = \{1, 2, 3\}$, $w_2 = \{\neg 1, 2, 3\}$, $w_3 = \{\neg 1, \neg 2, 3\}$. Let us add modal formulas with respect to the definition of Krypke semantic. For example in w_1, L2 *is true because* 2 *is in both w_1 and w_2 (the worlds that are reachable from w_1). For the sake of clarity, we do not put in this schema the negations of modal formulas. Now, consider $\mathcal{H}$. If w_1 was an extension, then it would be a ghost extension of degree* 1*, because* L2 *and* L3 *are true while neither* L1 *nor* L$\neg$1 *are. Similarly, for both worlds w_2 and w_3, if they were extensions, they would be stable extensions of degree* 0 *because for all $i \in \{1, 2, 3\}$, either* Li *or* L$\neg i$ *is true; they would then correspond to stable configurations of a related BDS because they do not admit any outward arcs.*

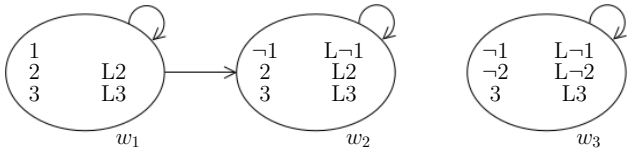


Figure 9: Kripke universe discussed in Example 4.

We use the Kripke structure $K = (W, R)$ whose finite set of worlds W is the set of all interpretations (called *canonical universe*) such that $w_k \, R \, w'_k$ if and only if w'_k is reachable from w_k and differs from w_k by one and only one proposition. Under these conditions, the ATG of any BDS is a Kripke structure. Since the modal system T is reflexive, loops appear on every world. Given such a framework, for any world w_k and any entity x, Lx = true if and only if $x =$ true in every w'_k reachable from w_k.

Example 1-continued Consider the previous function $g(x_1, x_2, x_3) = (\neg x_3, x_1, x_2)$ of Example 1. Similarly to Example 3-continued, constructing the related hypothesis theory leads the following set of formulas:

$$\mathrm{F}(G(g)) = \{\mathrm{H1} \rightarrow \mathrm{L2}, \mathrm{H2} \rightarrow \mathrm{L3}, \mathrm{H3} \rightarrow \mathrm{L}\neg 1, \mathrm{H}\neg 1 \rightarrow \mathrm{L}\neg 2, \mathrm{H}\neg 2 \rightarrow \mathrm{L}\neg 3, \mathrm{H}\neg 3 \rightarrow \mathrm{L1}\},$$

which allows to obtain the following 6 equivalent extensions, by focusing only on the Li and L$\neg i$ which are true:

$E1 = (\mathrm{L2}, \mathrm{L3}), E2 = (\mathrm{L}\neg 1, \mathrm{L3}), E3 = (\mathrm{L}\neg 1, \mathrm{L}\neg 2),$
$E4 = (\mathrm{L}\neg 2, \mathrm{L}\neg 3), E5 = (\mathrm{L1}, \mathrm{L}\neg 3), E6 = (\mathrm{L1}, \mathrm{L2}).$

Figure 9 is a simplified representation of the Krypke model associated with the theory $F(G(g))$. The 8 vertices are the worlds, and the arrows express the accessibility relation. There is a loop on each vertex because (T) is reflexive. The 6 extensions $E1, .., E6$ are represented by the 6 vertices whose degree of freedom is minimal (that is of degree 1). The other two vertices $\{1, \neg 2, 3\}$ and $\{\neg 1, 2, \neg 3\}$ are not extensions because they are not maximal. Note that these 6 extensions correspond to the 6 configurations of the stable cycle of the corresponding ATG of function g given in example 1 (Figure 11). Note also that the set of arrows that represent the accessibility relation contains the set of arrows that represent the ATG transitions of g. The missing arrows are the loops that represent the reflexivity. By taking in account that the degree of a world is a natural generalization of degree of an extension, we get the following results.

Proposition 7 *Let $K = (W, R)$ be a Kripke structure associated with an IG and its underlying hypothesis theory. If $w_k \in W$ is a world of degree δ, there are exactly δ worlds, different from w_k, reachable from w_k*

Theorem 3 *Let $\mathcal{S}$ be an asynchronous BDS, G and $\mathcal{G}$ the corresponding IG and ATGs and $\mathcal{T}(G)$ the hypothesis theory related to G.*

1. If E is a stable extension of $\mathcal{T}(G)$ and if $\{\mathrm{L}y_1, ..., \mathrm{L}y_n\}$ is the set of all Li *and* L$\neg i$ *that are true in E, and if the x_i are the Boolean values of the y_i, then $\{x_1, ..., x_n\}$ is a stable configuration of $\mathcal{S}$.*

2. $\{x_1, ..., x_n\}$ is a stable configuration of $\mathcal{T}(G)$, then there exists a stable extension E of $\mathcal{G}$ that contains $\{\mathrm{L}y_1, ..., \mathrm{L}y_n\}$, such that $y_i = i$ (resp $y_i = \neg i$) il $x_i = 1$, (resp. $x_i = 0$).

Theorem 4 *Let $\mathcal{S}$ be an asynchronous BDS of ATG $\mathcal{G}$, whose IG G is a negative circuit. Let $E1$ be a ghost extension of $\mathcal{T}(G)$. The set of all extensions equivalent to $E1$ corresponds to a stable cycle of $\mathcal{G}$.*

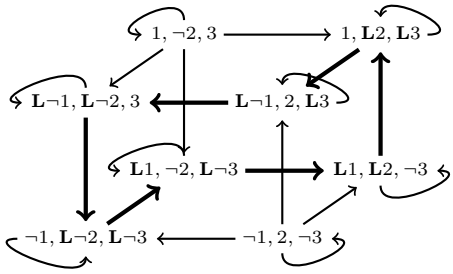


Figure 10: Krypke representation of F($G(g)$).

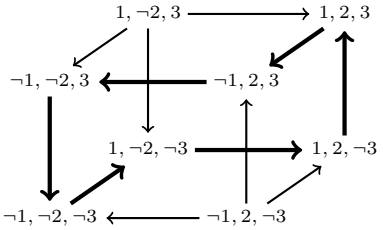


Figure 11: ATG of function g given Example 1.

Note 4 With same arguments as those used for the proof of proposition 7, if $deg(E)>1$ the Kripke model gives at least one possibility to exit from a cycle. The cycle is then unstable.

6 Conclusion

This paper is an attempt for representing BDSs into Hypothesis Logics; the difficulty is to find how to represent the dynamics. There is still much to study, especially regarding a generalization of Theorem 4 to unstable cycles. Note 4 above is a hint for such a study. Another perspective is the validation of BDS representation in $\mathcal{H}$ by the obtaining of fundamental theorems. The fact that the logical representation of a positive circuit has two stable mirror extensions and, that a negative circuit is equivalent to a single set of $2n$ equivalents ghost extensions is a step towards this validation because it corresponds to the esults set in (Remy et al. 2003). The case of synchronous transitions remains under consideration for a further study. Note that an extension is obtained by adding a consistent maximal set of hypotheses. Since it is possible to test whether consistency is preserved when adding each hypothesis, the computation of extensions is non-deterministic and constructive (which is not the case for DL and ASP).

References

Akman, O. E.; Watterson, S.; Parton, A.; Binns, N.; Millar, A. J.; and Ghazal, P. 2012. Digital clocks: simple Boolean models can quantitatively describe circadian systems. *Journal of The Royal Society Interface*.

Aracena, J.; González, M.; Zuñiga, A.; Mendez, M. A.; and Cambiazo, V. 2006. Regulatory network for cell shape changes during Drosophila ventral furrow formation. *Journal of Theoretical Biology* 239:49–62.

Chellas, B. 1980. *Modal logic, an introduction*. Cambridge University Press.

Davidich, M. I., and Bornholdt, S. 2008. Boolean network model predicts cell cycle sequence of fission yeast. *PLoS One* 3:e1672.

Demongeot, J.; Goles, E.; Morvan, M.; Noual, M.; and Sené, S. 2010. Attraction basins as gauges of the robustness against boundary conditions in biological complex systems. *PLoS One* 5:e11793 (18 pages).

Demongeot, J.; Elena, A.; Noual, M.; Sené, S.; and Thuderoz, F. 2011. "immunetworks", intersecting circuits and dynamics. *Journal of Theoretical Biology* 280:19–33.

Demongeot, J.; Noual, M.; and Sené, S. 2012. Combinatorics of Boolean automata circuits dynamics. *Discrete Applied Mathematics* 160:398–415.

Doncescu, A., and Siegel, P. 2015. *Emerging Trends in Computational Biology, Bioinformatics, and Systems Biology*. Elsevier. chapter DNA double-strand break-based nonmonotonic logic, 409–427.

Doncescu, A.; Siegel, P.; and Le, T. 2014. Representation and efficient algorithms for the study of cell signaling pathways. In *Proceedings of ICAI'2014*, 504–510. IEEE Computer Society.

Fauré, A.; Naldi, A.; Chaouyia, C.; and Thieffry, D. 2006. Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics* 22:e124–e131.

Jacob, F., and Monod, J. 1978. Genetic regulation mechanisms in the synthesis of proteins. In *Selected Papers in Molecular Biology by J. Monod*, 433–471. Academic Press.

Kauffman, S. A.; Peterson, C.; Samuelsson, B.; and Troein, C. 2003. Random Boolean network models and the yeast transcriptional network. *PNAS* 100:14796–14799.

Kauffman, S. A. 1969. Metabolic stability and epigenesis in randomy constructed nets. *Journal of Theoretical Biology* 22:437–467.

Kripke, S. A. 1963. Semantical analysis of modal logic I Normal modal propositional calculi. *Mathematical Logic Quarterly* 9:67–96.

Li, F.; Long, T.; Lu, Y.; Ouyang, Q.; and Tang, C. 2004. . *PNAS* 101:4781–4786.

Lifschitz, V. 1999. *The Logic Programming Paradigm: A 25-Year Perspective*. Springer. chapter Action languages, answer sets, and planning, 357–373.

Lukaszewicz, W. 1988. Considerations on Default Logic – An Alternative Approach. *Computational Intelligence* 4:1–16.

Melliti, T.; Regnault, D.; Richard, A.; and Sené, S. 2013. On the convergence of Boolean automata networks without negative cycles. In *Proceedings of AUTOMATA'2013*, volume 8155 of *LNCS*, 124–138. Springer.

Melliti, T.; Noual, M.; Regnault, D.; Sené, S.; and Sobieraj, J. 2015. Asynchronous dynamics of Boolean automata double-cycles. In *Proceedings of UCNC'2015*, volume 9252 of *LNCS*, 250–262. Springer.

Mendoza, L.; Thieffry, D.; and Alvarez-Buylla, E. R. 1999. Genetic control of flower morphogenesis in *Arabidopsis thaliana*. *Bioinformatics* 15:593–606.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.

Remy, E.; Mossé, B.; Chaouyia, C.; and Thieffry, D. 2003. A description of dynamical graphs associated to elementary regulatory circuits. *Bioinformatics* 19:ii172–ii178.

Remy, E.; Ruet, P.; and Thieffry, D. 2008. Graphic requirement for multistability and attractive cycles in a Boolean dynamical framework. *Advances in Applied Mathematics* 41:335–350.

Richard, A., and Comet, J.-P. 2007. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics* 155:2403–2413.

Richard, A. 2010. Negative circuits and sustained oscillations in asynchronous automata networks. *Advances in Applied Mathematics* 44:378–392.

Robert, F. 1986. *Discrete Iterations: A Metric Study*. Springer.

Roenneberg, T., and Merrow, M. 2003. The network of time: understanding the molecular circadian system. *Current Biology* 13:R198–R207.

Schwind, C., and Siegel, P. 1994. A modal logic for hypothesis theory. *Fundamenta Informaticae* 21:89–101.

Siegel, P., and Schwind, C. 1993. Modal logic based theory for non-monotonic reasonning. *Journal of Applied Non-classical Logic* 3:73–92.

Siegel, P.; Doncescu, A.; Risch, V.; and Sené, S. 2017. Vers une représentation des systèmes dynamiques discrets en logique des hypothèses *. Journées d'Intelligence Artificielle Fondamentale, Jul. 2017, Caen, France, hal-01566164 .

Thomas, R. 1973. Boolean formalization of genetic control circuits. *Journal of Theoretical Biology* 42:563–585.

Thomas, R. 1981. *Numerical Methods in the Study of Critical Phenomena*. Springer. chapter On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations, 180–193.

A critical assessment of Pollock's work on logic-based argumentation with suppositions

Mathieu Beirlaen
Ghent University
mathieubeirlaen@gmail.com

Jesse Heyninck
Ruhr-University Bochum
jesse.heyninck@ruhr-uni-bochum.de

Christian Straßer
Ruhr-University Bochum
christian.strasser@ruhr-uni-bochum.de

Abstract

John Pollock's pioneering work in logic-based argumentation continues to serve as a source of inspiration for researchers in the field of non-monotonic logic. In this paper we present and assess a largely unexplored line of Pollock's work: his system of suppositional argumentation. Via the construction of suppositional arguments, Pollock was able to represent a large additional class of important patterns of defeasible reasoning, such as conditionalization and reasoning by cases (dilemma). This resulted in a rich argumentation system that exceeds the expressive power of most present-day accounts of logic-based argumentation. In this paper, we offer three criticisms of Pollock's suppositional system, all of which relate in important ways to other systems of non-monotonic logic. Our aim with this contribution is to help shape the research agenda for advancing our understanding of formal aspects of suppositional reasoning.

1 Introduction

Our point of departure is John Pollock's pioneering work on reasoning and argumentation (Pollock 1987; 1991b; 1991a; 1994; 1995; 2008). Pollock was among the first to offer detailed accounts of the logical structure and strength of arguments, the nature of argumentative defeat, and the semantics of argument justification. His work was a clear source of inspiration for important later developments in formal argumentation, such as Dung's account of abstract argumentation (Dung 1995) or the ASPIC$^+$ framework for structured or logic-based argumentation (Modgil and Prakken 2013; 2014).[1]

An important yet largely unexplored part of Pollock's work on argumentation concerns his treatment of *suppositional reasoning*. Arguments involving suppositions or hypotheses are needed for representing a large class of defeasible inference patterns including conditionalization and reasoning by cases (also called "dilemma"). Consider the following example.

Example 1 (Kyoto protocol) *There are two candidates for an upcoming presidential election. The candidates had a debate in the capital. They were asked what measures are to be taken in order for the country to reach the Kyoto protocol objectives for reducing greenhouse gas emissions. The first candidate, a member of the purple party, argued that if she wins the election, she will reach the objectives by supporting investments in renewable energy. The second candidate, a member of the yellow party, argued that if she wins the election, she will reach the objectives by supporting sustainable farming methods.*

We have reasons to believe that, no matter which candidate wins the election, the Kyoto protocol objectives will be reached. If the purple candidate wins, she will support investments in renewable energy ($p \Rightarrow r$), which would in turn result in meeting the Kyoto objectives ($r \Rightarrow k$). Similarly, if the yellow candidate wins, she will support sustainable farming methods ($y \Rightarrow f$), which would result in meeting the Kyoto objectives ($f \Rightarrow k$). Since one of the two candidates is going to win ($p \vee y$), we can reason by cases to conclude that the Kyoto objectives will be reached (k). Schematically, this argument can be represented as follows:

$$p \vee y \left| \begin{matrix} [p] \Rightarrow r \Rightarrow k \\ [y] \Rightarrow f \Rightarrow k \end{matrix} \right| \rightsquigarrow k \qquad (1)$$

In this representation, double arrows "$\Rightarrow$" represent *defeasible* reasons. The inference from p to r, for instance, is rationally compelling yet inconclusive. (Indeed, conditional claims and reasons offered by politicians should be taken with a grain of salt, and should by no means be represented as deductive or truth-preserving.) The square brackets "[,]" are used to introduce *suppositions*. *If* p obtains, we have a defeasible argument for concluding k. Similarly, *if* y obtains, we have a defeasible argument for concluding k. Since we know that p or y, we can *discharge* these suppositions and conclude that k, as represented by the squiggly arrow "$\rightsquigarrow$".

Most present-day accounts of logic-based argumentation lack the expressive means to represent the argument in equation (1). Pollock's work is an exception. Already in 1987, Pollock extended his account of logic-based argumentation with the possibility to include suppositional arguments. He returned to this subject throughout the late eighties and early nineties, and offered a mature account of logic-based argumentation – including suppositions – in (Pollock 1995).

In this paper, we present (Section 2) and assess (Section 3) Pollock's 1995 account of suppositional argumentation.

[1] For an expert appreciation of Pollock's pioneering efforts in the field of formal argumentation, see (Prakken and Horty 2012).

We point to three problems of this account (Sections 3.1 – 3.3), and we analyze what might cause them (Section 4). (For lack of space, we do not present a *full* evaluation of Pollock's system, including all of the good news.) With this analysis we aim to advance our understanding of formal aspects suppositional reasoning, and to help shape future research on the representation of suppositional reasoning in formal argumentation and non-monotonic logic.

Notational preliminaries. $\vdash$ denotes the consequence relation of the propositional fragment of classical logic (**CL**). To obtain the formal language $\mathcal{L}$ of **CL**, we close a denumerable stock $p, q, r, \ldots$ of propositional letters under the usual **CL**-connectives $\neg, \vee, \wedge, \supset, \equiv$. We also add the verum constant $\top$ and the falsum constant $\bot$ to $\mathcal{L}$. We use lower case Greek letters $\varphi, \psi, \chi, \ldots$ as variables for members of $\mathcal{L}$, and upper case Greek letters $\Delta, \Gamma, \Theta, \ldots$ as variables for subsets of $\mathcal{L}$.

We will sometimes use the *contrariness operator* "$-$" as a notational shortcut: $\varphi = -\psi$ iff ($\varphi = \neg\psi$ or $\psi = \neg\varphi$).

2 Pollock on Suppositional Reasoning

Roughly, any theory of argumentation-based inference can be defined in terms of three elements. First, a set of well-constructed *arguments* has to be specified relative to a knowledge base. Second, a relation of *argumentative defeat* determines which arguments attack one another. And finally, a *semantics* specifies which arguments count as *justified* given a knowledge base, a set of well-constructed arguments relative to this base, and a relation of argumentative defeat. In Sections 2.1 – 2.3 below we set out how Pollock conceived each of these three elements in 1995. In our exposition we will sometimes deviate from Pollock's original presentation, indicating and motivating our alterations along the way.

2.1 Knowledge bases and arguments

Below we will think of a knowledge base as a pair containing a set of defeasible rules together with a set of background facts. For didactic purposes, the sets of defeasible rules we take into account will usually be very small, and always finite. This is slightly at odds with Pollock's characterization of defeasible reasons, since in practice no such knowledge base will ever contain *all* of our defeasible reasons. Since our arguments and examples below generalize in a straightforward manner to bigger and more realistic settings, we need not worry any further about this simplification.

Definition 1 (Knowledge base) *A* knowledge base $\mathcal{K}$ *is a pair* $\langle \mathcal{R}, \mathcal{F} \rangle$ *where* $\mathcal{R}$ *is a set of defeasible rules and where* $\mathcal{F} \subseteq \mathcal{L}$ *is a* **CL***-consistent*[2] *set of facts.*

The information given in Example 1, for instance, can be represented in terms of the knowledge base $\mathcal{K}_1 = \langle \{p \Rightarrow r, r \Rightarrow k, y \Rightarrow f, f \Rightarrow k\}, \{p \vee y\} \rangle$.

Below we represent an argument A as a pair $\langle \mathcal{S}; \Gamma \rangle$, where $\mathcal{S}$ is A's *construction sequence* and where Γ is the *hypothesis set* of A, which contains the suppositions on which we base A's conclusion. Arguments are built relative to a given knowledge base by means of the construction rules in Definition 2. The functions $\texttt{Conc}(A)$, $\texttt{Hyp}(A)$, and $\texttt{Sub}(A)$ keep track of argument A's conclusion, A's hypothesis set, and A's sub-arguments respectively.

Definition 2 (Arguments) *Given a knowledge base* $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$*, the set* $\texttt{Arg}(\mathcal{K})$ *of arguments for* $\mathcal{K}$ *contains all arguments constructed by means of the following rules:*

F*: Where* $\varphi \in \mathcal{F}$ *and* Γ *is any finite set of propositions,*

$$A : \langle \varphi; \Gamma \rangle$$

is an argument with $\texttt{Conc}(A) = \varphi$*,* $\texttt{Hyp}(A) = \Gamma$*, and* $\texttt{Sub}(A) = \{A\}$.

P*: Where* Γ *is a finite set of propositions and* $\varphi \in \Gamma$*,*

$$A : \langle [\varphi]; \Gamma \rangle$$

is an argument with $\texttt{Conc}(A) = \varphi$*,* $\texttt{Hyp}(A) = \Gamma$*, and* $\texttt{Sub}(A) = \{A\}$.

Rd*: If* $A_1, \ldots, A_n$ *are arguments with* $\texttt{Hyp}(A_i) = \Gamma$ *and* $\texttt{Conc}(A_i) = \varphi_i$ *for every* $i \in \{1, \ldots, n\}$*, and* $\varphi_1, \ldots, \varphi_n \Rightarrow \psi \in \mathcal{R}$*, then*

$$A : \langle A_1, \ldots, A_n \Rightarrow \psi; \Gamma \rangle$$

is an argument with $\texttt{Conc}(A) = \psi$*,* $\texttt{Hyp}(A) = \Gamma$*, and* $\texttt{Sub}(A) = \texttt{Sub}(A_1) \cup \ldots \cup \texttt{Sub}(A_n) \cup \{A\}$.

Rc*: If* $A_1, \ldots, A_n$ *are arguments with* $\texttt{Hyp}(A_i) = \Gamma$ *and* $\texttt{Conc}(A_i) = \varphi_i$ *for every* $i \in \{1, \ldots, n\}$*, and* $\varphi_1, \ldots, \varphi_n \vdash \psi$*, then*

$$A : \langle A_1, \ldots, A_n \rightarrow \psi; \Gamma \rangle$$

is an argument with $\texttt{Conc}(A) = \psi$*,* $\texttt{Hyp}(A) = \Gamma$*, and* $\texttt{Sub}(A) = \texttt{Sub}(A_1) \cup \ldots \cup \texttt{Sub}(A_n) \cup \{A\}$.

C*: If* B *is an argument with* $\texttt{Hyp}(B) = \Gamma \cup \{\varphi\}$ *and* $\texttt{Conc}(B) = \psi$*, then*

$$A : \langle B \rightsquigarrow \varphi \supset \psi; \Gamma \rangle$$

is an argument with $\texttt{Conc}(A) = \varphi \supset \psi$*,* $\texttt{Hyp}(A) = \Gamma$*, and* $\texttt{Sub}(A) = \texttt{Sub}(B) \cup \{A\}$.

D*: If* $B, C,$ *and* D *are arguments with*

- $\texttt{Hyp}(B) = \Gamma$*,* $\texttt{Hyp}(C) = \Gamma \cup \{\varphi\}$*, and* $\texttt{Hyp}(D) = \Gamma \cup \{\psi\}$*, and*
- $\texttt{Conc}(B) = \varphi \vee \psi$ *and* $\texttt{Conc}(C) = \texttt{Conc}(D) = \chi$*, then*

$$A : \langle B, C, D \rightsquigarrow \chi; \Gamma \rangle$$

is an argument with $\texttt{Conc}(A) = \chi$*,* $\texttt{Hyp}(A) = \Gamma$*, and* $\texttt{Sub}(A) = \{A\} \cup \texttt{Sub}(B) \cup \texttt{Sub}(C) \cup \texttt{Sub}(D)$.

A*: If* B *is an argument with* $\texttt{Conc}(B) = \varphi$ *and* $\texttt{Hyp}(B) = \Gamma$*, and* $\Gamma \subseteq \Delta$*, then*

$$A : \langle B; \Delta \rangle$$

is an argument with $\texttt{Conc}(A) = \varphi$*,* $\texttt{Hyp}(A) = \Delta$ *and* $\texttt{Sub}(A) = \{A\} \cup \texttt{Sub}(B)$

An argument A *is* hypothetical *if* $\texttt{Hyp}(A) \neq \emptyset$*; else it is* non-hypothetical*. An argument is* defeasible *if* ***Rd*** *was applied at least once in its construction sequence; else it is* strict *or* conclusive*.*

[2] $\mathcal{F} \subseteq \mathcal{L}$ is **CL***-consistent* iff $\mathcal{F} \nvdash \bot$.

The argument represented in equation (1) can be encoded as follows within Pollock's framework, on the basis of the knowledge base $\mathcal{K}_1$:

$$\begin{array}{ll} A_1: & \langle p \vee y; \emptyset\rangle \\ A_2: & \langle [p]; \{p\}\rangle \\ A_3: & \langle A_2 \Rightarrow r; \{p\}\rangle \\ A_4: & \langle A_3 \Rightarrow k; \{p\}\rangle \end{array} \quad \begin{array}{ll} A_5: & \langle [y]; \{y\}\rangle \\ A_6: & \langle A_5 \Rightarrow f; \{y\}\rangle \\ A_7: & \langle A_6 \Rightarrow k; \{y\}\rangle \\ A_8: & \langle A_1, A_4, A_7 \rightsquigarrow k; \emptyset\rangle \end{array}$$

A_1 is obtained via **F**, the rule for introducing facts in our knowledge base. A_2 and A_5 are obtained via the supposition introduction rule **P**. A_3, A_4, A_6, and A_7 are constructed by applications of defeasible rules in $\mathcal{K}_1$ via **Rd**. Finally, A_8 applies the dilemma rule **D** which allows us to discharge the suppositions p and y in A_4 and A_7 respectively in view of the information provided by A_1.

Alternatively, the argument represented in equation (1) can be encoded using the conditionalization rule **C** and the rule **Rc** for applying strict or conclusive (deductively valid) reasons:

$$\begin{aligned} A_9 &: \langle A_4 \rightsquigarrow p \supset k; \emptyset\rangle \\ A_{10} &: \langle A_7 \rightsquigarrow y \supset k; \emptyset\rangle \\ A_{11} &: \langle A_1, A_9, A_{10} \rightarrow k; \emptyset\rangle \end{aligned}$$

The construction of arguments A_8 and A_{11} suggests that there is a close correspondence between the argument construction rules **D** and **C**. Indeed, in Appendix A we show that each of these rules is admissible in the presence of the other. The set of justified conclusions of Pollock's 1995 system would remain unchanged if either we drop **C** while keeping **D** or we drop **D** while keeping **C**.

The only rule that was not applied in the construction of $A_1 - A_{11}$ above is the *foreign adoptions* rule **A**. This rule allows one to expand the hypothesis set of any argument by any additional hypothesis. For instance, we could apply this rule to expand $\texttt{Hyp}(A_3)$ with the additional hypothesis y:

$$A_{12} : \langle A_3, \{p, y\}\rangle$$

Note that $\texttt{Conc}(A_3) = \texttt{Conc}(A_{12}) = r$. The only difference between these arguments is that $\texttt{Hyp}(A_3) = \{p\}$ while $\texttt{Hyp}(A_{12}) = \{p, y\}$.

Our compact notation in Definition 2 is inspired by the way arguments are represented in the ASPIC$^+$ framework for structured argumentation (Modgil and Prakken 2013; 2014). It differs considerably from Pollock's original notation, in which arguments are represented as sequences of ordered quadruples. It is easily seen, however, that the rules in Definition 2 correspond exactly to Pollock's original rules, with one further proviso: Pollock used a single rule **R** to encompass both our **Rd** and **Rc**. We chose to introduce separate rules for applications of defeasible and conclusive rules, since this leads to a more transparent characterization of argumentative defeat.

2.2 Defeat

Pollock distinguished between rebutting defeaters and undercutting defeaters for arguments. Here, we focus exclusively on rebutting defeaters, for two reasons. The first is that our criticisms in Section 3 do not require the presence of undercutting defeaters, and that these criticisms hold *mutatis mutandis* for Pollock's system extended with undercutting defeat. The second reason is that, in the presence of undercutting defeaters, Pollock's 1995 system suffers from so-called contamination problems (cfr. infra) as shown in (Caminada 2005). Here is Pollock's 1995 conception of (rebutting) defeat:[3]

Definition 3 (Defeat) *An argument A* defeats *an argument B iff there are $A' \in \texttt{Sub}(A)$ and $B' \in \texttt{Sub}(B)$ such that*

- $\texttt{Hyp}(A') \subseteq \texttt{Hyp}(B')$,
- ***Rd** is the last rule applied in the construction of B', and*
- $\texttt{Conc}(A') = -\texttt{Conc}(B')$.

Suppose that we extend the knowledge base $\mathcal{K}_1$ from Example 1 with the further information that, at an earlier rally held in the countryside, the yellow candidate promised not to support sustainable farming methods so that the country can preserve its traditional farming methods ($y \Rightarrow \neg f$). This gives us the new argument

$$A_{13} : \langle A_5 \Rightarrow \neg f; \{y\}\rangle$$

By Definition 3, A_6 and A_{13} defeat one another relative to the extended knowledge base. Since A_6 is a sub-argument of A_7, A_8, A_{10}, and A_{11}, we also obtain that, for any $A_i \in \{A_7, A_8, A_{10}\}$, A_{13} and A_i defeat one another.

Example 2 *Let $\mathcal{K}_2 = \langle\{p \Rightarrow q, q \Rightarrow r, s \Rightarrow \neg r\}, \{p, s\}\rangle$. The following arguments all belong to $\texttt{Arg}(\mathcal{K}_2)$:*

$$\begin{array}{ll} A_1: & \langle p; \emptyset\rangle \\ A_2: & \langle A_1 \Rightarrow q; \emptyset\rangle \\ A_3: & \langle A_2 \Rightarrow r; \emptyset\rangle \\ A_4: & \langle s; \emptyset\rangle \\ A_5: & \langle A_4 \Rightarrow \neg r; \emptyset\rangle \end{array} \quad \begin{array}{ll} A_6: & \langle [q]; \{q\}\rangle \\ A_7: & \langle A_6 \Rightarrow r; \{q\}\rangle \\ A_8: & \langle A_7 \rightsquigarrow q \supset r; \emptyset\rangle \\ A_9: & \langle A_8, A_5 \rightarrow \neg q; \emptyset\rangle \end{array}$$

By Definition 3, A_3 and A_5 defeat one another. Since $A_5 \in \texttt{Sub}(A_9)$, A_3 also defeats A_9 and vice versa. Note that A_5 defeats A_7 but not vice versa, since $\texttt{Hyp}(A_5) \subset \texttt{Hyp}(A_7)$. A_7 is a sub-argument of A_8 and A_9, so the latter arguments are also defeated by A_5. Since both A_5 and A_7 are sub-arguments of A_9 the latter argument defeats itself.

*Importantly, arguments can only be defeated in conclusions obtained via **Rd**. Because of this, there is no way to defeat A_1, A_4, or A_6. For the same reason, A_9 defeats A_2 but not vice versa. The defeat relations between these arguments are represented by arrows in Figure 1. The arguments A_1, A_4, and A_6 are omitted in the figure, since they do not stand in any relation of argumentative defeat to any of the other arguments mentioned.*

[3] Definition 3 makes use of the contrariness operator "$-$" where Pollock used the classical negation operator "$\neg$". The use of "$-$" makes matters less cumbersome, while the net outcome remains the same. For instance, two arguments A and B with $\texttt{Hyp}(A) = \texttt{Hyp}(B)$, with **Rd** as the last rule applied in their construction, and with $\texttt{Conc}(B) = \neg\texttt{Conc}(A)$ defeat each other if we use "$-$". If we were to use "$\neg$" instead, B defeats A but not vice versa, while the argument $A' : \langle A \rightarrow \neg\neg\texttt{Conc}(A); \texttt{Hyp}(A)\rangle$ *does* defeat B_j.

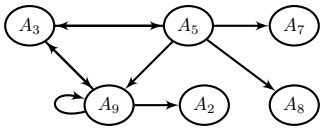


Figure 1: Argumentation graph for some arguments in Example 2.

2.3 Justification

Our formulation of Pollock's semantics is different yet equivalent to his original formulation in (Pollock 1994) and (Pollock 1995). We opt here for a more concise and more transparent formulation in terms of Dung's extensional semantics (Dung 1995).[4]

Definition 4 (Argumentation framework) *An* abstract argumentation framework *(AF) is a pair* $(\texttt{Arg}, \texttt{Def})$ *where* $\texttt{Arg}$ *is a set of arguments and* $\texttt{Def} \subseteq \texttt{Arg} \times \texttt{Arg}$ *is a binary relation of argumentative defeat.*

An AF can be represented as a graph with arguments as nodes and defeats as arrows, as in Figure 1. Relative to an AF, Dung defines a number of extensions – subsets of $\texttt{Arg}$ – on the basis of which we can evaluate arguments in $\texttt{Arg}$.

Definition 5 (Defense) *A set of arguments $\mathcal{X}$ defends an argument A iff every defeater of A is defeated by some $B \in \mathcal{X}$.*

Definition 6 (Extensions) *Let* $(\texttt{Arg}, \texttt{Def})$ *be an AF. If* $\mathcal{E} \subseteq \texttt{Arg}$ *is conflict-free, i.e. there are no* $A, B \in \mathcal{E}$ *for which* $(A, B) \in \texttt{Def}$*, then (i)* $\mathcal{E}$ *is a* complete extension *iff* $A \in \mathcal{E}$ *whenever* $\mathcal{E}$ *defends* A*; (ii)* $\mathcal{E}$ *is a* preferred extension *iff it is a set inclusion maximal complete extension; and (iii)* $\mathcal{E}$ *is the* grounded extension *iff it is the set inclusion minimal complete extension.*

Pollock's 1995 account of what it means for an argument to be justified relative to a knowledge base corresponds to Dung's *preferred skeptical semantics*:

Definition 7 (Justification) *An argument* $A \in \texttt{Arg}(\mathcal{K})$ *is* justified *relative to the knowledge base $\mathcal{K}$ if it belongs to* all *preferred extensions of the AF* $(\texttt{Arg}(\mathcal{K}), \texttt{Def})$*, where* $\texttt{Def}$ *is the defeat relation from Definition 3.*
We write $\mathcal{K} \mid\sim \varphi$ *iff there is a non-hypothetical argument* $A \in \texttt{Arg}(\mathcal{K})$ *with* $\texttt{Conc}(A) = \varphi$ *such that, for all preferred extensions $\mathcal{P}$ of the AF* $(\texttt{Arg}(\mathcal{K}), \texttt{Def})$*,* $A \in \mathcal{P}$*.*

[4]The correspondence between Dung's preferred skeptical semantics (see Def. 7) and Pollock's labeling-based semantics from 1994 and 1995 was established in (Jakobovits and Vermeir 1999, Sec. 6.3). Our use of the term "justified" is slightly different from Pollock's uses of this term in 1987 and afterwards. Especially in later work, Pollock thought of justification as a procedural, gradual notion which becomes stronger with an increase in time and means to offer additional insights into the premises. He used the term "warrant" for arguments which are justified-in-the-limit, i.e. justified in the ideal setting in which time and effort are of no importance. We do not require a gradual notion of argument justification for our present purposes, so we stick to the more common term "justified" to denote arguments that, on Pollock's view, are "warranted" or "justified-in-the-limit".

Applied to the original Example 1, Definition 7 delivers a unique preferred extension containing all of $A_1 - A_{12}$. As a result, each of $A_1 - A_{12}$ is justified relative to $\mathcal{K}_1$. This changes when we extend $\mathcal{K}_1$ with the rule $y \Rightarrow \neg f$. In the extended knowledge base, we obtain two preferred extensions. The first contains all of $A_1, \ldots, A_{12}$; it does not contain A_{13}. The second contains $A_1, \ldots, A_5, A_9, A_{12}, A_{13}$; it does not contain $A_6 - A_8, A_{10}, A_{11}$.

Applied to Example 2, we again obtain two preferred extensions. Each of these contains the undefeated arguments A_1, A_4, and A_6. For the other arguments given above, the extensions are depicted in Figure 2, where green arguments belong to the extension in question, and red arguments do not. Of course, the figures depict only a very small fragment of the full AF for Example 2, since $\texttt{Arg}(\mathcal{K}_2)$ contains plenty of further arguments. Adding all further arguments to this picture, however, would not affect the status of any of the arguments depicted. For the sake of clarity, we confine our illustrations to the most relevant arguments, and we safely exclude the other arguments. The same holds true for the other examples in the remainder of this paper.

Note that A_2 is justified relative to $\mathcal{K}_2$ (since it belongs to both extensions), while e.g. A_3 and A_5 are not, so $\mathcal{K}_2 \mid\sim q$, while $\mathcal{K}_2 \not\mid\sim r$ and $\mathcal{K}_2 \not\mid\sim \neg r$.

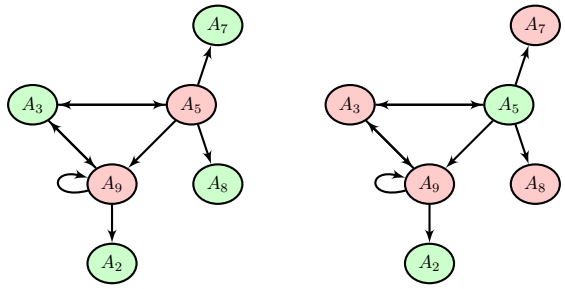


Figure 2: Preferred extensions of $\mathcal{K}_2$ in Example 2.

3 A Critical Assessment of Pollock's 1995 account

In Sections 3.1 – 3.3 we offer three criticisms of Pollock's system. The first is based on Pollock's so-called priority principle (Section 3.1). The second and third criticism are based on well-known and generally accepted postulates from the literature on conditional reasoning and nonmonotonic logic respectively (Sections 3.2 and 3.3).

3.1 The Priority Principle

Pollock believed the following principle to be a true principle of defeasible reasoning:

> In general, given a rebutting defeater for the last step of a defeasible argument, we work backwards withdrawing conclusions until we get to the last defeasible step of the argument. Defeat must extend backwards over deductive steps, because we cannot withdraw the conclusion of a deductive step without withdrawing the premise, but when we come to a defeasible step we

withdraw only the conclusion, retaining the premise and taking the prima facie reason to be defeated. I refer to this as the *priority principle* – we give priority to the earlier defeasible steps of an argument (Pollock 1991b, p. 387, emphasis in original)

The priority principle states that, given a chain

$$\begin{array}{ll} D_1: & \langle D_0 \Rightarrow \varphi_1, \emptyset\rangle \\ D_2: & \langle D_1 \Rightarrow \varphi_2, \emptyset\rangle \\ \vdots & \vdots \\ D_n: & \langle D_{n-1} \Rightarrow \varphi_n, \emptyset\rangle \end{array}$$

of applications of **Rd** and a defeater E with $\texttt{Conc}(E) = \neg\varphi_n$ for the *last* link of this chain, we 'cut off' the chain at the last defeasible step (assuming that there are no further arguments which defend D_n from the defeat by E). As a result, the argument D_n is no longer justified and, absent further defeaters, *all* of $D_1, \ldots, D_{n-1}$ are justified. In particular, we should not be able to use E for constructing a defeater for any of $D_1, \ldots, D_{n-1}$.

As illustrated above, the priority principle is respected for Example 2, since $\mathcal{K}_2 \mid\!\sim q$. Pollock himself used this example to illustrate the priority principle for his account from Section 2, and to point out that the principle fails for some of his earlier accounts (Pollock 1991b, Sec. 10). We will return to this point in Section 4.2. For now, it suffices to see that Pollock defended the priority principle, and that he was prepared to modify his account of argumentative defeat in order to respect it. But does the priority principle hold in general for Pollock's account presented in Section 2? Unfortunately, it does not, as shown in Example 3.[5]

Example 3 (Failure of the priority principle) *Let* $\mathcal{K}_3 = \langle\{p \Rightarrow q, q \Rightarrow \neg s, q \Rightarrow \neg t, r \Rightarrow s, \neg r \Rightarrow t\}, \{p\}\rangle$. *The following arguments all belong to* $\texttt{Arg}(\mathcal{K}_3)$*:*

$$\begin{aligned} A_1 &: \langle p; \emptyset\rangle \\ A_2 &: \langle A_1 \Rightarrow q; \emptyset\rangle \\ A_3 &: \langle A_2 \Rightarrow \neg s; \emptyset\rangle \\ A_4 &: \langle A_2 \Rightarrow \neg t; \emptyset\rangle \\ A_5 &: \langle\langle[r]; \{r\}\rangle \Rightarrow s; \{r\}\rangle \rightarrow s \vee t; \{r\}\rangle \\ A_6 &: \langle\langle[\neg r]; \{\neg r\}\rangle \Rightarrow t; \{\neg r\}\rangle \rightarrow s \vee t; \{\neg r\}\rangle \\ A_7 &: \langle\langle[q]; \{q\}\rangle \Rightarrow \neg s; \{q\}\rangle \rightsquigarrow q \supset \neg s; \emptyset\rangle \\ A_8 &: \langle\langle[q]; \{q\}\rangle \Rightarrow \neg t; \{q\}\rangle \rightsquigarrow q \supset \neg t; \emptyset\rangle \\ A_9 &: \langle\langle \rightarrow r \vee \neg r; \emptyset\rangle, A_5, A_6 \rightsquigarrow s \vee t; \emptyset\rangle \\ A_{10} &: \langle A_7, A_8, A_9 \rightarrow \neg q; \emptyset\rangle \\ A_{11} &: \langle A_3, A_9 \rightarrow t; \emptyset\rangle \\ A_{12} &: \langle A_4, A_9 \rightarrow s; \emptyset\rangle \end{aligned}$$

In order for the priority principle to be respected, we need A_2 *to be justified relative to* $\mathcal{K}_3$*. Unfortunately, it is not.* A_2 *is defeated by* A_{10}*, and the latter argument defends itself against its defeaters* A_3*,* A_4*,* A_{11}*, and* A_{12}*. As a result, we obtain a preferred extension containing* A_{10} *– part of this extension is depicted in Figure 3.*

[5]The argument $\langle \rightarrow r \vee \neg r; \emptyset\rangle \in \texttt{Sub}(A_9)$ applies **Rc** without reference to any sub-arguments in view of the **CL**-valid inference $\emptyset \vdash r \vee \neg r$.

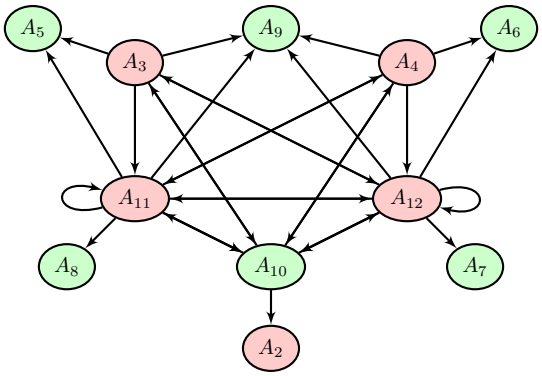


Figure 3: Sample of a preferred extension of $\mathcal{K}_3$.

Example 3 makes clear that, perhaps contrary to Pollock's own expectations, there are cases in which the priority principle fails for his system, and in which we can let defeat extend backwards not just over deductive steps, but also over defeasible steps. Pollock himself admitted that there are *contraposible* defeasible reasons for which this backward move is allowed. Still, he maintained that not all defeasible reasons are contraposible in this sense, and that the priority principle must come out true in a theory of defeasible reasoning, see in particular (Pollock 1991b).

Our discussion on Pollock's priority principle finds a close parallel in the literature on default logic. Default logicians, like Pollock, tend to reject the view that defeasible rules are contraposible. Indeed, default rules are not contraposible in Reiter's default logic (Reiter 1980), and the priority principle is respected for Reiter's formalism. Moreover, enrichments of Reiter's original proposal that allow for versions of Pollock's dilemma (**D**) or conditionalization (**C**) rules have been criticized for licensing contraposition of default rules (Moinard 1994; Roos 1998). It seems, then, that most default logicians, like Pollock, believe that the priority principle ought to be respected for defeasible reasoning.

3.2 Argumentation and the Ramsey Test

In his 1968 article "A theory of conditionals" Robert Stalnaker set out to answer the question "How does one evaluate a conditional statement?" His answer crucially turns on a suggestion made earlier by Frank Ramsey (Ramsey 1931). In the words of Stalnaker, Ramsey's suggestion amounts to a simple thought experiment. To evaluate a conditional,

> add the antecedent (hypothetically) to your stock of knowledge (or beliefs), and then consider whether or not the consequent is true. Your belief about the conditional should be the same as your hypothetical belief, under this condition, about the consequent (Stalnaker 1968, p. 101).

This test for the acceptance or rejection of conditional statements is nowadays known as the *Ramsey Test*. It figures prominently in the literature on theories of conditionals – see e.g. (Arlo-Costa and Egré 2016; Bennett 2003). The Ramsey test is a useful tool for assessing conditional statements

the antecedent of which is false or undetermined, and it explicitly connects the evaluation of conditionals with suppositional reasoning.

The Ramsey test is about conditionals, and not about arguments. But we can easily turn the test into a device for evaluating suppositional arguments. Suppose we want to evaluate a simple suppositional argument A relative to a knowledge base $\mathcal{K} = \langle \mathcal{R}, \mathcal{F} \rangle$ containing a rule $\varphi \Rightarrow \psi$:

$$A : \langle \langle [\varphi]; \{\varphi\} \rangle \Rightarrow \psi; \{\varphi\} \rangle$$

To check whether A is justified relative to $\mathcal{K}$, we can add its supposition φ to the facts in $\mathcal{K}$ and check whether the non-suppositional argument B is justified relative to the resulting extended knowledge base $\langle \mathcal{R}, \mathcal{F} \cup \{\varphi\} \rangle$:

$$B : \langle \langle \varphi; \emptyset \rangle \Rightarrow \psi; \emptyset \rangle$$

It ought to be that B is justified relative to $\langle \mathcal{R}, \mathcal{F} \cup \{\varphi\} \rangle$ (provided that $\mathcal{F} \cup \{\varphi\}$ is **CL**-consistent) if and only if A is justified relative to $\mathcal{K}$. This is the argumentative analogue of Stalnaker's advice for evaluating conditionals. We refer to it as the *Argumentative Ramsey Test* (ART), and consider this principle equally acceptable as the original Ramsey Test. Examples 4 and 5 illustrate the failure of the ART for Pollock's 1995 system in both directions.

Example 4 (Right-left failure of the ART) *Let $\mathcal{K}_4 = \langle \mathcal{R}_4, \mathcal{F}_4 \rangle$ with $\mathcal{R}_4 = \{p \Rightarrow \neg q, p \Rightarrow s, s \Rightarrow q, r \Rightarrow \neg s\}$ and $\mathcal{F}_4 = \{r\}$. Note that A_1 is justified relative to $\mathcal{K}_4$: A_1 is defeated by A_4, but defended by the undefeated A_2:*

$$A_1 : \langle \langle [p]; \{p\} \rangle \Rightarrow \neg q; \{p\} \rangle$$
$$A_2 : \langle \langle r; \emptyset \rangle \Rightarrow \neg s; \emptyset \rangle$$
$$A_3 : \langle \langle [p]; \{p\} \rangle \Rightarrow s; \{p\} \rangle$$
$$A_4 : \langle A_3 \Rightarrow q; \{p\} \rangle$$

However, A'_1 is not justified relative to $\langle \mathcal{R}_4, \mathcal{F}_4 \cup \{p\} \rangle$:

$$A'_1 : \langle \langle p; \emptyset \rangle \Rightarrow \neg q; \emptyset \rangle$$
$$A'_3 : \langle \langle p; \emptyset \rangle \Rightarrow s; \emptyset \rangle$$
$$A'_4 : \langle A'_3 \Rightarrow q; \emptyset \rangle$$

A'_4 defeats not only A'_1; it also defeats A_2 in view of the symmetrical defeat between A_2 and A'_3. As a result, we obtain a preferred extension $\mathcal{P}$ with $A'_3, A'_4 \in \mathcal{P}$ and $A'_1, A'_2 \notin \mathcal{P}$.

Example 5 (Left-right failure of the ART) *Let $\mathcal{K}_5 = \langle \mathcal{R}_5, \emptyset \rangle$ with $\mathcal{R}_5 = \{p \Rightarrow \neg q, \neg p \Rightarrow q, r \Rightarrow \neg p, \neg r \Rightarrow q\}$. First, note that A_2 below is justified relative to $\langle \mathcal{R}_5, \{p\} \rangle$: A_2 is defeated by A_6, but the undefeated A_1 defeats A_3 and A_6, defending A_2:*

$$A_1 : \langle p; \emptyset \rangle$$
$$A_2 : \langle A_1 \Rightarrow \neg q; \emptyset \rangle$$
$$A_3 : \langle \langle [r]; \{r\} \rangle \Rightarrow \neg p; \{r\} \rangle$$
$$A_4 : \langle A_3 \Rightarrow q; \{r\} \rangle$$
$$A_5 : \langle \langle [\neg r]; \{\neg r\} \rangle \Rightarrow q; \{\neg r\} \rangle$$
$$A_6 : \langle \langle \rightarrow r \vee \neg r; \emptyset \rangle, A_4, A_5 \rightsquigarrow q; \emptyset \rangle$$

In order for the ART to be respected, it ought to be that A'_2 is justified relative to $\mathcal{K}_5$. But A'_2 cannot be defended against the defeat by A_6, since $A_1 \notin \texttt{Arg}(\mathcal{K}_5)$ and since A'_1 does not defeat A_6. As a result, A'_2 is not justified relative to $\mathcal{K}_5$.

$$A'_1 : \langle [p]; \{p\} \rangle \qquad A'_2 : \langle A'_1 \Rightarrow \neg q; \{p\} \rangle$$

3.3 Disjunction in the Premisses

The property of *disjunction in the premisses* – we borrow the name from (Makinson 2003) – is generally regarded as a desirable principle for disjunctive reasoning:[6]

$$\text{If } \langle \mathcal{R}, \mathcal{F} \cup \{\varphi\} \rangle \mid\sim \chi \text{ and } \langle \mathcal{R}, \mathcal{F} \cup \{\psi\} \rangle \mid\sim \chi, \text{ then } \langle \mathcal{R}, \mathcal{F} \cup \{\varphi \vee \psi\} \rangle \mid\sim \chi \quad \text{(OR)}$$

Unfortunately, OR fails for Pollock's 1995 system, as our next example shows.

Example 6 (Failure of OR) *Let $\mathcal{R}_6 = \{p \Rightarrow r \vee s, r \Rightarrow t, t \Rightarrow v, s \Rightarrow v, q \Rightarrow v, r \Rightarrow u, u \Rightarrow \neg t, p \Rightarrow \neg u\}$. First, note that A_3 below is justified relative to $\langle \mathcal{R}_6, \{p\} \rangle$. A_3 is defeated by A_4, but the undefeated A_5 defends A_3:*

$$A_1 : \langle \langle \langle [r]; \{r\} \rangle \Rightarrow t; \{r\} \rangle \Rightarrow v; \{r\} \rangle$$
$$A_2 : \langle \langle [s]; \{s\} \rangle \Rightarrow v; \{s\} \rangle$$
$$A_3 : \langle \langle \langle p; \emptyset \rangle \Rightarrow r \vee s; \emptyset \rangle, A_1, A_2 \rightsquigarrow v; \emptyset \rangle$$
$$A_4 : \langle \langle \langle [r]; \{r\} \rangle \Rightarrow u; \{r\} \rangle \Rightarrow \neg t; \{r\} \rangle$$
$$A_5 : \langle \langle p; \emptyset \rangle \Rightarrow \neg u; \emptyset \rangle$$

Next, consider the knowledge base $\langle \mathcal{R}_6, \{q\} \rangle$. Here, the argument A_6 stands undefeated and is justified:

$$A_6 : \langle \langle q; \emptyset \rangle \Rightarrow v; \emptyset \rangle$$

Finally, consider $\langle \mathcal{R}_6, \{p \vee q\} \rangle$, where we have the non-hypothetical argument A_{11} for the conclusion v:

$$A_7 : \langle \langle \langle [r]; \{p, r\} \rangle \Rightarrow t; \{p, r\} \rangle \Rightarrow v; \{p, r\} \rangle$$
$$A_8 : \langle \langle [s]; \{p, s\} \rangle \Rightarrow v; \{p, s\} \rangle$$
$$A_9 : \langle \langle \langle [p]; \{p\} \rangle \Rightarrow r \vee s; \{p\} \rangle, A_7, A_8 \rightsquigarrow v; \{p\} \rangle$$
$$A_{10} : \langle \langle [q]; \{q\} \rangle \Rightarrow v; \{q\} \rangle$$
$$A_{11} : \langle \langle p \vee q; \emptyset \rangle, A_9, A_{10} \rightsquigarrow v; \emptyset \rangle$$

Note that A_7, A_9, and A_{11} are defeated by A_4. There is no defense against this defeat: $A_5 \notin \texttt{Arg}(\langle \mathcal{R}_6, \{p \vee q\} \rangle)$, and A_{12} does not defeat A_4 since $\texttt{Hyp}(A_{12}) \not\subseteq \texttt{Hyp}(A_4)$:

$$A_{12} : \langle \langle [p]; \{p\} \rangle \Rightarrow \neg u; \{p\} \rangle$$

As a result, A_{11} is not justified relative to $\langle \mathcal{R}_6, \{p \vee q\} \rangle$. More generally, we have $\langle \mathcal{R}_6, \{p\} \rangle \mid\sim v$ and $\langle \mathcal{R}_6, \{q\} \rangle \mid\sim v$, while $\langle \mathcal{R}_6, \{p \vee q\} \rangle \not\mid\sim v$, in clear violation to OR.[7]

4 Further Analysis

The priority principle, the ART, and OR all fail for Pollock's 1995 system of logic-based argumentation with suppositions. We will not attempt a defense of these principles here. Instead, we offer some further insights as to what causes their failure in Pollock's system.

[6] In their influential paper on preferential models, Kraus, Lehmann & Magidor refer to this principle by the name "OR" (Kraus, Lehmann, and Magidor 1990).

[7] Note that, alternatively, we could have constructed A_9 via applications of **A** to A_1 and A_2 respectively:

$$A'_9 : \langle \langle [p]; \{p\} \rangle \Rightarrow r \vee s; \{p\} \rangle, \langle A_1; \{p, r\}, \langle A_2; \{p, s\} \rangle \rightsquigarrow v; \{p\} \rangle$$

This does not affect the outcome for Example 6: A'_9 is defeated by A_4 and vice versa. So we obtain a preferred extension which does not contain A'_9, and which does not contain the argument A'_{11} which substitutes A'_9 for A_9 in the construction sequence of A_{11}.

4.1 Why Does the Priority Principle Fail?

On Pollock's account, the rules **C** and **D** for discharging hypotheses can be applied without further restrictions in the construction of arguments. We believe it is the unrestricted use of these dischargement rules which ultimately leads to the failure of the priority principle. More precisely, we believe greater care should be taken when we sequentially apply dischargement rules in an argument's construction. The following example will help us explicate this point.

Example 7 *Let* $\mathcal{K}_7 = \langle\{p \Rightarrow q, q \Rightarrow r, r \Rightarrow \neg q, \neg q \Rightarrow t\}, \emptyset\rangle$.

$A_1 : \langle [p]; \{p\}\rangle$ $\quad$ $A_4 : \langle A_3 \Rightarrow \neg q; \{p\}\rangle$
$A_2 : \langle A_1 \Rightarrow q; \{p\}\rangle$ $\quad$ $A_5 : \langle A_4 \Rightarrow t; \{p\}\rangle$
$A_3 : \langle A_2 \Rightarrow r; \{p\}\rangle$ $\quad$ $A_6 : \langle A_5 \rightsquigarrow p \supset t; \emptyset\rangle$

The defeasible rules in $\mathcal{K}_7$ *form a chain with conflicting sub-conclusions* q *and* $\neg q$*. As expected, the arguments* $A_4 - A_6$ *in this chain are not justified in view of the defeat between* A_2 *and* A_4*, which makes* $A_4 - A_6$ *self-defeating. This seems desirable: we would not want to detach the conclusion* $p \supset t$ *relative to* $\mathcal{K}_7$*, since it was obtained over the conflicting sub-conclusions* q *and* $\neg q$*. But there is a different road to the same conclusion: we can 'cut up' the chain* $[p] \Rightarrow q \Rightarrow r \Rightarrow \neg q \Rightarrow t$ *into two separate chains* $[p] \Rightarrow q \Rightarrow r$ *and* $[r] \Rightarrow \neg q \Rightarrow t$*. Since these chains are based on different hypotheses, no defeat arises between them. After discharging their respective hypotheses we can 'glue' the chains back together, and we obtain the justified argument* A_{12} *for the conclusion* $p \supset t$*.*

$A_7 : \langle A_3 \rightsquigarrow p \supset r; \emptyset\rangle$ $\quad$ $A_{10} : \langle A_9 \Rightarrow t; \{r\}\rangle$
$A_8 : \langle [r]; \{r\}\rangle$ $\quad$ $A_{11} : \langle A_{10} \rightsquigarrow r \supset t; \emptyset\rangle$
$A_9 : \langle A_8 \Rightarrow \neg q; \{r\}\rangle$ $\quad$ $A_{12} : \langle A_7, A_{11} \rightarrow p \supset t; \emptyset\rangle$

Figure 4 depicts part of the unique preferred extension of $\mathcal{K}_7$ *for the arguments constructed above. Arguments* A_1*,* A_8*,* A_9*, and* A_{10} *stand undefeated and are not included in the picture.*

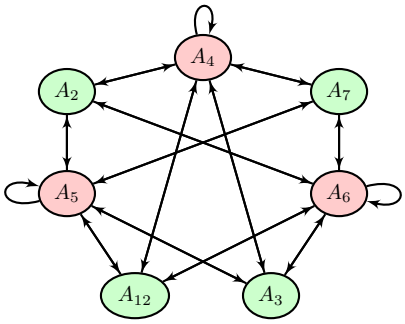


Figure 4: A sample of the unique preferred extension of $\texttt{Arg}(\mathcal{K}_7)$.

In the construction of A_{12} in Example 7 we were able to 'shield off' the conflict between A_{12}'s sub-conclusions q and $\neg q$ by arriving at them on the basis of different assumptions. As a result, A_{12} is not self-defeating. This move, we believe, is too permissive. In the construction of A_{12} we *relied* on *both* q and $\neg q$ just as we relied on both of these sub-conclusions in the construction of A_5. Splitting up the reasoning chain

$$\big([p] \Rightarrow q \Rightarrow r \Rightarrow \neg q \Rightarrow t\big) \rightsquigarrow p \supset t$$

into two chains

$$\big([p] \Rightarrow q \Rightarrow r\big) \rightsquigarrow p \supset r \text{ and } \big([r] \Rightarrow \neg q \Rightarrow t\big) \rightsquigarrow r \supset t$$

and then 'glueing' these chains together as in A_{12}'s construction, should not make this argument any more acceptable than the self-defeating argument A_6. A straightforward fix for this asymmetry between the epistemic status of A_6 and A_{12} respectively, is to prohibit sequential applications of the dischargement rules **C** and **D**.[8] This would effectively block the construction of A_{12}, in which **C** was applied twice in a row (first in A_7, then in A_{11}). A more fine-grained solution would be to keep better track of dependencies when sequentially discharging hypotheses in an argument's construction, so that e.g., in the construction of A_{12} the dependency on the conflicting sub-conclusions q and $\neg q$ would cause self-defeat.

How does all this relate to the failure of the priority principle? Reconsider Example 3 from Section 3.1, in which the argument A_2 ought to be justified in order for the priority principle to be respected. We found instead that A_2 is not justified in view of its defeat by A_{10}. A closer look reveals that we discharged hypotheses on three different occasions in A_{10}'s construction sequence: we applied **C** in A_7 and A_8 respectively, and we applied **D** in A_9. And, like in the construction of A_{12} in Example 7, we 'shielded off' conflicts in doing so, as the following schematic representation of these sub-arguments of A_{10} makes clear:

$$\big([q] \Rightarrow \neg s\big) \rightsquigarrow q \supset \neg s \qquad (A_7)$$

$$\big([q] \Rightarrow \neg t\big) \rightsquigarrow q \supset \neg t \qquad (A_8)$$

$$\rightarrow r \vee \neg r \left| \begin{matrix} [r] \Rightarrow s \\ [\neg r] \Rightarrow t \end{matrix} \right| \rightsquigarrow s \vee t \qquad (A_9)$$

As in Example 7, a suitable restriction on sequential applications of dischargement rules would suffice to prevent the argument A_{10} from Example 3 from belonging to a preferred extension of its knowledge base. More boldly, a prohibition on sequential applications of dischargement rules would block its construction altogether. Any of these fixes would result in A_2 being justified in Example 3, and would defuse our counter-example to Pollock's priority principle. Which of these fixes is most appropriate, and how it is best spelled out, is a matter we leave for future research.

4.2 Why do the ART and OR Fail?

Pollock's earliest account of logic-based argumentation with suppositions dates back to 1987. The system presented there

[8] One may still allow for *nested* applications of these rules, e.g., when **C** is applied in the subargument $(A; \Gamma \cup \{\phi\})$ of $(B; \Gamma) = (A; \Gamma \cup \{\phi\}) \rightsquigarrow (\phi \supset \psi)$. In contrast to sequential applications, when nesting these rules, we do not discharge hypotheses before charging them in later applications of the same rules.

differs from the 1995 system in four notable ways. First, it uses the grounded semantics (see Def. 7) as an underlying semantical engine instead of the preferred skeptical semantics. Second, it makes use of identity ($\texttt{Hyp}(A') = \texttt{Hyp}(B')$) rather than subset inclusion ($\texttt{Hyp}(A') \subseteq \texttt{Hyp}(B')$) in Def. 3. Third, it lacks the argument construction rules **A** and **D**. And fourth, it comes with an *a priori* exclusion of self-defeating arguments: self-defeaters are not considered part of the AF relative to a given knowledge base.

In 1991 Pollock moved from identity to subset inclusion in his account of defeat, a move to which he referred as the *subset rule*. He also introduced the foreign adoptions rule **A**, and pointed out that the subset rule is "closely related" to **A** (Pollock 1991b, p. 389). The subset rule and foreign adoptions go hand-in-hand indeed: in the presence of **A**, we can indirectly let an argument A defeat an argument B with a conflicting conclusion whenever $\texttt{Hyp}(A) \subset \texttt{Hyp}(B)$, even if we replace "$\subseteq$" with "=" in Def. 3 (call the resulting account =-*defeat*). For instance, A_1 does not =-defeat A_2, but A_3, which applies **A** to A_1, does =-defeat A_2:

$$A_1 : \langle\langle [p]; \{p\}\rangle \Rightarrow q; \{p\}\rangle$$
$$A_2 : \langle\langle [p]; \{p, r\}\rangle, \langle [r]; \{p, r\}\rangle \Rightarrow \neg q; \{p, r\}\rangle$$
$$A_3 : \langle A_1; \{p, r\}\rangle$$

In Appendix B we show that, even if we move to =-defeat, the set of arguments justified relative to a given knowledge base remains the same as long as we also have **A** around. To obtain a system in which the *identity rule* holds – as in Pollock's 1987 system – we move to =-defeat *and* drop the foreign adoptions rule.

Pollock's main motivation for introducing the subset rule and the foreign adoptions rule is that, without these rules, the priority principle fails. He used Example 2 to illustrate this point. In an account with the identity rule, the argument A_5 in this example would no longer defeat any of A_7, A_8, or A_9. A_9 would no longer defeat itself, and we would obtain a preferred extension which does not contain A_2, in violation to the priority principle.

Unfortunately, we saw in Section 3.1 that even in the presence of the subset rule and the foreign adoptions rule, the priority principle fails for Pollock's 1995 account. Moreover, a closer inspection of Examples 4, 5, and 6 shows that it is the subset rule which is to blame for the failure of the ART and OR for Pollock's system.

First, consider Example 4 and suppose that we implement the identity rule by moving to =-defeat and by disallowing applications of **A**. In the resulting system, A_2 no longer defeats A_4 since $\texttt{Hyp}(A_2) \neq \texttt{Hyp}(A_4)$. The argument $A_2' : \langle\langle r; \{p\}\rangle \Rightarrow \neg s; \{p\}\rangle$ *does* defeat A_4, but the defeat is symmetrical: A_4 also defeats A_2', so we obtain a preferred extension which contains A_4, and which contains neither A_2' nor A_1. The upshot is that A_1 is not justified relative to $\mathcal{K}_4$, and that the example fails as a counter-example to the ART.

Next, consider Example 5 and suppose again that we implement the identity rule. As with the subset rule, A_2 is justified relative to $\langle\mathcal{R}_5, \{p\}\rangle$: A_2 is =-defeated by A_6, but A_6 is defeated (in A_3) by the undefeated argument $\langle p; \{r\}\rangle$. For the ART to fail for this example, A_2' ought *not* be justified relative to $\mathcal{K}_5$. Note that A_6 does not =-defeat A_2' since $\texttt{Hyp}(A_6) \neq \texttt{Hyp}(A_2')$. The argument A_6' below *does* =- defeat A_2'. However, the undefeated argument $\langle p; \{r, p\}\rangle$ =-defeats A_6' (in A_3') and thereby defends A_2', so that A_2' remains justified relative to $\mathcal{K}_5$, and so the example fails as a counter-example to the ART.

$$A_3' : \langle\langle [r]; \{r, p\}\rangle \Rightarrow \neg p; \{r, p\}\rangle$$
$$A_4' : \langle A_3' \Rightarrow q; \{r, p\}\rangle$$
$$A_5' : \langle\langle [\neg r]; \{\neg r, p\}\rangle \Rightarrow q; \{\neg r, p\}\rangle$$
$$A_6' : \langle\langle \rightarrow r \vee \neg r; \{p\}\rangle, A_4', A_5' \rightsquigarrow q; \{p\}\rangle$$

Finally, consider Example 6. Here, the crucial thing to note is that, if we move to a system with the identity rule, the argument A_3 is no longer justified relative to $\langle\mathcal{R}_6, \{p\}\rangle$. A_3 is =-defeated by A_4, but A_5 no longer defends A_3 from this defeat. Instead, A_4 is =-defeated by the argument $A_5' : \langle\langle p; \{r\}\rangle \Rightarrow \neg u; \{r\}\rangle$. The =-defeat is symmetrical (A_4 also =-defeats A_5') but still we obtain a preferred extension of $\langle\mathcal{R}_6, \{p\}\rangle$ that contains A_4 and that contains neither A_3 nor A_5'. Thus, $\langle\mathcal{R}_6, \{p\}\rangle \not\sim v$ and the example fails as a counter-example to OR.

We have pointed to the subset rule as the cause of the failure of the ART and OR for Pollock's 1995 system, and we have shown, for all relevant examples, that this failure is prevented if we move from the subset rule to the identity rule. It seems, then, that by implementing the identity rule we can preserve OR and the ART, and avoid two out of the three criticisms offered against Pollock's 1995 system in Section 3. Unfortunately though, moving to the identity rule comes with a severe additional problem: it leads to a violation of the *non-interference* property, which was proposed in (Caminada, Carnielli, and Dunne 2012) as a rationality postulate for argumentation-based logics. The non-interference postulate demands that syntactically disjoint knowledge bases[9] do not *contaminate* one another in the sense that uniting these knowledge bases ought not lead to the unacceptability of arguments that are acceptable in one of the original knowledge bases.

Example 8 *Let* $\mathcal{K}_8 = \langle\{p \Rightarrow q, \neg p \Rightarrow \neg q, r \Rightarrow q, \neg r \Rightarrow \neg q, s \Rightarrow t\}, \{p \vee r, \neg p \vee \neg r, s\}\rangle$. *The argument* A_7 *has no atoms in common with any of* $A_1 - A_6$*, so one would expect* A_7 *to be justified relative to* $\mathcal{K}_8$*:*

$$A_1 : \langle\langle [p]; \{p\}\rangle \Rightarrow q; \{p\}\rangle$$
$$A_2 : \langle\langle [r]; \{r\}\rangle \Rightarrow q; \{r\}\rangle$$
$$A_3 : \langle\langle p \vee r; \emptyset\rangle, A_1, A_2 \rightsquigarrow q; \emptyset\rangle$$

$$A_4 : \langle\langle [\neg p]; \{\neg p\}\rangle \Rightarrow \neg q; \{\neg p\}\rangle$$
$$A_5 : \langle\langle [\neg r]; \{\neg r\}\rangle \Rightarrow \neg q; \{\neg r\}\rangle$$
$$A_6 : \langle\langle \neg p \vee \neg r; \emptyset\rangle, A_4, A_5 \rightsquigarrow \neg q; \emptyset\rangle$$
$$A_7 : \langle\langle s; \emptyset\rangle \Rightarrow t; \emptyset\rangle$$

[9] Two knowledge bases are *syntactically disjoint* if they have no atomic sub-formulas in common.

A_7 is defeated by A_8:

$$A_8 : \langle A_3, A_6 \to \neg t; \emptyset \rangle$$

In Pollock's 1995 system (with the subset rule) A_8 is self-defeating, since A_3 defeats $A_4 - A_6$, and since A_6 defeats $A_1 - A_3$. Since A_8 is self-defeating, A_7 remains justified.

In a system with the identity rule, however, A_8 is no longer self-defeating. Indeed, no =-defeats arise between the arguments $A_1 - A_6$. Moreover, all =-defeaters of A_8 are in turn =-defeated by other arguments. The argument $A_3^{\neg p}$ below, for instance, mirrors the construction of A_3 but adds the hypothesis $\neg p$ so that it =-defeats A_4 and A_6; but $A_3^{\neg p}$ is in turn defeated by the undefeatable argument A_9:

$$A_1^{\neg p} : \langle \langle [p]; \{p, \neg p\} \rangle \Rightarrow q; \{p, \neg p\} \rangle$$
$$A_2^{\neg p} : \langle \langle [r]; \{r, \neg p\} \rangle \Rightarrow q; \{r, \neg p\} \rangle$$
$$A_3^{\neg p} : \langle \langle p \vee r; \{\neg p\} \rangle, A_1^{\neg p}, A_2^{\neg p} \rightsquigarrow q; \{\neg p\} \rangle$$
$$A_9 : \langle \langle [p]; \{p, \neg p\} \rangle, \langle [\neg p]; \{p, \neg p\} \rangle \to \neg q; \{p, \neg p\} \rangle$$

Analogously to the construction of $A_3^{\neg p}$, we can construct the argument $A_3^{\neg r}$ which mirrors the construction of A_3 but adds the hypothesis $\neg r$ (to obtain this argument, simply replace all occurrences of "$\neg p$" with "$\neg r$" in the construction of $A_3^{\neg p}$). $A_3^{\neg r}$ then =-defeats A_5 and A_6, but A_{10} in turn =-defeats $A_3^{\neg r}$:

$$A_{10} : \langle \langle [r]; \{r, \neg r\} \rangle, \langle [\neg r]; \{r, \neg r\} \rangle \to \neg q; \{r, \neg r\} \rangle$$

Analogously to the construction of $A_3^{\neg p}$ and $A_3^{\neg r}$, we can construct the arguments A_6^p and A_6^r, which mirror the construction of A_6 but add the hypothesis p, respectively r. It is easily, seen, however, that these arguments too are in turn defeated by the undefeatable arguments A_{11} and A_{12} respectively:

$$A_{11} : \langle \langle [p]; \{p, \neg p\} \rangle, \langle [\neg p]; \{p, \neg p\} \rangle \to q; \{p, \neg p\} \rangle$$
$$A_{12} : \langle \langle [r]; \{r, \neg r\} \rangle, \langle [\neg r]; \{r, \neg r\} \rangle \to q; \{r, \neg r\} \rangle$$

Altogether, the undefeatible arguments $A_9 - A_{12}$ defend A_8 against the =-defeats by $A_3^{\neg p}$, $A_3^{\neg r}$, A_6^p, and A_6^r. As a result, A_7 is not justified relative to $\mathcal{K}_8$. In fact, the seemingly incoherent argument A_8 is justified*, as it belongs to the unique preferred extension relative to $\mathcal{K}_8$ – see also Figure 5.*

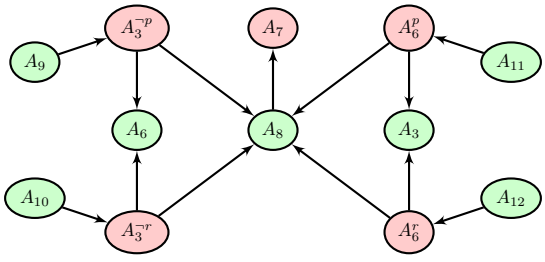


Figure 5: A sample of the unique preferred extension of $\texttt{Arg}(\mathcal{K}_8)$ with the identity rule.

This outcome for Example 8 is clearly counter-intuitive. To see how it causes a violation of non-interference, partition $\mathcal{K}_8$ into the syntactically disjoint knowledge bases $\mathcal{K}_8^a - \langle \{s \Rightarrow t\}, \{s\} \rangle$ and $\mathcal{K}_8^b = \langle \{p \Rightarrow q, \neg p \Rightarrow \neg q, r \Rightarrow q, \neg r \Rightarrow \neg q\}, \{p \vee r, \neg p \vee \neg r\} \rangle$. Note that the argument $\langle \langle s; \emptyset \rangle \Rightarrow t; \emptyset \rangle$ is justified relative to $\mathcal{K}_8^a$, while in Example 8 we showed that this argument is not justified relative to $\mathcal{K}_8$, which unites $\mathcal{K}_8^a$ and $\mathcal{K}_8^b$.

We end with two observations. The first is that Example 8 also illustrates a violation of the non-interference postulate for Pollock's 1987 system of logic-based argumentation with suppositions. Indeed, we made use of the identity rule in Example 8, the argument A_8 is not self-defeating, and the unique preferred extension corresponds to the unique grounded extension for Example 8. [10]

The second observation is that the subset rule poses a dilemma for Pollock-style accounts of suppositional reasoning. If valid, it leads to a failure of the ART and OR. If invalid, it leads to a failure of the non-interference property. Neither option seems desirable. It remains to investigate to what extent this dilemma generalizes to other accounts of non-monotonic reasoning in the presence of suppositions.

Acknowledgments Mathieu Beirlaen's research was funded by the Flemish Research Foundation (FWO-Vlaanderen). The research of Jesse Heyninck and Christian Straßer was sponsored by a Sofja Kovalevskaja award of the Alexander von Humboldt Foundation, funded by the German Ministry for Education and Research.

References

Arlo-Costa, H., and Egré, P. 2016. The logic of conditionals. In Zalta, E. N., ed., *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2016 edition.

Bennett, J. 2003. *A Philosophical Guide to Conditionals*. Oxford University Press.

Caminada, M.; Carnielli, W.; and Dunne, P. 2012. Semi-stable semantics. *Journal of Logic and Computation* 22(5):1207–1254.

Caminada, M. 2005. Collapse in formal argumentation systems. Technical report, Utrecht University.

Dung, P. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence* 77(2):321–357.

Jakobovits, H., and Vermeir, D. 1999. Robust semantics for argumentation frameworks. *Journal of Logic and Computation* 9(2):215–261.

Kraus, S.; Lehmann, D.; and Magidor, M. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44:167–207.

Makinson, D. 2003. Bridges between classical and nonmonotonic logic. *Logic Journal of the IGPL* 11:69–96.

[10] We made use of the dilemma rule **D** in the construction of some of the arguments in Example 8, but, as we pointed out above, the use of this rule can be circumvented by using the conditionalization rule **C** – see also Appendix A.

Modgil, S., and Prakken, H. 2013. A general account of argumentation with preferences. *Artificial Intelligence* 195:361–397.

Modgil, S., and Prakken, H. 2014. The ASPIC+ framework for structured argumentation: a tutorial. *Argument & Computation* 5(1):31–62.

Moinard, Y. 1994. Reasoning by cases without contraposition in default logic. In *Proceedings of the 11th European Conference on Artificial Intelligence*, ECAI'94, 381–385. New York, NY, USA: John Wiley & Sons, Inc.

Pollock, J. 1987. Defeasible reasoning. *Cognitive Science* 11(4):481–518.

Pollock, J. 1991a. A theory of defeasible reasoning. *International Journal of Intelligent Systems* 6:33–54.

Pollock, J. 1991b. Self-defeating arguments. *Minds and Machines* 1:367–392.

Pollock, J. 1994. Justification and defeat. *Artificial intelligence* 67:377–407.

Pollock, J. 1995. *Cognitive Carpentry. A Blueprint for How to Build a Person*. MIT Press.

Pollock, J. 2008. Defeasible reasoning. In Adler, J. E., and Rips, L. J., eds., *Reasoning. Studies of Human Inference and Its Foundations*. Cambridge University Press. 451–470.

Prakken, H., and Horty, J. 2012. An appreciation of John Pollock's work on the computational study of argument. *Argument and Computation* 3:1–19.

Ramsey, F. 1931. General propositions and causality. In Braithwaite, R., ed., *The Foundations of Arithmetic and Other Logical Essays*. London: Routledge. 237–255.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13:81–132.

Roos, N. 1998. Reasoning by cases in default logic. *Artificial Intelligence* 99(1):165–183.

Stalnaker, R. C. 1968. A theory of conditionals. In Rescher, N., ed., *Studies in Logical Theory (American Philosophical Quarterly* supplementary monograph series). Basil Blackwell, Oxford. 98–112.

Appendix

A Conditionalization and Dilemma

The procedure π recursively transforms an argument A into an argument A' without occurrences of the dilemma rule **D**.

1. if **D** was not used in A's construction, $\pi(A) = A$.
2. if $A = \langle B, C, D \rightsquigarrow \chi; \Gamma\rangle$,
   $\pi(A) = \langle \pi(B), \pi(C) \rightsquigarrow (\varphi \supset \chi), \pi(D) \rightsquigarrow (\psi \supset \chi) \rightarrow \chi; \Gamma\rangle$.
3. if $A = \langle A_1, \ldots, A_n \Rightarrow \chi; \Gamma\rangle$,
   $\pi(A) = \langle \pi(A_1), \ldots, \pi(A_n) \Rightarrow \chi; \Gamma\rangle$.
4. if $A = \langle A_1, \ldots, A_n \rightarrow \chi; \Gamma\rangle$,
   $\pi(A) = \langle \pi(A_1), \ldots, \pi(A_n) \rightarrow \chi; \Gamma\rangle$.
5. if $A = \langle B; \Gamma\rangle$, $\pi(A) = \langle \pi(B); \Gamma\rangle$.
6. if $A = \langle B \rightsquigarrow \varphi \supset \psi; \Gamma\rangle$, $\pi(A) = \langle \pi(B) \rightsquigarrow \varphi \supset \psi; \Gamma\rangle$.

The procedure μ recursively transforms an argument A into an argument A' without occurrences of the conditionalization rule **C**. Steps 1, 3, 4, and 5 of this procedure are as in π above, except that we replace occurrences of "π" with "μ" (in 1 we also replace "**D**" with "**C**"). For steps 2 and 6,

2. if $A = \langle B, C, D \rightsquigarrow \chi; \Gamma\rangle$,
   $\mu(A) = \langle \mu(B), \mu(C), \mu(D) \rightsquigarrow \chi; \Gamma\rangle$
6. if $A = \langle B \rightsquigarrow \varphi \supset \psi; \Gamma\rangle$,
   $\pi(A) = \langle\langle \rightarrow \psi \vee \neg\psi; \Gamma\rangle, \langle \mu(B) \rightarrow \psi \supset \chi; \Gamma \cup \{\psi\}\rangle,$
   $\langle\langle [\neg\psi]; \{\neg\psi\}\rangle \rightarrow \psi \supset \chi; \{\neg\psi\}\rangle \rightarrow \psi \supset \chi; \Gamma\rangle$

Where $\eta \in \{\pi, \mu\}$ it follows by an induction on the structure of $\eta(A)$ that (i) $\mathtt{Conc}(A) = \mathtt{Conc}(\eta(A))$ and (ii) the set of defeaters of A and $\eta(A)$ is identical. Consequently, A is justified iff $\eta(A)$ is justified.

B Foreign Adoptions and the Subset Rule

Let $\subseteq$-defeat be the defeat relation from Def. 3. Where $\dagger \in \{\subseteq, =\}$, A *directly* $\dagger$*-defeats* B iff $(\mathtt{Hyp}(A)\dagger\mathtt{Hyp}(B)$, **Rd** is the last rule applied in B's construction, and $\mathtt{Conc}(A) = -\mathtt{Conc}(B)$. Note that:

(i) If A $\dagger$-defeats B, then some $A' \in \mathtt{Sub}(A)$ directly $\dagger$-defeats some $B' \in \mathtt{Sub}(B)$.

(ii) If A directly $\subseteq$-defeats B, then $\langle A; \mathtt{Hyp}(B)\rangle$ directly =-defeats B.

Let $\mathtt{Def}_\subseteq$ [resp. $\mathtt{Def}_=$] denote the relation of $\subseteq$-defeat [resp. =-defeat]. Let $\mathcal{K}$ be any knowledge base, and $\mathcal{AF}_\subseteq = (\mathtt{Arg}(\mathcal{K}), \mathtt{Def}_\subseteq)$ while $\mathcal{AF}_= = (\mathtt{Arg}(\mathcal{K}), \mathtt{Def}_=)$.

Theorem 1 *$\mathcal{E}$ is a complete extension of $\mathcal{AF}_\subseteq$ iff $\mathcal{E}$ is a complete extension of $\mathcal{AF}_=$.*

Proof. ($\Rightarrow$) Suppose $\mathcal{E}$ is a complete extension of $\mathcal{AF}_\subseteq$. If $(A, B) \in \mathtt{Def}_=$ then $(A, B) \in \mathtt{Def}_\subseteq$, so either $A \notin \mathcal{E}$ or $B \notin \mathcal{E}$. This shows that $\mathcal{E}$ is conflict-free in $\mathcal{AF}_=$. It remains to show that (a) if $A \in \mathcal{E}$, then $\mathcal{E}$ defends A in $\mathcal{AF}_=$, and (b) if $\mathcal{E}$ defends A in $\mathcal{AF}_=$, then $A \in \mathcal{E}$. For (a), suppose $A \in \mathcal{E}$ and $(C, A) \in \mathtt{Def}_=$. Then $(C, A) \in \mathtt{Def}_\subseteq$ so there is a $B \in \mathcal{E}$ such that $(B, C) \in \mathtt{Def}_\subseteq$. By (i) and (ii), there are $B' \in \mathtt{Sub}(B)$ and $C' \in \mathtt{Sub}(C)$ such that $D = \langle B'; \mathtt{Hyp}(C')\rangle$ =-defeats B. Any =-defeater of D is a =-defeater of B, and so $D \in \mathcal{E}$ and A is defended in $\mathcal{AF}_=$. For (b), suppose $\mathcal{E}$ defends A in $\mathcal{AF}_=$ and $(C, A) \in \mathtt{Def}_\subseteq$. By (i) and (ii), there are $C' \in \mathtt{Sub}(C)$ and $A' \in \mathtt{Sub}(A)$ such that $E = \langle C'; \mathtt{Hyp}(A')\rangle$ =-defeats A. Hence there is a $B \in \mathcal{E}$ such that $(B, E) \in \mathtt{Def}_=$ and, by extension, $(B, E) \in \mathtt{Def}_\subseteq$. Thus B defends A in $\mathcal{AF}_\subseteq$ and $A \in \mathcal{E}$.
($\Leftarrow$) Suppose $\mathcal{E}$ is a complete extension of $\mathcal{AF}_=$. If $(A, B) \in \mathtt{Def}_\subseteq$ then, by (i) and (ii) there are $A' \in \mathtt{Sub}(A)$ and $B' \in \mathtt{Sub}(B)$ such that $C = \langle A'; \mathtt{Hyp}(B')\rangle$ =-defeats B. Thus either $C \notin \mathcal{E}$ or $B \notin \mathcal{E}$. $C \in \mathcal{E}$ iff $A' \in \mathcal{E}$ since any =-defeater of C is a =-defeater of A' and vice versa. Since $A' \in \mathtt{Sub}(A)$, it follows that either $A \notin \mathcal{E}$ or $B \notin \mathcal{E}$, which shows that $\mathcal{E}$ is conflict-free in $\mathcal{AF}_\subseteq$. It remains to show that (a') if $A \in \mathcal{E}$, then $\mathcal{E}$ defends A in $\mathcal{AF}_\subseteq$, and (b') if $\mathcal{E}$ defends A in $\mathcal{AF}_\subseteq$, then $A \in \mathcal{E}$. These arguments are roughly analogous to (a) and (b) above. The details are safely left to the reader. ■

On Iterated Contraction: syntactic characterization, representation theorem and limitations of the Levi identity*

Sébastien Konieczny
CRIL - CNRS
Université d'Artois
Lens, France
konieczny@cril.fr

Ramón Pino Pérez†
Facultad de Ciencias
Universidad de Los Andes
Mérida, Venezuela
pino@ula.ve

Abstract

In this paper we study iterated contraction in the epistemic state framework, offering a counterpart of the work of Darwiche and Pearl for iterated revision. We provide pure syntactical postulates for iterated contraction, that is, the postulates are expressed only in terms of the contraction operator. We establish a representation theorem for these operators. Our results allow to highlight the relationships between iterated contraction and iterated revision. In particular we show that iterated revision operators form a larger class than that of iterated contraction operators. As a consequence of this, in the epistemic state framework, the Levi identity has limitations; namely, it doesn't allow to define all iterated revision operators.

Introduction

Belief change theory (Alchourrón, Gärdenfors, and Makinson 1985; Gärdenfors 1988; Katsuno and Mendelzon 1991; 1992; Hansson 1999; Fermé and Hansson 2011) aims at modelling the evolution of the logical beliefs of an agent according to new inputs the agent receives.

The two main classes of operators are revision operators, which allow to correct some wrong beliefs of the agent, and contraction operators, which allow to remove some pieces of beliefs from the beliefs of the agent.

Contraction and revision, though being different processes, are closely linked. Two *identities* allow to define contraction from revision and vice-versa. One can define a revision operator from a contraction operator by the *Levi identity*, which states that, in order to define a revision by α, one can first perform a contraction by $\neg\alpha$ and then an expansion[1] by α (Levi 1977). Conversely, one can define a contraction operator from a revision operator by using the *Harper identity*: what is true after contraction by α is what is true in the current state and in the result of the revision by $\neg\alpha$ (Harper 1977). To give the formal definition of these identities, let us denote by K a theory (a deductively closed set of logical sentences), and let α be a formula. Let us denote $\star$ a revision operator, $\div$ a contraction operator, and $\oplus$ the expansion:

Levi identity	$K \star \alpha = (K \div \neg\alpha) \oplus \alpha$
Harper identity	$K \div \alpha = K \cap (K \star \neg\alpha)$

The connection obtained through these identities is very strong, since one obtains in fact a bijection between the set of revision operators and the set of contraction operators (Gärdenfors 1988). So, in the AGM framework these two classes of operators are two sides of the same coin, and one can study either revision or contraction, depending on which operator is chosen as more basic/natural.

Although intrinsically a dynamic process, initial works on belief change only address the (static) one-step change (Alchourrón, Gärdenfors, and Makinson 1985; Gärdenfors 1988; Katsuno and Mendelzon 1991), and were not able to cope with iterated change.

After many unsuccessful attempts, a solution for modelling iterated revision was provided by Darwiche and Pearl (Darwiche and Pearl 1997). They provide additional postulates to govern iterated change. These additional constraints for iteration cannot use simple logical theories for belief representation, as used in the one-step case. One has to shift to a more powerful representation, epistemic states, which allow to encode the revision strategy of the agent, and allow to ensure dynamic coherence of changes.

It is interesting to note that the work of Darwiche and Pearl was dedicated to iterated revision. One could expect that a characterization of iterated contraction could be obtained safely from generalizations of the identities. In fact this is not the case. First, until now there is no proposal for postulates nor representation theorem for iterated contraction. There were some works on iterated contraction that we will discuss in the related work section, but no real counterpart of the work of Darwiche and Pearl for contraction existed so far. This is what we propose in this paper.

The class of iterated contraction operators obtained is very interesting in different respects. It allows to obtain a better understanding of belief change theory. Actually, some of the consequences of our representation theorem are quite surprising.

*This paper has been published in the proceedings of the 11th International Conference on Scalable Uncertainty Management (SUM'17).

†Current address: Yachay Tech University, School of Mathematics and Computer Science, Urcuquí, Ecuador. E-mail: rpino@yachaytech.edu.ec

[1] See (Gärdenfors 1988) for exact definition, but one can safely identify expansion with conjunction/union in most cases.

First, when comparing the two representation theorems (the one for iterated contraction and the one for iterated revision), it is clear that they share the same class of (faithful) assignments. This means that the difference between iterated contraction and iterated revision is not a matter of nature, but a matter of degree (revision being a bigger change than contraction, not a different kind of change). This also means that there is a deep relationship between iterated contraction and iterated revision via the representation theorems and the assignments. One could think that the generalization of the Levi and Harper identities could be easily attained thanks to these theorems. Nevertheless, it is not the case.

More surprisingly, we prove that there are more iterated revision operators than contraction operators (oppositely to the bijection obtained in the classical - AGM - framework). As a consequence, one can not expect to have generalizations of the Levi identity for the iterated case: some iterated revision operators are out of reach from iterated contraction ones. This seems to suggest that iterated belief revision operators are more basic than iterated belief contraction ones.

In the next Section we will provide the formal preliminaries for this paper. Then we give the logical postulates for modelling iterated contraction. After that we provide a representation theorem for these contractions. Then we study the links between iterated contraction and iterated revision. Finally we discuss some related work.

Preliminaries

We consider a propositional language $\mathcal{L}$ defined from a finite set of propositional variables $\mathcal{P}$ and the standard connectives. Let $\mathcal{L}^*$ denote the set of consistent formulae of $\mathcal{L}$.

An interpretation ω is a total function from $\mathcal{P}$ to $\{0,1\}$. The set of all interpretations is denoted by $\mathcal{W}$. An interpretation ω is a model of a formula $\phi \in \mathcal{L}$ if and only if it makes it true in the usual truth functional way. $[\![\alpha]\!]$ denotes the set of models of the formula α, i.e., $[\![\alpha]\!] = \{\omega \in \mathcal{W} \mid \omega \models \alpha\}$. $\vdash$ denotes implication between formulae, i.e. $\alpha \vdash \beta$ means $[\![\alpha]\!] \subseteq [\![\beta]\!]$.

$\leq$ denotes a pre-order on $\mathcal{W}$ (i.e., a reflexive and transitive relation), and $<$ denotes the associated strict order defined by $\omega < \omega'$ if and only if $\omega \leq \omega'$ and $\omega' \not\leq \omega$. A pre-order is *total* if for all $\omega, \omega' \in \mathcal{W}$, $\omega \leq \omega'$ or $\omega' \leq \omega$. If $A \subseteq \mathcal{W}$, then the set of minimal elements of A with respect to a total pre-order $\leq$, denoted by $\min(A, \leq)$, is defined by $\min(A, \leq) = \{\omega \in A \mid \nexists \omega' \in A$ such that $\omega' < \omega\}$.

We will use epistemic states to represent the beliefs of the agent, as usual in iterated belief revision (Darwiche and Pearl 1997). An epistemic state Ψ represents the current beliefs of the agent, but also additional conditional information guiding the revision process (usually represented by a pre-order on interpretations, a set of conditionals, a sequence of formulae, etc). Let $\mathcal{E}$ denote the set of all epistemic states. A projection function $B : \mathcal{E} \longrightarrow \mathcal{L}^*$ associates to each epistemic state Ψ a consistent formula $B(\Psi)$, that represents the current beliefs of the agent in the epistemic state Ψ. We will call models of the epistemic state the models of its beliefs, i.e. $[\![\Psi]\!] = [\![B(\Psi)]\!]$.

A concrete and very useful representation of epistemic states are total pre-orders over interpretations. In this representation, if $\Psi = \leq$, then $B(\Psi)$ is a propositional formula which satisfies $[\![B(\Psi)]\!] = \min(\mathcal{W}, \leq)$. We call this concrete representation of epistemic states the *canonical representation*.

For simplicity purpose we will only consider in this paper consistent epistemic states and consistent new information. Thus, we consider change operators as functions $\circ$ mapping an epistemic state and a consistent formula into a new epistemic state, i.e. in symbols, $\circ : \mathcal{E} \times \mathcal{L}^* \longrightarrow \mathcal{E}$. The image of a pair (Ψ, α) under $\circ$ will be denoted by $\Psi \circ \alpha$.

Let us now recall Darwiche and Pearl proposal for iterated revision (Darwiche and Pearl 1997). Darwiche and Pearl modified the list of KM postulates (Katsuno and Mendelzon 1991) to work in the more general framework of epistemic states:

(R*1) $B(\Psi \star \alpha) \vdash \alpha$

(R*2) If $B(\Psi) \wedge \alpha \nvdash \bot$ then $B(\Psi \star \alpha) \equiv \varphi \wedge \alpha$

(R*3) If $\alpha \nvdash \bot$ then $B(\Psi \star \alpha) \nvdash \bot$

(R*4) If $\Psi_1 = \Psi_2$ and $\alpha_1 \equiv \alpha_2$ then $B(\Psi_1 \star \alpha_1) \equiv B(\Psi_2 \star \alpha_2)$

(R*5) $B(\Psi \star \alpha) \wedge \psi \vdash B(\Psi \star (\alpha \wedge \psi))$

(R*6) If $B(\Psi \star \alpha) \wedge \psi \nvdash \bot$ then $B(\Psi \star (\alpha \wedge \psi)) \vdash B(\Psi \star \alpha) \wedge \psi$

For the most part, the DP list is obtained from the KM list by replacing each φ by $B(\Psi)$ and each $\varphi \star \alpha$ by $B(\Psi \star \alpha)$. The only exception to this is (R*4), which is stronger than its simple translation.

In addition to this set of basic postulates, Darwiche and Pearl proposed a set of postulates devoted to iteration:

(DP1) If $\alpha \vdash \mu$ then $B((\Psi \star \mu) \star \alpha) \equiv B(\Psi \star \alpha)$

(DP2) If $\alpha \vdash \neg\mu$ then $B((\Psi \star \mu) \star \alpha) \equiv B(\Psi \star \alpha)$

(DP3) If $B((\Psi \star \alpha) \vdash \mu$ then $B((\Psi \star \mu) \star \alpha) \vdash \mu$

(DP4) If $B((\Psi \star \alpha) \nvdash \neg\mu$ then $B((\Psi \star \mu) \star \alpha) \nvdash \neg\mu$

And then they give a representation theorem in terms of pre-orders on interpretations:

Definition 1 *A faithful assignment is a mapping that associates to any epistemic state Ψ a total pre-order $\leq_\Psi$ on interpretations such that:*

1. *If $\omega \models B(\Psi)$ and $\omega' \models B(\Psi)$, then $\omega \simeq_\Psi \omega'$*

2. *If $\omega \models B(\Psi)$ and $\omega' \not\models B(\Psi)$, then $\omega <_\Psi \omega'$*

3. *If $\Psi_1 = \Psi_2$, then $\leq_{\Psi_1} = \leq_{\Psi_2}$*

Theorem 1 ((Darwiche and Pearl 1997)) *An operator $\star$ satisfies (R*1)-(R*6) if and only if there is a faihtful assignment that maps each epistemic state Ψ to a total pre-order on interpretations $\leq_\Psi$ such that:*

$$[\![\Psi \star \mu]\!] = \min([\![\mu]\!], \leq_\Psi)$$

Theorem 2 ((Darwiche and Pearl 1997)) *Let $\star$ be a revision operator that satisfies (R*1)-(R*6). This operator satisfies (DP1)-(DP4) if and only if this operator and its faithful assignment satisfies :*

(CR1) *If $\omega \models \mu$ and $\omega' \models \mu$, then $\omega \leq_\Psi \omega' \Leftrightarrow \omega \leq_{\Psi\star\mu} \omega'$*

(CR2) *If $\omega \models \neg\mu$ and $\omega' \models \neg\mu$, then $\omega \leq_\Psi \omega' \Leftrightarrow \omega \leq_{\Psi\star\mu} \omega'$*

(CR3) *If $\omega \models \mu$ and $\omega' \models \neg\mu$, then $\omega <_\Psi \omega' \Rightarrow \omega <_{\Psi\star\mu} \omega'$*

(CR4) *If* $\omega \models \mu$ *and* $\omega' \models \neg\mu$,
then $\omega \leq_\Psi \omega' \Rightarrow \omega \leq_{\Psi\star\mu} \omega'$

The first aim of this paper is to provide a similar direct characterization of iterated contraction. Then we will study the links between iterated revision and iterated contraction operators.

Iterated Contraction

Let us first give the basic postulates for contraction of epistemic states. We use the contraction postulates given for propositional logic formulas in (Caridroit, Konieczny, and Marquis 2015), that are equivalent to the original AGM ones (Alchourrón, Gärdenfors, and Makinson 1985), that we only adapt for epistemic states:

(C1) $B(\Psi) \vdash B(\Psi - \alpha)$

(C2) If $B(\Psi) \nvdash \alpha$, then $B(\Psi - \alpha) \vdash B(\Psi)$

(C3) If $B(\Psi - \alpha) \vdash \alpha$, then $\vdash \alpha$

(C4) $B(\Psi - \alpha) \wedge \alpha \vdash B(\Psi)$

(C5) If $\alpha_1 \equiv \alpha_2$ then $B(\Psi - \alpha_1) \equiv B(\Psi - \alpha_2)$

(C6) $B(\Psi - (\alpha \wedge \beta)) \vdash B(\Psi - \alpha) \vee B(\Psi - \beta)$

(C7) If $B(\Psi - (\alpha \wedge \beta)) \nvdash \alpha$, then $B(\Psi - \alpha) \vdash B(\Psi - (\alpha \wedge \beta))$

(C1) states that contraction can just remove some information, so the beliefs of the posterior epistemic state are weaker than the beliefs of the prior one. (C2) says that if the epistemic state does not imply the formula by which one wants to contract, then the posterior epistemic state will have the same beliefs as the prior one. (C3) is the success postulate, it states that the only case where contraction fails to remove a formula from the beliefs of the agent is when this formula is a tautology. (C4) is the recovery postulate, it states that if we do the contraction by a formula followed by a conjunction by this formula, then we will recover the initial beliefs. This ensures that no unnecessary information is discarded during the contraction. (C5) is the irrelevance of syntax postulate, that says that the syntax does not have any impact on the result of the contraction. (C6) says that the contraction by a conjunction implies the disjunction of the contractions by the conjuncts. (C7) says that if α is not removed during the contraction by the conjunction by $\alpha \wedge \beta$, then the contraction by α implies the contraction by the conjunction.

Now let us introduce the postulates for iterated contraction:

(C8) If $\neg\alpha \vdash \gamma$ then $B(\Psi - (\alpha \vee \beta)) \vdash B(\Psi - \alpha)$
$\Leftrightarrow B(\Psi - \gamma - (\alpha \vee \beta)) \vdash B(\Psi - \gamma - \alpha)$

(C9) If $\gamma \vdash \alpha$ then $B(\Psi - (\alpha \vee \beta)) \vdash B(\Psi - \alpha)$
$\Leftrightarrow B(\Psi - \gamma - (\alpha \vee \beta)) \vdash B(\Psi - \gamma - \alpha)$

(C10) If $\neg\beta \vdash \gamma$ then $B(\Psi - \gamma - (\alpha \vee \beta)) \vdash B(\Psi - \gamma - \alpha)$
$\Rightarrow B(\Psi - (\alpha \vee \beta)) \vdash B(\Psi - \alpha)$

(C11) If $\gamma \vdash \beta$ then $B(\Psi - \gamma - (\alpha \vee \beta)) \vdash B(\Psi - \gamma - \alpha)$
$\Rightarrow B(\Psi - (\alpha \vee \beta)) \vdash B(\Psi - \alpha)$

(C8) expresses the fact that if a contraction by a disjunction implies the contraction by one of the disjuncts, then it will be the same if we first contract by a formula that is a consequence of the negation of that disjunct. (C9) captures the fact that if a contraction by a disjunction implies the contraction by one of the disjuncts, then it will be the same if we first contract by a formula that implies this disjunct. (C10) expresses the fact that if a contraction by a disjunction implies the contraction by one of the disjunct after a contraction by a formula that is a consequence of the negation of the other disjunct, then it was already the case before this contraction. (C11) captures the fact that if a contraction by a disjunction implies the contraction by one of the disjunct after a contraction by a formula that implies the other disjunct, then it was already the case before this contraction.

The operators satisfying (C1)-(C7) will be called contraction operators. The operators satisfying (C1)-(C11) will be called iterated contraction operators.

Representation Theorem

Let us now provide a representation theorem for iterated contraction operators in terms of faithful assignments. These assignments associate to each epistemic state a total pre-order on interpretations, this total pre-order representing the relative plausibility of each interpretation, and the current beliefs of the agent being the most plausible ones.

And let us now state the basic theorem for contraction postulates in the epistemic state framework.

Theorem 3 *An operator* $-$ *satisfies the postulates* ***(C1) - (C7)*** *if and only if there exists a faithful assignment that associates to each epistemic state* Ψ *a total pre-order* $\leq_\Psi$ *on interpretations such that*

$$[\![\Psi - \alpha]\!] = [\![\Psi]\!] \cup \min([\![\neg\alpha]\!], \leq_\Psi)$$

We will say that the faithful assignment given by the previous theorem *represents* the operator $-$. The proof of this theorem follows the same lines as the proof for the representation theorem in the propositional case (Caridroit, Konieczny, and Marquis 2015). For space reason it will not be included here. We concentrate our effort in proving a representation theorem for iterated contraction operators.

Let us now state the representation theorem for iterated contraction.

Theorem 4 *Let* $-$ *be a contraction operator that satisfies* ***(C1) - (C7)****. This operator satisfies* ***(C8) - (C11)*** *if and only if this operator and its faifhtul assigment satisfies:*

4. *If* $\omega, \omega' \in [\![\gamma]\!]$ *then* $\omega \leq_\Psi \omega' \Leftrightarrow \omega \leq_{\Psi-\gamma} \omega'$

5. *If* $\omega, \omega' \in [\![\neg\gamma]\!]$ *then* $\omega \leq_\Psi \omega' \Leftrightarrow \omega \leq_{\Psi-\gamma} \omega'$

6. *If* $\omega \in [\![\neg\gamma]\!]$ *and* $\omega' \in [\![\gamma]\!]$ *then* $\omega <_\Psi \omega' \Rightarrow \omega <_{\Psi-\gamma} \omega'$

7. *If* $\omega \in [\![\neg\gamma]\!]$ *and* $\omega' \in [\![\gamma]\!]$ *then* $\omega \leq_\Psi \omega' \Rightarrow \omega \leq_{\Psi-\gamma} \omega'$

We will call a faithful assignment that satisfies properties 4 *to* 7 *an iterated faithful assignment.*

Condition 4 captures the fact that the plausibility between the models of γ is exactly the same before the contraction and after the contraction by γ. Condition 5 captures the fact that the plausibility between the models of $\neg\gamma$ is exactly

the same before the contraction and after the contraction by γ. Conditions 4 and 5 will be called *rigidity conditions* (called also ordered preservation conditions in (Ramachandran, Nayak, and Orgun 2012)). Condition 6 captures the fact that if a model of $\neg\gamma$ is strictly more plausible than a model of γ before the contraction by γ then it will be the case after the contraction by γ. Condition 7 captures the fact that the plausibility of the models of $\neg\gamma$ does not decrease with respect the models of γ after contraction by γ. More precisely, if a model of $\neg\gamma$ is at least as plausible as a model of γ before the contraction it will be the case after the contraction by γ. Conditions 6 and 7 will be called *non-worsening conditions*.

Please note that the iterated faithful assignments are directly related to the ones for iterated revision (cf. Theorem 2), there are only some inversions in conditions 6 and 7 (compared to CR3 and CR4), that are due to the fact that one can see a contraction by $\neg\alpha$ as a softer change (improvement (Konieczny and Pino Pérez 2008)) than a revision by α (see discussion at the beginning of Section).

For space reasons we only give a sketch of the proof of Theorem 4. An iterated contraction operator is indeed a contraction operator. Thus, we know by Theorem 3 that there is a faithful assignment representing it. Thus the proof of Theorem 4 will consist in proving that under the assumption of the basic contraction postulates, the iteration postulates entail conditions 4-7 for the faithful assignment representing the operator. And reciprocally, that if the iterated faithful assignment represents the operator the postulates of the iteration are satisfied. Thus, from now on, in this section we suppose that $-$ is a contraction operator and $\Psi \mapsto \leq_\Psi$ is the faithful assignment representing it, that is the equation in Theorem 3 holds. Actually, we prove the following facts which are enough to conclude:

(i) The assignment satisfies condition 4 if and only if postulate (C8) holds.

(ii) The assignment satisfies condition 5 if and only if postulate (C9) holds.

(iii) Suppose the contraction operator satisfies (C8). Then the assignment satisfies condition 6 if and only if postulate (C10) holds.

(iv) The assignment satisfies condition 7 if and only if postulate (C11) holds.

In order to give a flavor of the whole proof we give the proof of the Fact (iii) (which is perhaps a little more complicated than the other three facts). First we have the following observations:

Observation 1 *Suppose that the assignment satisfies condition 4, then for any μ such that $\mu \vdash \gamma$ we have* $\min([\![\mu]\!], \leq_\Psi) = \min([\![\mu]\!], \leq_{\Psi-\gamma})$.

Observation 2 *Condition 6 is equivalent to the following condition:*

6'. *If $\omega \in [\![\neg\gamma]\!]$ and $\omega' \in [\![\gamma]\!]$ then $\omega' \leq_{\Psi-\gamma} \omega \Rightarrow \omega' \leq_\Psi \omega$*

Now we proceed to prove (iii). Note that, since α, β and γ are any formulas, using (C5), the postulate (C10) can be rewritten as follows:

If $\beta \vdash \gamma$ then $B((\Psi - \gamma) - \neg(\alpha \wedge \beta)) \vdash B((\Psi - \gamma) - \neg\alpha) \Rightarrow B(\Psi - \neg(\alpha \wedge \beta)) \vdash B(\Psi - \neg\alpha)$

First we prove that postulate (C10) entails condition 6 of an iterated assignment. By Observation 2, it is enough to prove that (C10) entails 6'. Thus, assume (C10) holds. Suppose $\omega \in [\![\neg\gamma]\!]$, $\omega' \in [\![\gamma]\!]$ and $\omega' \leq_{\Psi-\gamma} \omega$. We want to show that $\omega' \leq_\Psi \omega$.

Let α and β be formulas such that $[\![\alpha]\!] = \{\omega, \omega'\}$ and $[\![\beta]\!] = \{\omega'\}$. Note that $\{\omega'\} \subseteq \min([\![\alpha]\!], \leq_{\Psi-\gamma})$ because $\omega' \leq_{\Psi-\gamma} \omega$. We have also $\min([\![\alpha \wedge \beta]\!], \leq_{\Psi-\gamma}) = \{\omega'\}$. Then, we have $[\![(\Psi - \gamma) - \neg(\alpha \wedge \beta)]\!] = [\![\Psi - \gamma]\!] \cup \min([\![\alpha \wedge \beta]\!], \leq_{\Psi-\gamma})$; the last expression is equal to $[\![\Psi - \gamma]\!] \cup \{\omega'\}$ which is a subset of $[\![\Psi - \gamma]\!] \cup \min([\![\alpha]\!], \leq_{\Psi-\gamma})$. But this last expression is $[\![(\Psi - \gamma) - \neg\alpha]\!]$. So, $[\![(\Psi - \gamma) - \neg(\alpha \wedge \beta)]\!] \subseteq [\![(\Psi - \gamma) - \neg\alpha]\!]$, that is $B((\Psi - \gamma) - \neg(\alpha \wedge \beta)) \vdash B((\Psi - \gamma) - \neg\alpha)$. Then, by (C10), we have $B(\Psi - \neg(\alpha \wedge \beta)) \vdash B(\Psi - \neg\alpha)$, that is $[\![\Psi]\!] \cup \{\omega'\} \subseteq [\![\Psi]\!] \cup \min(\{\omega, \omega'\}, \leq_\Psi)$. Therefore, $\omega' \in [\![\Psi]\!]$ or $\omega' \in \min(\{\omega, \omega'\}, \leq_\Psi)$. In both cases we get $\omega' \leq_\Psi \omega$ (In the first case we use the fact that $\leq_\Psi$ is a faithful assignment, in particular $[\![\Psi]\!] = \min(\mathcal{W}, \leq_\Psi)$).

Now we prove that Condition 6 entails Postulate (C10). Assume that $\beta \vdash \gamma$. We suppose that $\alpha \wedge \beta \nvdash \bot$ (the other case is trivial because the contraction by a tautology doesn't change the beliefs). Suppose $B((\Psi - \gamma) - \neg(\alpha \wedge \beta)) \vdash B((\Psi - \gamma) - \neg\alpha)$, that is

$$[\![(\Psi - \gamma) - \neg(\alpha \wedge \beta)]\!] \subseteq [\![(\Psi - \gamma) - \neg\alpha]\!] \quad (1)$$

We want to show that $B(\Psi - \neg(\alpha \wedge \beta)) \vdash B(\Psi - \neg\alpha)$, that is to say

$$[\![\Psi - \neg(\alpha \wedge \beta)]\!] \subseteq [\![\Psi - \neg\alpha]\!] \quad (2)$$

By Theorem 3, equation (1) and equation (2) can be rewritten, respectively, as

$$[\![\Psi - \gamma]\!] \cup \min([\![\alpha \wedge \beta]\!], \leq_{\Psi-\gamma}) \subseteq [\![\Psi - \gamma]\!] \cup \min([\![\alpha]\!], \leq_{\Psi-\gamma}) \quad (3)$$

and

$$[\![\Psi]\!] \cup \min([\![\alpha \wedge \beta]\!], \leq_\Psi) \subseteq [\![\Psi]\!] \cup \min([\![\alpha]\!], \leq_\Psi) \quad (4)$$

First we are going to prove that (3) entails

$$\min([\![\alpha \wedge \beta]\!], \leq_{\Psi-\gamma}) \subseteq \min([\![\alpha]\!], \leq_{\Psi-\gamma}) \quad (5)$$

In order to see that, take $\omega \in \min([\![\alpha \wedge \beta]\!], \leq_{\Psi-\gamma})$. Then, by Equation (3), we have either $\omega \in [\![\Psi - \gamma]\!]$ or $\omega \in \min([\![\alpha]\!], \leq_{\Psi-\gamma})$. In the first case, by the fact of having a faithful assignment, $\omega \in \min(\mathcal{W}, \leq_{\Psi-\gamma})$ and therefore $\omega \in \min([\![\alpha]\!], \leq_{\Psi-\gamma})$. In the second case is trivial. Thus, in any case we have 5.

Now, towards a contradiction, suppose that (4) doesn't hold. Thus, there exists $\omega \in \min([\![\alpha \wedge \beta]\!], \leq_\Psi)$ such that $\omega \notin \min([\![\alpha]\!], \leq_\Psi)$. So there is $\omega' \in \min([\![\alpha]\!], \leq_\Psi)$ such that

$$\omega' <_\Psi \omega \quad (6)$$

Note that $\omega \models \beta$, and, by hypothesis, $\beta \vdash \gamma$, thus $\omega \models \gamma$. We are going to consider the following two cases: $\omega' \models \gamma$ and $\omega' \models \neg\gamma$

Suppose we are in the first case, *i.e.* $\omega' \models \gamma$. Then, by Condition 4 (equivalent to our assumption of Postulate (C8)), $\omega' <_{\Psi-\gamma} \omega$. Now, suppose we are in the second case, *i.e.* $\omega' \models \neg\gamma$. Then, because $\omega \models \gamma$ and (6), by Condition 6, $\omega' <_{\Psi-\gamma} \omega$. Thus in any case we have

$$\omega' <_{\Psi-\gamma} \omega \tag{7}$$

Since $\alpha \wedge \beta \vdash \gamma$, by Observation 1, $\omega \in \min([\![\alpha \wedge \beta]\!], \leq_{\Psi-\gamma})$. Then, by (5), $\omega \in \min([\![\alpha]\!], \leq_{\Psi-\gamma})$. But this is a contradiction with (7).

Iterated Contraction vs Iterated Revision

We would like to investigate now the relationship between iterated contraction and iterated revision.

A natural tendency would be to try to generalize Levi and Harper Identity to the iterated case. In fact some related works followed this path (Nayak et al. 2006; Chopra et al. 2008; Booth and Chandler 2016).

In the following we will first argue and show that it is not so simple. We will also show that there are some problems when one follows this way for connecting iterated contraction and iterated revision. Actually, we will show that in the iterated case, they are not two sides of a same coin (i.e. two classes of operators linked by a bijection), but that they rather are two instances of a same kind of change operators, and that the link and difference is just a matter of degree of change.

Identities in the General Case

Let us first recall the Levy and Harper Identities:

Levi identity	$\Psi \star \alpha = (\Psi \div \neg\alpha) \oplus \alpha$
Harper identity	$\Psi \div \alpha = \Psi \sqcap (\Psi \star \neg\alpha)$

Let us note the problems of using these identities for iterated contraction and revision in the epistemic state framework. First, in the AGM case, these two identities are definitional, that means that, for instance, using Levi identity one can obtain the revision operator $\star$ that defines the theory $\Psi\star\alpha$ from the right side of the identity using the contraction and expansion operators. But in our general framework, epistemic states are abstract objects, which can only be apprehended at the logical level from the projection function B. Thus, in this general framework, we do not fully know what $\Psi \div \neg\alpha$ is, and so we can not use it to define what should be $\Psi\star\alpha$. So using a definitional equality here is difficult. The only way to proceed seems to be choosing a particular representation of epistemic states to work with (such as total preorders over the interpretations), but then the results are given on this representation and not in the general case. Second, whereas $\oplus$ and $\sqcap$ have a clear meaning in the AGM framework, one has to figure out a definition in the epistemic state framework. That is by itself a non-trivial task (studying the possible definitions of $\sqcap$ is one of the main aims of (Booth and Chandler 2016)).

A possibility would be to restrict these identities to the beliefs of the epistemic states, as :

Belief Levi equivalence	$B(\Psi \star \alpha) \equiv B(\Psi \div \neg\alpha) \oplus \alpha$
Belief Harper equivalence	$B(\Psi \div \alpha) \equiv B(\Psi) \vee B(\Psi \star \neg\alpha)$

But we do not have identities anymore, but only equivalences that are not definitional. So one has to first identify two operators $\star$ and $\div$ and check that they are linked through these equivalences.

Identities under the Canonical Representation

So to go further we have to commit to a particular representation of epistemic states. The canonical one, using total preorders (described in Section), can be used together with the faithful assignment, to define completely the new epistemic state after contraction (or revision). Thus, suppose that we have a faithful assignment $\Psi \mapsto \leq_\Psi$. We identify Ψ with $\leq_\Psi$, and we define $\leq_{\Psi-\gamma}$ satisfying the properties (4-7) of Theorem 4. Then, by Theorem 4, the operator defined by $\Psi - \gamma = \leq_{\Psi-\gamma}$ is an iterated contraction operator. In such a case we say that the operator $-$ is *given by* the assignment. We can proceed, in the same way when the assignment satisfies the requirements of an iterated assignment (for revision (Darwiche and Pearl 1997)) and then the operator defined by $\Psi \star \alpha = \leq_{\Psi\star\alpha}$ is an iterated revision operator.

Thus, one can restate the identities on the total pre-orders associated to the epistemic states by the operators[2]:

Tpo Levi identity	$\leq_{\Psi\star\alpha} = \leq_{\Psi\div\neg\alpha} \oplus\alpha$
Tpo Harper identity	$\leq_{\Psi\div\alpha} = \leq_\Psi \sqcap \leq_{\Psi\star\neg\alpha}$

So now we can define these pre-orders using the identities and check that we correctly obtain operators with the expected properties. The only remaining problem is to define the operators $\oplus$ and $\sqcap$ in this setting.

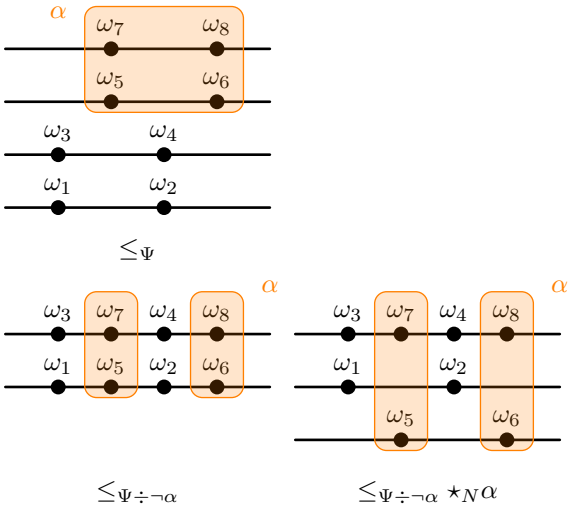


Figure 1: From Contraction to Revision

As for $\oplus$ let us show that using Boutilier natural revision operator $\star_N$ (Boutilier 1996) is a correct option, in the sense that using this operator as $\oplus$ we obtain a DP (Darwiche and Pearl (Darwiche and Pearl 1997)) revision operator (see Proposition 1).

Let us recall the definition of this operator on total preorders, that amounts to look at the most plausible models

[2]Tpo means Total pre-order.

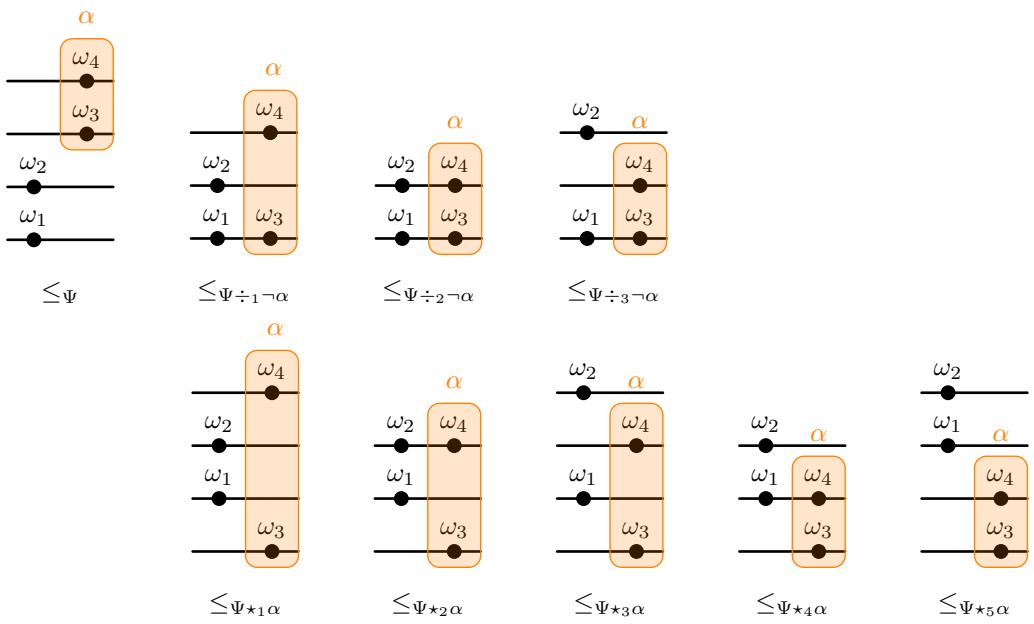


Figure 2: All possible contractions by $\neg\alpha$ and revisions by α from $\leq_\Psi$

of the new piece of information and define them as the new most plausible interpretations while nothing else changes: Let $\leq_\Psi$ be the pre-order associated to the epistemic state Ψ by the faithful assignment, and let α be the new piece of information, then $\leq_{\Psi \star_N \alpha}$ (we will also use the equivalent notation $\leq_\Psi \star_N \alpha$) is defined as:

- If $\omega \models \min([\![\alpha]\!], \leq_\Psi)$ and $\omega' \not\models \min([\![\alpha]\!], \leq_\Psi)$, then $\omega <_{\Psi \star_N \alpha} \omega'$
- In all the other cases $\omega \leq_{\Psi \star_N \alpha} \omega'$ iff $\omega \leq_\Psi \omega'$

Then one can show that the Tpo Levi identity holds for the iterated case:

Proposition 1 *Let $\div$ be an iterated contraction operator given by its assignment $\Psi \mapsto \leq_\Psi$. Then the assignment defined by $\leq_{\Psi \star \alpha} = \leq_{\Psi \div \neg\alpha} \star_N \alpha$ satisfies properties (CR1)-(CR4), and can be used to define a Darwiche and Pearl iterated revision operator in the framework of the canonical representation of epistemic states.*

This proposition implies in particular that to each iterated contraction operator one can associate a corresponding iterated revision operator. So, this means that the cardinality of the class of iterated revision operators obtained via the Tpo Levi identity is at least equal to the cardinality of the class of iterated contraction operators. Note that this observation does not depend on the interpretation of the symbol $\oplus$ utilized.

The following example illustrates the use of our concrete Tpo Levi identity.

Example 1 *Let us consider the total pre-order $\leq_\Psi$ represented in Figure 1. So in that figure $[\![\leq_\Psi]\!] = \{\omega_1, \omega_2\}$ and $[\![\alpha]\!] = \{\omega_5, \omega_6, \omega_7, \omega_8\}$. In this Figure the lower an interpretation is, the more plausible it is. For instance in $\leq_\Psi$ we have that $\omega_1 <_\Psi \omega_3$. An iterated contraction by $\neg\alpha$ is a change that increases (improves) the plausibility of the models of α, with the condition that the most plausible models of α in $\leq_\Psi$ joins (become as plausible as) the most plausible models of $\leq_\Psi$ (We give on such possibility for $\leq_{\Psi \div \neg\alpha}$). The relation between the models of α doesn't change after contraction and nor does the relation between the models of $\neg\alpha$. From this, to define a revision, one can just select these most plausible models of α and take them as the most plausible models using Boutilier's natural revision ($\leq_{\Psi \div \neg\alpha} \star_N \alpha$).*

The converse process, that is, defining iterated contraction operators starting from iterated revision operators using the Tpo Harper identity, requires in particular to find a correct definition for $\sqcap$. This problem is investigated by Booth and Chandler (Booth and Chandler 2016), where they show that there is not a single, canonical way to proceed.

One can see this as an additional richness of the iterated framework. However, this richness of the epistemic state representation has its counterparts. In fact in the iterated case there are more revision operators than contraction ones. More precisely we have the following result:

Theorem 5 *There are more iterated revision operators than iterated contraction operators. In particular, this entails that it is impossible to find an interpretation of the expansion $\oplus$ in the Tpo Levi identity in order to obtain all the iterated revision operators via this identity.*

Proof: The fact that there are no more contraction operators than revision operators is a direct consequence of proposition 1. To show that they are more revision operators than contraction operators consider $\leq_\Psi$ and α as illustrated in Figure 2. Due to rigidity conditions for iterated contraction, there are only three possible different outputs as results of contraction of $\leq_\Psi$ by $\neg\alpha$ ($\leq_{\Psi \div_1 \neg\alpha}$, $\leq_{\Psi \div_2 \neg\alpha}$ and

$\leq_{\Psi \div_3 \neg\alpha}$). Contrastingly there are five different possible output for revision. That is due to the rigidity postulates for iterated revision. Three of these possible revision outputs can be obtained from contraction outputs using natural revision as in the previous example ($\leq_{\Psi \star_1 \alpha}$, $\leq_{\Psi \star_2 \alpha}$, $\leq_{\Psi \star_3 \alpha}$). The other two ones are $\leq_{\Psi \star_4 \alpha}$ and $\leq_{\Psi \star_5 \alpha}$, that are not related to any contraction result using the identity. ■

The previous theorem is important because it tells us that in the iterated case the Levy identity has limitations. This theorem is also important since it gives us a true distinction between classical AGM framework and the iterated framework. In the classical AGM framework there is a bijection between revision and contraction operators. Contraction is often considered as a more fundamental operator since a revision can be defined, through Levi identity, as a contraction followed by a conjunction (expansion). In the iterated case there are more iterated revision than iterated contraction operators, so the more general change operator seems to be revision.

One can object that $\leq_{\Psi \star_4 \alpha}$ and $\leq_{\Psi \star_5 \alpha}$ could maybe be obtained through Levi identity by using another definition of $\oplus$ than Boutilier's natural revision operator, as we used here. But this does not change the fact that there is only three possible contraction results versus five possible revision results, and then with any alternative, there is still no way to define a bijection. We can just define a relation between $\div$ and a couple $(\star, \oplus)$, that is far from AGM original idea of this identity, and that does not contradicts our cardinality argument.

As a matter of fact there is a generalization of iterated revision operators, called improvement operators from which one can obtain iterated revision operators and at the same time iterated contraction operators. We make some brief comments about this in the next section.

Related Works

In a previous work (Konieczny and Pino Pérez 2008; Konieczny, Medina Grespan, and Pino Pérez 2010) improvement operators are defined as a general class of iterative change operators, that contains Darwiche and Pearl iterated revision operators as special case. Actually, there is a more general class of improvement operators called *basic improvement* operators (Medina Grespan and Pino Pérez 2013). The postulates characterizing these operators say that at least a part of the new piece of information improves and the whole new piece of information does not worsen (this corresponds to postulates C3 and C4 of DP (Darwiche and Pearl 1997)).

Improvement operators are defined semantically on faithful assignments as an increase of plausibility of models of the new piece of information. This increase of plausibility can be more or less restricted, which leads to different families of operators (Konieczny, Medina Grespan, and Pino Pérez 2010). But clearly the increase of plausibility of iterated contraction operators is limited due to the fact that the most plausible models of the new piece of information can not become more plausible than the models of the previous beliefs of the agent. Whereas for revision there is no such constraint, and so much more freedom is granted for improvement.

The following proposition says that our iterated contraction operators are also weak improvement operators (by the negation of the input)

Proposition 2 *Let $\div$ be an iterated contraction operator, then the operator $\hat{\div}$ defined as $\Psi \hat{\div} \alpha = \Psi \div \neg\alpha$ is a weak improvement operator (Konieczny and Pino Pérez 2008), moreover it is a basic improvement operator.*

Thus, the previous proposition and the fact that iterated revision operators are also basic improvement operators seems to mean that this class of operator as the most primitive operators in iterated belief change.

Chopra et al. (Chopra et al. 2008) also give postulates for iterated contraction, but they did it using iterated revision operators in their postulates, so the iterated contraction operators are not defined independently, but as a byproduct of iterated revision ones. Actually their starting point is a couple of given operators $*$ and $-$ that satisfy the Levi and Harper equivalences. Then, they characterize the four iterated semantic properties of Definition 4 in terms of syntactical postulates mixing the operators $*$ and $-$. In this work we propose a direct characterization of iterated contraction operators (not depending of any iterated revision operator).

Booth and Chandler (Booth and Chandler 2016) investigate the problem of the definition of iterated contraction through the Harper Identity for the concrete case of preorders on interpretations. The paper shows the richness of the question. In this paper we explain this richness by the fact that there are much more iterated revision operators than iterated contraction operators, so this means that several different iterated revision operators correspond, via the Harper identity, to the same iterated contraction operator.

The work of Ramachadran et al. (Ramachandran, Nayak, and Orgun 2012) is very interesting. It concerns the characterization of three iterated contraction operator in the framework of the canonical representation of epistemic states. They give a pure syntactical characterization of these three operators. However they don't characterize the full class of iterated contraction operators.

Conclusion

To sum up, in this paper we proposed the first direct logical characterization of the class of iterated contraction operators having an iterated behavior similar to the one proposed by Darwiche and Pearl for iterated revision operators. We stated a representation theorem in terms of total pre-orders on interpretations. We discussed the fact that there is no easy way to generalize the Levi and Harper identity in the iterated case, but more importantly, that this is not a primordial issue, since, conversely to the classical (AGM) case, these two classes of operators are not linked by a bijection in the iterated case. There are more iterated revision operators than iterated contraction operators, and both are special cases of the more general class of improvement operators, where iterated contractions produce a smaller change than iterated revision operators. So, these two classes of change operators are not different in nature, but in degree of change.

As future work we plan to compare our proposal with works on iterated contraction in other (non DP) frameworks

such as (Hild and Spohn 2008; Hansson 2010; 2012).

Acknowledgments

The authors would like to thank the reviewers for their helpful comments.

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic* 50:510–530.

Booth, R., and Chandler, J. 2016. Extending the harper identity to iterated belief change. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 987–993. IJCAI/AAAI Press.

Boutilier, C. 1996. Iterated revision and minimal change of conditional beliefs. *Journal of Philosophical Logic* 25(3):262–305.

Caridroit, T.; Konieczny, S.; and Marquis, P. 2015. Contraction in propositional logic. In *Thirteenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'15)*, 186–196.

Chopra, S.; Ghose, A.; Meyer, T. A.; and Wong, K. 2008. Iterated belief change and the recovery axiom. *J. Philosophical Logic* 37(5):501–520.

Darwiche, A., and Pearl, J. 1997. On the logic of iterated belief revision. *Artificial Intelligence* 89:1–29.

Fermé, E., and Hansson, S. O. 2011. AGM 25 years. *Journal of Philosophical Logic* 40(2):295–331.

Gärdenfors, P. 1988. *Knowledge in flux*. MIT Press.

Hansson, S. O. 1999. *A Textbook of Belief Dynamics. Theory Change and Database Updating*. Kluwer.

Hansson, S. O. 2010. Multiple and iterated contraction reduced to single-step single-sentence contraction. *Synthese* 173(2):153–177.

Hansson, S. O. 2012. Global and iterated contraction and revision: An exploration of uniform and semi-uniform approaches. *Journal of Philosophical Logic* 41(1):143–172.

Harper, W. L. 1977. Rational conceptual change. In *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association,1976, VOL 2*, 462–494. East Lansing, Mich.: Philosophy of Science Association.

Hild, M., and Spohn, W. 2008. The measurement of ranks and the laws of iterated contraction. *Artificial Intelligence* 172(10):1195 – 1218.

Katsuno, H., and Mendelzon, A. O. 1991. Propositional knowledge base revision and minimal change. *Artificial Intelligence* 52:263–294.

Katsuno, H., and Mendelzon, A. O. 1992. On the difference between updating a knowledge base and revising it. In *Belief Revision*. Cambridge University Press. 183–203.

Konieczny, S., and Pino Pérez, R. 2008. Improvement operators. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation And Reasoning (KR 2008)*, 177–187.

Konieczny, S.; Medina Grespan, M.; and Pino Pérez, R. 2010. Taxonomy of improvement operators and the problem of minimal change. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation And Reasoning (KR 2010)*, 161–170.

Levi, I. 1977. Subjunctives, dispositions and chances. *Synthese* 34:423–455.

Medina Grespan, M., and Pino Pérez, R. 2013. Representation of basic improvement operators. In *Trends in Belief Revision and Argumentation Dynamics*. College Publications. 195–227.

Nayak, A. C.; Goebel, R.; Orgun, M. A.; and Pham, T. 2006. Taking Levi identity seriously: A plea for iterated belief contraction. In Lang, J.; Lin, F.; and Wang, J., eds., *Knowledge Science, Engineering and Management, First International Conference, KSEM 2006, Guilin, China, August 5-8, 2006, Proceedings*, volume 4092 of *Lecture Notes in Computer Science*, 305–317. Springer.

Ramachandran, R.; Nayak, A. C.; and Orgun, M. A. 2012. Three approaches to iterated belief contraction. *J. Philosophical Logic* 41(1):115–142.

Splitting Epistemic Logic Programs

Pedro Cabalar
University of Corunna
Corunna, Spain
cabalar@udc.es

Jorge Fandinno
University of Toulouse
IRIT, CNRS, France
jorge.fandinno@irit.fr

Luis Fariñas del Cerro
University of Toulouse
IRIT, CNRS, France
farinas@irit.fr

Abstract

Epistemic logic programs constitute an extension of the stable models semantics to deal with new constructs called *subjective literals*. Informally speaking, a subjective literal allows checking whether some regular literal is true in all stable models or in some stable model. As it can be imagined, the associated semantics has proved to be non-trivial, as the truth of the subjective literal may interfere with the set of stable models it is supposed to query. As a consequence, no clear agreement has been reached and different semantic proposals have been made in the literature. Unfortunately, comparison among these proposals has been limited to a study of their effect on individual examples, rather than identifying general properties to be checked. In this paper, we propose an extension of the well-known splitting property for logic programs to the epistemic case. To this aim, we formally define when an arbitrary semantics satisfies the epistemic splitting property and examine some of the consequences that can be derived from that, including its relation to conformant planning and to epistemic constraints. Interestingly, we prove (through counterexamples) that most of the existing proposals fail to fulfill the epistemic splitting property, except the original semantics proposed by Gelfond in 1991.

Introduction

The language of *epistemic specifications*, proposed by Gelfond (1991), constituted an extension of disjunctive logic programming that introduced modal operators to quantify over the set of stable models (Gelfond and Lifschitz, 1988) of a program. These new constructs were later incorporated as an extension of the Answer Set Programming (ASP) paradigm in different solvers and implementations – see (Lecrerc and Kahl, 2018b) for a recent survey. The new constructs, *subjective literals*, have the form $\mathbf{K}\,l$ and $\mathbf{M}\,l$ and allow respectively checking whether regular literal l is true in every stable model (cautious consequence) or in some stable model (brave consequence) of the program. In many cases, these subjective literals can be seen as simple queries, but what makes them really interesting is their use in rule bodies, what may obviously affect the set of stable models they are meant to quantify. This feature makes them suitable for modelling introspection but, at the same time, easily involves cyclic specifications whose intuitive behaviour is not always easy to define. For instance, the semantics of an epistemic logic program may yield alternative sets of stable models, each set being called a *world view*. Deciding the intuitive world views of a cyclic specification has motivated a wide debate in the literature. In fact, in (Gelfond, 1991) original semantics or in its extension (Truszczyński, 2011), some cyclic examples manifested self-supportedness, so Gelfond (2011) himself and, later on, other authors (Kahl, 2014; Kahl et al., 2015; Fariñas del Cerro, Herzig, and Su, 2015; Shen and Eiter, 2017; Son et al., 2017) proposed different variants trying to avoid unintended results, without reaching a clear agreement. Unfortunately, comparison among these variants was limited to studying their effect on a set of "test" examples. This methodology has proven to fall short in such an uncertain context: confidence in any proposal is always subject to the appearance of new counterintuitive examples. A much stronger method would be defining instead formal properties to be established, as this would cover complete families of examples and, hopefully, could help to reach an agreement on some language fragments. For instance, one would expect that, at least, the existing approaches agreed on their interpretation of acyclic specifications. Regretfully, as we will show later, this is not the case.

In this paper we propose a candidate property, we call *epistemic splitting*, that not only defines an intuitive behaviour for stratified epistemic specifications but also goes further, extending the splitting theorem (Lifschitz and Turner, 1994), well-known for standard logic programs, to the epistemic case. Informally speaking, we say that an epistemic logic program can be split if a part of the program (the *top*) only refers to the atoms of the other part (the *bottom*) through subjective literals. A given semantics satisfies epistemic splitting if, given any splitted program, it is possible to get its world views by first obtaining the world views of the bottom and then using the subjective literals in the top as a "query" on the bottom part previously obtained. If epis-

temic splitting holds, the semantics immediately satisfies other properties. For instance, if the use of epistemic operators is stratified, the program has a unique world view at most. Similarly, epistemic constraints (those only consisting of subjective literals) can be guaranteed to be monotonic: they only rule out candidate world views. As we will see, however, only (Gelfond, 1991) satisfies epistemic splitting among the previously cited approaches. So, somehow, the recent attempts to fix the behaviour of cycles has neglected the attention on the effects produced on acyclic specifications.

The rest of the paper is organized as follows. First, we motivate the main idea through a well-known example. After that, we recall basic definitions of (non-epistemic) answer set programming and splitting, introduces the language of epistemic specifications and defines (Gelfond, 1991) semantics. In the next section, we proceed to define the property of epistemic splitting and study some of its consequences. Then, we formally prove that (Gelfond, 1991) satisfies this property while we provide counterexamples for the other approaches, concluding the paper after that.

Motivation

To illustrate the intuition behind our proposal, let us consider the well-known standard example introduced in (Gelfond, 1991).

Example 1. *A given college uses the following set of rules to decide whether a student X is eligible for a scholarship:*

$$\begin{aligned} \mathit{eligible}(X) &\leftarrow \mathit{high}(X) && (1)\\ \mathit{eligible}(X) &\leftarrow \mathit{minority}(X), \mathit{fair}(X) && (2)\\ {\sim}\mathit{eligible}(X) &\leftarrow {\sim}\mathit{fair}(X), {\sim}\mathit{high}(X) && (3) \end{aligned}$$

Here, '$\sim$' stands for strong negation and $\mathit{high}(X)$ and $\mathit{fair}(X)$ refer to the grades level of student X. We want to encode the additional college criterion:

> *"*The students whose eligibility is not determined by the college rules should be interviewed by the scholarship committee.*"*

as another rule in the program. □

The problem here is that, for deciding whether $\mathit{eligible}(X)$ "*can be determined*," we need to check if it holds in all the answer sets of the program, that is, if it is one of the cautious consequences of the latter. For instance, if the only available information for some student *mike* is the disjunction

$$\mathit{fair}(\mathit{mike}) \vee \mathit{high}(\mathit{mike}) \qquad (4)$$

we get that program $\{(1)-(4)\}$ has two stable models, $\{\mathit{fair}(\mathit{mike})\}$ and $\{\mathit{high}(\mathit{mike}), \mathit{eligible}(\mathit{mike})\}$ so $\mathit{eligible}(\mathit{mike})$ cannot be determined and an interview should follow. Of course, if we just want to query cautious and brave consequences of the program, we can do it inside ASP. For instance, the addition of constraint:

$$\bot \leftarrow \mathit{eligible}(\mathit{mike})$$

allows us deciding if $\mathit{eligible}(\mathit{mike})$ is a cautious consequence by just checking that the resulting program has no answer sets. The difficulty comes from the need to *derive* new information from a cautious consequence. This is where subjective literals come into play. Rule

$$\begin{aligned} \mathit{interview}(X) \leftarrow\ & \mathtt{not}\ \mathbf{K}\ \mathit{eligible}(X), \\ & \mathtt{not}\ \mathbf{K}\ {\sim}\mathit{eligible}(X) \end{aligned} \qquad (5)$$

allows us to prove that $\mathit{interview}(X)$ holds when neither $\mathit{eligible}(X)$ nor ${\sim}\mathit{eligible}(X)$ are cautious consequences of (1)-(4). The novel feature here is that (5) is also part of the program, and so, it affects the answer sets queried by $\mathbf{K}$ too, which would actually be:

$$\{\mathit{fair}(\mathit{mike}), \mathit{interview}(\mathit{mike})\} \qquad (6)$$

$$\{\mathit{high}(\mathit{mike}), \mathit{eligible}(\mathit{mike}), \mathit{interview}(\mathit{mike})\} \qquad (7)$$

So, there is a kind of cyclic reasoning: operators $\mathbf{K}$ and $\mathbf{M}$ are used to query a set of stable models that, in their turn, may depend on the application of that query. In the general case, this cyclic reasoning is solved by resorting to multiple world views, but in our particular example, however, this does not seem to be needed. One would expect that separating the queried part (1)-(4) and the rule that makes the query (5) should be correct, since the first four rules do not depend on (5) and the latter exclusively consults them without interacting with their results. This same reasoning could be applied if we added one more level such as, for instance, by including the rule:

$$\mathit{appointment}(X) \leftarrow \mathbf{K}\ \mathit{interview}(X) \qquad (8)$$

The two answer sets of program $\{(1)-(5)\}$ contain $\mathit{interview}(\mathit{mike})$ and so $\mathit{appointment}(\mathit{mike})$ can be added to both answer sets incrementally. This method of analysing a program by division into independent parts shows a strong resemblance to the *splitting theorem* (Lifschitz and Turner, 1994), well-known in standard ASP. Splitting is applicable when the program can be divided into two parts, the *bottom* and the *top*, in such a way that the bottom never refers to head atoms in the top. When this happens, we can first compute the answer sets of the bottom and then, for each one, simplify the top accordingly, getting new answer sets that complete the information. We could think about different ways of extending this method for the case of epistemic logic programs, depending on how restrictive we want to be on the programs where it will be applicable. However, we will choose a very conservative case, looking for a wider agreement on the proposed behaviour. The condition we will impose is that our top program can only refer to atoms in the bottom through epistemic operators. In this way, the top is seen as a set of rules that derive facts from epistemic queries on the bottom. Thus, each world view W of the bottom will be used to replace the subjective literals in the top by their truth value with respect to W. For the sake of completeness, we recall next the basic definitions of ASP and splitting, to proceed with a formalization of epistemic splitting afterwards.

Background of ASP and splitting

Given a set of atoms At, a *regular literal*[1] is either an atom or constant, $a \in At \cup \{\top, \bot\}$, or its default negation, $\texttt{not}\ a$. A *rule* r is an implication of the form:

$$a_1 \vee \cdots \vee a_n \leftarrow L_1, \ldots, L_m \tag{9}$$

with $n \geq 0$ and $m \geq 0$, where each $a_i \in At$ is an atom and each L_j a regular literal. The left hand disjunction of (9) is called the rule *head* and abbreviated as $Head(r)$. When $n = 0$, it corresponds to $\bot$ and r is called a *constraint*. The right hand side of (9) is called the rule *body* and abbreviated as $Body(r)$. When $m = 0$, the body corresponds to $\top$ and r is called a *fact* (in this case, the body and the arrow symbol are usually omitted). A *program* Π is a set of rules. We write $Atoms(F)$ to represent the set of atoms occurring in any syntactic construct F (a literal, head, body, rule or program). A propositional interpretation I is a set of atoms. We assume that strong negation '$\sim a$' is just another atom in At and that the constraint:

$$\bot \leftarrow a, \sim a$$

is implicitly included in the program. We allow the use of variables, but understood as abbreviations of their possible ground instances. Given any syntactic construct F, we write $I \models F$ to stand for "I satisfies F" in classical propositional logic, where the commas correspond to conjunctions, '$\texttt{not}$' corresponds (under this interpretation) to classical negation and '$\leftarrow$' is just a reversed material implication. An interpretation I is a *(classical) model* of a program Π if it satisfies all its rules. The *reduct* of a program Π with respect to some propositional interpretation I, in symbols Π^I, is obtained by replacing in Π every negative literal $\texttt{not}\ a$ by $\top$ if $I \models \texttt{not}\ a$ or by $\bot$ otherwise. A propositional interpretation I is a *stable model* of a program Π iff it is a $\subseteq$-minimal model of Π^I. By SM[Π], we denote the set of all stable models of Π.

The following is a well-known property in ASP.

Property 1 (Supraclassicality)**.** *Any stable model of a program* Π *is also a classical model of* Π.

We recall next the splitting theorem for ASP, beginning with the following definition.

Definition 1 (Splitting set)**.** *A set of atoms* $U \subseteq$ At *is a* splitting set *of a program* Π *if, for each rule* $r \in \Pi$*, one of the following conditions hold*

i) $Atoms(r) \subseteq U$,

ii) $Atoms(Head(r)) \cap U = \emptyset$

When this happens, we identify two disjoint subprograms, the bottom *and the* top*, respectively defined as*

$$b_U(\Pi) \stackrel{\text{def}}{=} \{\ r \in \Pi \mid Atoms(r) \subseteq U\ \}$$
$$t_U(\Pi) \stackrel{\text{def}}{=} \Pi \setminus b_U(\Pi) \qquad \square$$

[1]For a simpler description of program transformations, we allow truth constants with their usual meaning.

As an example, consider program Π_1:

$$a \leftarrow \texttt{not}\ b \tag{10}$$
$$b \leftarrow \texttt{not}\ a \tag{11}$$
$$c \vee d \leftarrow \texttt{not}\ a \tag{12}$$
$$d \leftarrow a, \texttt{not}\ b \tag{13}$$

It is easy to see that $U = \{a, b\}$ is a splitting set that divides the program into two parts: the bottom $b_U(\Pi_1) = \{(10), (11)\}$ and the top $t_U(\Pi_1) = \{(12), (13)\}$.

The keypoint of splitting is computing stable models of $b_U(\Pi)$ alone and using each one, I, to simplify $t_U(\Pi)$ accordingly. Given a splitting set U for Π and an interpretation $I \subseteq U$, we define the program $e_U(\Pi, I)$ as a transformation of the top program, $t_U(\Pi)$, where we replace each atom $a \in U$ from the splitting set by: $\top$ if $a \in I$ or $\bot$ otherwise. A pair $\langle I_b, I_t \rangle$ is said to be a *solution* of Π with respect to U iff I_b is a stable model of $b_U(\Pi)$ and I_t is a stable model of $e_U(\Pi, I_b)$. For instance, for Π_1, the bottom has two answer sets $\{a\}$ and $\{b\}$, so we get the respective simplifications $e_U(\Pi_1, \{a\})$:

$$c \vee d \leftarrow \texttt{not}\ \top \qquad d \leftarrow \top, \texttt{not}\ \bot$$

and $e_U(\Pi_1, \{b\})$:

$$c \vee d \leftarrow \texttt{not}\ \bot \qquad d \leftarrow \bot, \texttt{not}\ \top$$

The former has stable model $\{d\}$ so $\langle \{a\}, \{d\} \rangle$ is one solution. The latter has stable models $\{c\}$ and $\{d\}$ that yield other two solutions $\langle \{b\}, \{c\} \rangle$ and $\langle \{b\}, \{d\} \rangle$.

Theorem 1 (From Lifschitz and Turner 1994)**.** *Let* U *be a splitting set of program* Π*. A propositional interpretation* $I \subseteq At$ *is a stable model of* Π *iff there is a solution* $\langle I_b, I_t \rangle$ *of* Π *w.r.t* U *such that* $I = I_b \cup I_t$*.* $\square$

Given the three solutions we obtained before, the splitting theorem guarantees that $\{a\} \cup \{d\}$, $\{b\} \cup \{c\}$ and $\{b\} \cup \{d\}$ are the three stable models of our example program Π_1.

One interesting observation is that any constraint r with $Atoms(r) \subseteq U$ is now included in the bottom $b_U(\Pi)$ but also satisfies condition ii) in Def. 1 (it has no head atoms at all) and could be moved to the top $t_U(\Pi)$ instead. Having this in mind, let us provide now a relaxed definition of bottom and top programs in the following way. We say that the pair $\langle \hat{b}_U(\Pi), \hat{t}_U(\Pi) \rangle$ is an *arbitrary splitting* of program Π with respect to splitting set U if: $\hat{b}_U(\Pi) \cap \hat{t}_U(\Pi) = \emptyset$, $\hat{b}_U(\Pi) \cup \hat{t}_U(\Pi) = \Pi$, all rules in B_U satisfy Def. 1.i) and all rules in T satisfy Def. 1.ii). With this definition, constraints on atoms in U can be arbitrarily placed in $\hat{b}_U(\Pi)$ or in $\hat{t}_U(\Pi)$.

Corollary 1. *Theorem 1 still holds if we define* $b_U(\Pi)$ *and* $t_U(\Pi)$ *to be any arbitrary splitting* $\langle \hat{b}_U(\Pi), \hat{t}_U(\Pi) \rangle$ *of program* Π *with respect to splitting set* U*.* $\square$

Epistemic specifications

We extend now the syntax of ASP to the language of epistemic specifications. Given a set of atoms At, we call

subjective literal to any expression of the form $\mathbf{K}\, l$, $\mathbf{M}\, l$, $\texttt{not}\ \mathbf{K}\, l$ or $\texttt{not}\ \mathbf{M}\, l$, for any regular literal l. We keep the same syntax for rules as in (9) excepting that body literals L_j can also be subjective literals now. Given rule r we define the sets $Body(r)^{reg}$ and $Body^{sub}(r)$ respectively containing the regular and the subjective literals in $Body(r)$. Rules or programs are *regular* if they do not contain subjective literals. We say that a rule is a *subjective constraint* if it is a constraint, $Head(r) = \bot$, and its body exclusively consists of subjective literals, that is $Body(r) = Body^{sub}(r)$.

We can define the concept of *model* of a program, in a similar way as we did for classical models in regular ASP. A *modal interpretation* $\mathcal{M} = \langle W, I\rangle$ is pair where I is a propositional interpretation and $W \subseteq 2^{At}$ is a non-empty set of propositional interpretations. A modal interpretation $\mathcal{M} = \langle W, I\rangle$ *satisfies* a literal L, written $\langle W, I\rangle \models L$, if

1. $\langle W, I\rangle \models \top$,
2. $\langle W, I\rangle \not\models \bot$,
3. $\langle W, I\rangle \models a$ if $a \in I$, for any atom $a \in At$,
4. $\langle W, I\rangle \models \mathbf{K}\, l$ if $\langle W, I'\rangle \models l$ for all $I' \in W$,
5. $\langle W, I\rangle \models \mathbf{M}\, l$ if $\langle W, I'\rangle \models l$ for some $I' \in W$, and
6. $\langle W, I\rangle \models \texttt{not}\ L$ if $\langle W, I\rangle \not\models L$.

Since for a subjective literal L, $\langle W, I\rangle \models L$ does not depend on I, we sometimes write $W \models L$. For a rule r of the form (9), we write $\langle W, I\rangle \models r$ iff either $\langle W, I\rangle \models a_i$ for some $1 \leq i \leq n$ or $\langle W, I\rangle \not\models L_j$ for some $1 \leq j \leq m$. We say that $\langle W, I\rangle$ is a *model* of a program Π, written $\langle W, I\rangle \models \Pi$, if it satisfies all its rules. Among the possible models of an epistemic logic program, all semantic approaches agree on selecting some preferred models called *world views*, each one being characterized by the W component. These world views satisfy a similar property to that of supraclassicality (Property 1) in non-epistemic ASP. In this case, however, rather than talking about classical models, we resort to modal logic S5, so all world views of a program are also S5 models of the program. This property can be formally stated as follows:

Property 2 (Supra-S5)**.** *A semantics satisfies* supra-S5 *when for every world view W of an epistemic program Π and for every $I \in W$, $\langle W, I\rangle \models \Pi$.* □

To the best of our knowledge, all existing semantics satisfy supra-S5. Another property that is shared by all semantics is that, when Π is a regular ASP program (it has no modal epistemic operators) then it has a unique world view containing all the stable models of Π. We will formalize this property in the following way.

Property 3 (Supra-ASP)**.** *A semantics satisfies* supra-ASP *if for any regular program Π either Π has a unique world view $W = \mathrm{SM}[\Pi] \neq \emptyset$ or $\mathrm{SM}[\Pi] = \emptyset$ and Π has no world view at all.* □

Originally, some semantics like (Gelfond, 1991) or (Truszczyński, 2011), allowed empty world views $W = \emptyset$ when the program has no stable models, rather than leaving the program without world views. Since this feature is not really essential, we exclusively refer to non-empty world views in this paper.

We define next a useful transformation extending the idea of reduct to epistemic specifications, and generalized for a given signature.

Definition 2 (Subjective reduct)**.** *The* subjective reduct *of a program Π with respect to a set of propositional interpretations W and a signature $U \subseteq$ At, also written Π_U^W, is obtained by replacing each subjective literal L with $Atoms(L) \subseteq U$ by: $\top$ if $W \models L$ or by $\bot$ otherwise. When $U = At$ we just write Π^W.* □

We use the same notation Π^W as for the standard reduct, but ambiguity is removed by the type of W (a set of interpretations now). This subjective reduct can be used to define (Gelfond, 1991) (G91) semantics in the following way.

Definition 3 (G91-world view)**.** *A non-empty set of interpretations W is a* G91-world view *of an epistemic program Π if $W = \mathrm{SM}[\Pi^W]$.* □

We will not provide the formal definitions of the rest of semantics compared in this paper, since none of them satisfies our goal property of epistemic splitting. In those cases, it will suffice with providing counterexamples and the reader can check their behaviour by resorting to the corresponding original definition.

Epistemic splitting

We proceed now to introduce our definition of the epistemic splitting property. To do so, we begin extending the idea of splitting set.

Definition 4 (Epistemic splitting set)**.** *A set of atoms $U \subseteq$ At is said to be an* epistemic splitting set *of a program Π if for any rule r in Π one of the following conditions hold*

i) $Atoms(r) \subseteq U$,
ii) $(Atoms(Body^{reg}(r) \cup Head(r))) \cap U = \emptyset$

As before, we define an arbitrary splitting *of Π as a pair $\langle B_U(\Pi), T_U(\Pi)\rangle$ satisfying $B_U(\Pi) \cap T_U(\Pi) = \emptyset$, $B_U(\Pi) \cup T_U(\Pi) = \Pi$, all rules in $B_U(\Pi)$ satisfy i) and all rules in $T_U(\Pi)$ satisfy ii).* □

With respect to Definition 1, we have replaced the condition for the top program, $Atoms(Head(r)) \cap U = \emptyset$, by the new condition ii), which in other words means that the top program may only refer to atoms U in the bottom through epistemic operators. Note that this introduces a new kind of "dependence," so that, as happens with head atoms, regular literals in the body also depend on atoms in subjective literals. For instance, if $U = \{p, q\}$, the program

$$p \vee q \qquad\qquad s \leftarrow p, \mathbf{K}\, q$$

would not be splittable due to the second rule, since $s \notin U$ and we would also need the regular literal $p \notin U$. The reason for this restriction is to avoid imposing (to

a potential semantics) a fixed way of evaluating p with respect to the world view $[\{p\}, \{q\}]$ for the bottom.

Another observation is that we have kept the definition of $B_U(\Pi)$ and $T_U(\Pi)$ non-deterministic, in the sense that some rules can be arbitrarily included in one set or the other. In our case, these rules correspond to subjective constraints on atoms in U, since these are the only cases that may satisfy conditions i) and ii) simultaneously.

If we retake our example program $\Pi_2 = \{(1) - (5)\}$, we can see that $U = \{high(mike), fair(mike), eligible(mike), minority(mike)\}$ is an epistemic splitting set that divides the program into the bottom $B_U(\Pi_2) = \{(1) - (4)\}$ and the top $T_U(\Pi_2) = \{(5)\}$. As in regular splitting, the idea is computing first the world views of the bottom program $B_U(\Pi)$ and for each one, W, simplifying the corresponding subjective literals in the top program. Given an epistemic splitting set U for a program Π and set of interpretations W, we define $E_U(\Pi, W) \stackrel{\text{def}}{=} T_U(\Pi)^W_U$, that is, we make the subjective reduct of the top with respect to W and signature U. A pair $\langle W_b, W_t\rangle$ is said to be a *solution* of Π with respect to an epistemic splitting set U if W_b is a world view of $B_U(\Pi)$ and W_t is a world view of $E_U(\Pi, W_b)$. Notice that this definition is semantic-dependent in the sense that each alternative semantics for epistemic specifications will define its own solutions for a given U and Π, since it defines the selected world views for a program in a different way. Back to our example, notice that $B_U(\Pi_2)$ is a regular program without epistemic operators. Thus, any semantics satisfying supra-ASP will provide $W_b = [\{fair(mike)\}, \{high(mike), eligible(mike)\}]$ as unique world view for the bottom. The corresponding simplification of the top would be $E_U(\Pi_s, W_b)$ containing (after grounding) the single rule:

$$interview(mike) \leftarrow \texttt{not}\ \bot, \texttt{not}\ \bot$$

Again, this program is regular and its unique world view would be $W_t = [\{interview(mike)\}]$. Now, in the general case, to reconstruct the world views for the global program we define the operation:

$$W_b \sqcup W_t \;=\; \{\; I_b \cup I_t \;\mid\; I_b \in W_b \text{ and } I_t \in W_t \;\}$$

(remember that both the bottom and the top may produce multiple world views, depending on the program and the semantics we choose). In our example, $W_b \sqcup W_t$ would exactly contain the two stable models (6), (7) we saw in the introduction.

Property 4 (Epistemic splitting)**.** *A semantics satisfies* epistemic splitting *if for any epistemic splitting set U of any program Π: W is a world view of Π iff there is a solution $\langle W_b, W_t\rangle$ of Π with respect to U such that $W = W_b \sqcup W_t$.* □

In the example, this means that the world view we obtained in two steps is indeed the unique world view of the whole program, under any semantics satisfying epistemic splitting. Uniqueness of world view was obtained in this case because both the bottom program $B_U(\Pi_2)$ and the top, after simplification, $E_U(\Pi_2, W_b)$ were regular programs and we assumed supra-ASP. In fact, as we see next, we can still get a unique world view (at most) when there are no cyclic dependences among subjective literals. This mimics the well-known result for *stratified negation* in logic programming (van Gelder, 1988; Apt, Blair, and Walker, 1988). Let us define a modal dependence relation among atoms in a program Π so that $dep(a, b)$ is true iff there is a rule $r \in \Pi$ such that $a \in Atoms(Head(r) \cup Body^{reg}(r))$ and $b \in Atoms(Body^{sub}(r))$.

Definition 5. *We say that an epistemic program Π is* modally stratified *if we can assign an integer mapping $\lambda : \text{At} \to \mathbb{N}$ to each atom such that $\lambda(a) > \lambda(b)$ for any pair of atoms a, b satisfying $dep(a, b)$.* □

Take, for instance, the extended program $\Pi_3 = \{(1) - (5), (8)\}$. We can assign atoms $high(mike)$, $fair(mike)$, $minority(mike)$ and $eligible(mike)$ layer 0. Then $interview(mike)$ could be assigned layer 1 and, finally, $appointment(mike)$ can be located at layer 2. So, Π_3 is modally stratified.

Theorem 2. *Let Π be a finite, modally stratified program. Then, any semantics satisfying supra-ASP and epistemic splitting assigns, at most, a unique world view to Π.* □

The proof of the theorem just relies on multiple applications of splitting to each layer backwards and the fact that each simplification $E_U(\Pi, W_b)$ will be a regular program. This is very easy to see in the extended example Π_3. We can split the program using as U all atoms but $appointment(mike)$ to get a bottom Π_2 and a top $\{(8)\}$. Program Π_2 can be splitted in its turn as we saw before, producing the unique world view $\{(6), (7)\}$. Then $E_U(\Pi_3, \{(6), (7)\})$ contains the single rule $appointment(mike) \leftarrow \top$ that is a regular program whose unique world view is $[\{appointment(mike)\}]$ and, finally, the combination of both world views yields again a unique world view $[(6) \cup \{appointment(mike)\}, (7) \cup \{appointment(mike)\}]$.

Another consequence of epistemic splitting is that subjective constraints will have a monotonic behaviour. Note first that, for a subjective constraint r, we can abbreviate $\langle W, I\rangle \models r$ as $W \models r$ since the I component is irrelevant. Additionally, $W \models r$ means that $Body(r) = Body^{sub}(r)$ is falsified, since $Head(r) = \bot$.

Property 5 (subjective constraint monotonicity)**.** *A semantics satisfies* subjective constraint monotonicity *if, for any epistemic program Π and any subjective constraint r, W is a world view of $\Pi \cup \{r\}$ iff both W is a world view of Π and $W \models r$.* □

Theorem 3. *Epistemic splitting implies subjective constraint monotonicity.* □

Proof. Suppose we use a semantics satisfying epistemic splitting. For any program Π and any epistemic constraint r, we can always take the whole set of atoms $U = Atoms(\Pi \cup \{r\})$ as epistemic splitting set for

$\Pi' = \Pi \cup \{r\}$ and take $B_U(\Pi') = \Pi$ and $T_U(\Pi') = \{r\}$. For any world view W of $B_U(\Pi')$ two things may happen. A first possibility is $W \models r$, and so the body of r has some false subjective literal in W, so $E_U(\Pi', W)$ would be equivalent to $\bot \leftarrow \bot$. Then, the unique world view for the top would be $W_t = [\emptyset]$ and $W \sqcup W_t = W$. A second case is $W \not\models r$, so all literals in the body are satisfied and $E_U(\Pi', W)$ would be equivalent to $\bot \leftarrow \top$ which has no world views. To sum up, we get exactly those world views W of Π that satisfy r. □

To conclude the exploration of consequences of epistemic splitting, let us consider a possible application to conformant planning. To this aim, consider the following simple example.

Example 2. *To turn on the light in a room, we can toggle one of two lamps l_1 or l_2. In the initial state, lamp l_1 is plugged but we ignore the state of l_2. Our goal is finding a plan that guarantees we get light in the room in one step.*

A possible logic program that encodes this scenario for a single transition[2] could be Π_4:

$$\begin{aligned} & plugged(l_1) \\ & plugged(l_2) \vee {\sim} plugged(l_2) \\ light \quad & \leftarrow \quad toggle(L), plugged(L) \\ \bot \quad & \leftarrow \quad toggle(l_1), toggle(l_2) \end{aligned}$$

for $L \in \{l_1, l_2\}$. As we can see, $toggle(l_1)$ would constitute a conformant plan, since we obtain *light* regardless of the initial state, while this does not happen with plan $toggle(l_2)$. In order to check whether a given sequence of actions $A_0, \ldots, A_n$ is a valid conformant plan one would expect that, if we added those facts to the program, a subjective constraint should be sufficient to check that the goal holds in all the possible outcomes. In our example, we would just use:

$$\bot \leftarrow \texttt{not}\ \mathbf{K}\, light \tag{14}$$

and check that the program $\Pi_4 \cup \{toggle(L)\} \cup \{(14)\}$ has some world view, varying $L \in \{l_1, l_2\}$. Subjective constraint monotonicity guarantees that the addition of this "straighforward" formalisation has the expected meaning.

This method would only allow testing if the sequence of actions constitutes a conformant plan, but does not allow generating those actions. A desirable feature would be the possibility of applying the well-known ASP methodology of separating the program into three sections: generate, define and test. In our case, the "define" and the "test" sections would respectively be Π_4 and (14), but we still miss a "generate" part, capable of considering different alternative conformant plans. The problem in this case is that we cannot use a simple choice:

$$toggle(L) \vee {\sim} toggle(L)$$

because this would allow a same action to be executed in some of the stable models and not executed in others, all inside a *same* world view. Let us assume that our epistemic semantics has some way to non-deterministically generate a world view in which either $\mathbf{K}\, a$ or $\mathbf{K}\, \texttt{not}\ a$ holds using a given set of rules[3] $Choice(a)$. Then, take the program Π_5 consisting of rules

$$Choice(toggle(L)) \tag{15}$$

with $L \in \{l_1, l_2\}$ plus Π_4 and (14). If our semantics satisfies epistemic splitting, it is safe to obtain the world views in three steps: generate first the alternative world views for $toggle(l_1)$ and $toggle(l_2)$ using (15), apply Π_4 and rule out those world views not satisfying the goal *light* in all situations using (14). To fulfill the preconditions for applying splitting, we would actually need to replace regular literal $toggle(L)$ by $\mathbf{K}\, toggle(L)$ in all the bodies of Π_4, but this is safe in the current context. Now, we take the bottom program to obtain 4 possible world views $W_0 = [\{toggle(l_1)\}]$, $W_1 = [\{toggle(l_2)\}]$, $W_2 = [\{toggle(l_1), toggle(l_2)\}]$ and $W_3 = [\emptyset]$. When we combine them with the top Π_4 we obtain W_0' consisting of two stable models:

$$\begin{aligned} & \{toggle(l_1), plugged(l_2), light, \ldots\} \\ & \{toggle(l_1), {\sim} plugged(l_2), light, \ldots\} \end{aligned}$$

and W_1' consisting of other two stable models:

$$\begin{aligned} & \{toggle(l_2), plugged(l_2), light, \ldots\} \\ & \{toggle(l_2), {\sim} plugged(l_2), \ldots\} \end{aligned}$$

where the latter does not contain *light*. Finally, constraint (14) would rule out W_1'.

To sum up, epistemic splitting provides a natural way of formulating conformant planning problems by a separation into three sections: a generation part, the usual encoding of the actions scenario and a test part consisting of a subjective constraint to guarantee that the goal is always reached.

Splitting in some existing semantics

In this section we study the property of epistemic splitting for the approaches mentioned in the introduction. We will begin by proving that G91 actually satisfies this property. To this aim, we start with some definitions and auxiliary results. Given a set of propositional interpretations $W \subseteq 2^{At}$ and a set of atoms U, by $W_{|U} \stackrel{\text{def}}{=} \{\ I \cap U \mid I \in U\ \}$, we denote the restriction of W to U. Given a set of atoms U, by $\overline{U}$, we denote its complement $At \setminus U$.

Observation 1. *Let W be a set of propositional interpretations and $U \subseteq$ At be a set of atoms. Then, for any subjective literal L with $Atoms(L) = \{a\}$:*

i) if $a \in U$, then $W \models L$ iff $W_{|U} \models L$,

[2] For simplicity, we omit time arguments or inertia, as they are not essential for the discussion

[3] For instance, in Gelfond (1991), this could be just the rule $a \leftarrow \texttt{not}\ \mathbf{K}\, \texttt{not}\ a$. Other semantics may have alternative ways of expressing this intended behaviour.

ii) if $a \notin U$, then $W \models L$ iff $W_{|\overline{U}} \models L$,

Proposition 1. *Let Π be a program that accepts an epistemic splitting set $U \subseteq At$ and let W be a set of propositional interpretations. Let $W_b = W_{|U}$ and $W_t = W_{|\overline{U}}$. Then, we get*

i) $B_U(\Pi)^W = B_U(\Pi)^{W_b}$,
ii) $T_U(\Pi)^W = E_U(\Pi, W_b)^{W_t}$, and
iii) $\Pi^W = B_U(\Pi)^{W_b} \cup E_U(\Pi, W_b)^{W_t}$.

Proof. First, since every rule $r \in B_U(\Pi)$ satisfies $Atoms(Body^{sub}(r)) \subseteq U$, from Observation 1, it follows that $B_U(\Pi)^W = B_U(\Pi)^{W_b}$. Furthemore, for any program Γ, it is easy to check that $\Gamma^W = (\Gamma_U^{W_b})^{W_t}$, that is, applying the reduct w.r.t W is the same than applying it w.r.t. to its projection in U and afterwards to the remaining part. Thus, we get $T_U(\Pi)^W = (T_U(\Pi)_U^{W_b})^{W_t} = E_U(\Pi, W_b)^{W_t}$. Finally, we have that $\Pi^W = (B_U(\Pi) \cup T_U(\Pi))^W = B_U(\Pi)^W \cup T_U(\Pi)^W$ and, thus, the result holds. □

Theorem 4 (Main theorem)**.** *Semantics G91 satisfies epistemic splitting.*

Proof. Let W be some set of propositional interpretations and let $W_b = W_{|U}$ and $W_t = W_{|\overline{U}}$. By definition, W is a world view of Π if and only if $W = \mathrm{SM}[\Pi^W]$. Furthermore, since U is a modal splitting set of Π, it is easy to check that U is also a regular splitting set of the regular programa Π^W. Hence, from Corollary 1, we get that W is a world view of Π iff $W = \mathrm{SM}[\Pi^W] =$

$$\left\{ I_b \cup I_t \;\middle|\; I_b \in \mathrm{SM}[\hat{b}_U(\Pi^W)] \text{ and } I_t \in \mathrm{SM}[\hat{e}_U(\Pi^W, I_b)] \right\}$$

for some arbitrary splitting $\langle \hat{b}_U(\Pi^W), \hat{t}_U(\Pi^W) \rangle$. Note that all rules belonging to $B_U(\Pi)$ have all atoms from U. Hence, we take $\hat{b}_U(\Pi^W) \stackrel{\text{def}}{=} B_U(\Pi)^W = B_U(\Pi)^{W_b}$ (Proposition 1). Similarly, we also take $\hat{t}_U(\Pi^W) \stackrel{\text{def}}{=} T_U(\Pi)^W = E_U(\Pi, W_b)^{W_t}$. Then, we get

$$\hat{e}_U(\Pi^W, I_b) = \hat{e}_U(\hat{t}_U(\Pi^W), I_b) = \hat{e}_U(E_U(\Pi, W_b)^{W_t}, I_b)$$

Notice also that no atom occurring in $E_U(\Pi, W_b)^{W_t}$ belongs to U, which implies that $\hat{e}_U(E_U(\Pi, W_b)^{W_t}, I_b) = E_U(\Pi, W_b)^{W_t}$. Replacing above, we have that W is a world view of Π iff W is equal to

$$\{I_b \cup I_t \mid I_b \in \mathrm{SM}[B_U(\Pi)^{W_b}], I_t \in \mathrm{SM}[E_U(\Pi, W_b)^{W_t}]\}$$

iff

$$W = \{I_b \cup I_t \mid I_b \in W_b' \text{ and } I_t \in W_t'\}$$

with $W_b' = \mathrm{SM}[B_U(\Pi)^{W_b}]$ and $W_t' = \mathrm{SM}[E_U(\Pi, W_b)^{W_t}]$ iff $W = W_b' \sqcup W_t'$. Hence, it only remains to be shown that both $W_b = W_b'$ and $W_t = W_t'$ hold. Note that $I \in W_b = W_{|U}$ iff $I = I' \cap U$ for some $I' \in W$ iff $I = (I_b \cup I_t) \cap U$ for some $I_b \in W_b'$ and $I_t \in W_t'$ iff $I = (I_b \cap U) \cup (I_t \cap U)$ for some $I_b \in W_b'$ and $I_t \in W_t'$ iff $I = I_b$ for some $I_b \in W_b'$. The fact $W_t = W_t'$ follows in an analogous way. □

A similar proof can be developed to show that (Truszczyński, 2011), that generalises[4] (Gelfond, 1991) from subjective literals to subjective formulas, also satisfies epistemic splitting.

To illustrate the behaviour of other semantics with respect to splitting, we will use several examples. Let us take the program Π_6 consisting of $\{(10), (11)\}$ and the rule:

$$c \vee d \;\leftarrow\; \texttt{not}\ \mathbf{K}\, a \tag{16}$$

The set $U = \{a, b\}$ splits the program into the bottom, (10)-(11) and the top (16). The bottom has a unique world view $W_b = [\{a\}, \{b\}]$ so $\mathbf{K}\, a$ does not hold and the top is simplified as $E_U(\Pi_6, W_b)$ containing the unique rule:

$$c \vee d \leftarrow \texttt{not}\ \bot \tag{17}$$

This program has a unique world view $W_t = [\{c\}, \{d\}]$ that, combined with W_b yields $[\{a, c\}, \{b, c\}, \{a, d\}, \{b, d\}]$ as the unique solution for Π_6, for any semantics satisfying epistemic splitting (and so, also for G91). Let us elaborate the example a little bit further. Suppose we add now the constraint:

$$\bot \leftarrow c \tag{18}$$

The top must also include this rule and has now a unique stable model $W_t = [\{d\}]$, so the world view for the complete program would be $[\{a, d\}, \{b, d\}]$. Finally, let us forbid the inclusion of atom d too:

$$\bot \leftarrow d \tag{19}$$

so we consider $\Pi_7 = \{(10), (11), (16), (18), (19)\}$. This last constraint leaves the simplified top program $E_U(\Pi_6, W_b) = \{(17), (18), (19)\}$ without stable models, so epistemic splitting would yield that program Π_7 has no world view at all. This is the result we obtain, indeed, in (Gelfond, 1991, 2011)[5] and in (Truszczyński, 2011). Surprisingly, recent approaches like (Kahl et al., 2015; Fariñas del Cerro, Herzig, and Su, 2015; Shen and Eiter, 2017; Son et al., 2017) yield world view $[\{a\}]$, violating the epistemic splitting property. For instance, in the case of (Kahl et al., 2015), the reduct of Π_7 with respect to $[\{a\}]$ is the program

$$\begin{aligned} a &\leftarrow \texttt{not}\ b \\ b &\leftarrow \texttt{not}\ a \\ c \vee d &\leftarrow \texttt{not}\ a \\ \bot &\leftarrow c \\ \bot &\leftarrow d \end{aligned}$$

which has a unique stable model $\{a\}$.

[4] In fact, Truszczyński (2011) defines several semantics but, among them, we refer here to the epistemic stable model semantics.

[5] These two semantics actually produce empty world views, but as we said before, we disregard them, as they just point out that the program has no solution.

As a second example, take the program Π_8 consisting of the same bottom program $\{(10), (11)\}$ and the rule:

$$c \leftarrow \mathbf{K}\, a \tag{20}$$

As expected, all approaches agree that Π_8 has a unique world view $W_b = [\{a\}, \{b\}]$ because $\mathbf{K}\, a$ is not satisfied and rule (20) is not applicable. Under epistemic splitting, we get that $E_U(\Pi_8, W_b)$ is the rule:

$$c \leftarrow \bot \tag{21}$$

whose unique world view is $[\emptyset]$, so that $W_b \sqcup [\emptyset] = W_b$. But let us further elaborate the example taking Π'_8 containing Π_8 plus:

$$\bot \leftarrow \texttt{not}\ c \tag{22}$$

Under epistemic splitting, the new top $E_U(\Pi'_8, W_b)$ contains now (22) and (21) which have no stable models. As a result, no world view can be combined with W_b and we obtain that Π'_8 has no world views at all. This is the result we obtain under (Gelfond, 1991; Truszczyński, 2011), which agree that the program is inconsistent. However, Gelfond (2011) joins (Kahl et al., 2015; Fariñas del Cerro, Herzig, and Su, 2015; Shen and Eiter, 2017; Son et al., 2017) in the group of approaches that provide the world view $[\{a, c\}]$. That is, in all these approaches, adding a constraint intended to remove all world views that do not satisfy c, may surprisingly lead to justify c. Note that, according to (Gelfond, 1991; Truszczyński, 2011), the reduct of Π'_8 with respect to $[\{a, c\}]$ is

$$a \leftarrow \texttt{not}\ b \tag{23}$$
$$b \leftarrow \texttt{not}\ a \tag{24}$$
$$c \leftarrow \top \tag{25}$$
$$\bot \leftarrow \texttt{not}\ c \tag{26}$$

which has two stable models, $\{a, c\}$ and $\{b, c\}$, so $[\{a, c\}]$ is not a world view. In contrast, the reduct with respect to (Gelfond, 2011) and (Kahl et al., 2015) is

$$a \leftarrow \texttt{not}\ b \tag{27}$$
$$b \leftarrow \texttt{not}\ a \tag{28}$$
$$c \leftarrow a \tag{29}$$
$$\bot \leftarrow \texttt{not}\ c \tag{30}$$

which has a unique stable model $\{a, c\}$, so $[\{a, c\}]$ is a world view.

Conclusions

We have introduced a formal property for semantics of epistemic specifications. This property that we call *epistemic splitting* has a strong resemblance to the splitting theorem well-known for regular ASP programs. Epistemic splitting can be applied when we can divide an epistemic logic program into a bottom part for a subset U of atoms and a top part, that only refers to atoms in U through subjective literals (those using modal epistemic operators). When this happens, the property of splitting states that we should be able to compute the world views of the program in two steps: first, computing the world views of the bottom and, second, using each bottom world view W to replace subjective literals for atoms in U in the top by their truth value with respect to W.

We have studied several consequences of epistemic splitting: for instance, if the program is stratified with respect to subjective literals then it will have a unique world view, at most. Another consequence is that constraints only consisting of subjective literals will have a monotonic behaviour, ruling out world views that satisfy the constraint body.[6] We have also explored how epistemic splitting may facilitate the simple application of the generate-define-test methodology, well-known in ASP, to the formalisation of conformant planning.

Our study of the main semantics in the literature has shown that only the original semantics (Gelfond, 1991) (G91), and its generalisation (Truszczyński, 2011), satisfy epistemic splitting while the rest of approaches we considered no, as we showed with counterexamples. We do not mean with this, however, that G91 is always intuitive. As it is well-known, G91 suffers from self-supportedness: for instance, the program consisting of the single rule $p \leftarrow \mathbf{K}\, p$ yields two world views $[\emptyset]$ and $[\{p\}]$ but the latter justifies p by the mere assumption of $\mathbf{K}\, p$ without further evidence, something that seems counterintuitive. What we claim instead is that G91 has a reasonable behaviour when subjective literals are stratified. Unfortunately, later attempts to solve self-supportedness on cyclic epistemic specifications have somehow spoiled that feature.

It is also worth to mention that a different notion of splitting for epistemic logic programs was formerly studied in (Watson, 2000). In this work, the splitting of the program is done with respect to the objective literals instead of the subjective ones. As a result, this non-modal kind of splitting can be used to simply the computation of the world views of a program in a different way, but it gives little insight as a means to compare different semantics.

Finally, let us mention that future work should obviously involve the search for a suitable semantics that avoids self-supportedness while preserving the epistemic splitting property as well as other suitable properties that allow us assert that such semantics agree with our intuitive expectations.

References

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming.* Morgan Kaufmann. 89–148.

Fariñas del Cerro, L.; Herzig, A.; and Su, E. I. 2015. Epistemic equilibrium logic. In *Proc. of the*

[6] The lack of monotonicity suffered by epistemic constraints in some semantics has been recently discussed by Lecrerc and Kahl (2018a).

Intl. Joint Conference on Artificial Intelligence (IJCAI'15), 2964–2970. AAAI Press.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of the 5th Intl. Conference on Logic Programming (ICLP'88)*, 1070–1080.

Gelfond, M. 1991. Strong introspection. In Dean, T. L., and McKeown, K., eds., *Proceedings of the AAAI Conference*, volume 1, 386–391. AAAI Press/The MIT Press.

Gelfond, M. 2011. New semantics for epistemic specifications. In *LPNMR*, volume 6645 of *Lecture Notes in Computer Science*, 260–265. Springer.

Kahl, P.; Watson, R.; Balai, E.; Gelfond, M.; and Zhang, Y. 2015. The language of epistemic specifications (refined) including a prototype solver. *Journal of Logic and Computation*.

Kahl, P. T. 2014. *Refining the semantics for epistemic logic programming*. Ph.D. Dissertation.

Lecrerc, A. P., and Kahl, P. T. 2018a. Epistemic logic programs with world view constraints. In *Technical communication, 34th International Conference on Logic Programming (ICLP'2018)*.

Lecrerc, A. P., and Kahl, P. T. 2018b. A survey of advances in epistemic logic program solvers. In *Proc. of the 11th Intl. Workshop on Answer Set Programming and other Computer Paradigms (ASPOCP'18)*.

Lifschitz, V., and Turner, H. 1994. Splitting a logic program. In *Proc. of the Intl. Conference on Logic Programming (ICLP'94)*, 23–37. MIT Press.

Shen, Y., and Eiter, T. 2017. Evaluating epistemic negation in answer set programming (extended abstract). In *Proc. of the Intl. Joint Conference on Artificial Intelligence (IJCAI'17)*, 5060–5064.

Son, T. C.; Le, T.; Kahl, P. T.; and Leclerc, A. P. 2017. On computing world views of epistemic logic programs. In *Proc. of the Intl. Joint Conference on Artificial Intelligence (IJCAI'15)*, 1269–1275. ijcai.org.

Truszczyński, M. 2011. Revisiting epistemic specifications. In *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning*, volume 6565 of *Lecture Notes in Computer Science*, 315–333. Springer.

van Gelder, A. 1988. Negation as failure using tight derivations for general logic programs. In *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann. 149–176.

Watson, R. 2000. A splitting set theorem for epistemic specifications. *CoRR: Proceedings of the 8th International Workshop on Non-Monotonic Reasoning, NMR 2000* cs.AI/0003038.

Causal reasoning in a logic with possible causal process semantics

Marc Denecker† and **Bart Bogaerts**† and **Joost Vennekens**‡
firstname.lastname@cs.kuleuven.be
† KU Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium
‡ KU Leuven, Department of Computer Science, Campus De Nayer, 2860 Sint-Katelijne-Waver, Belgium

Abstract

We point to several kinds of knowledge that play an important role in controversial examples of actual causation. One is knowledge about the causal mechanisms in the domain and the causal processes that result from them. Another is knowledge of what conditions trigger such mechanisms and under what conditions it can be preempted.

We argue that to solve questions of actual causation, such knowledge needs to be made explicit. To this end, we develop a new language in the family of CP-logic, in which causal mechanisms and causal processes are formal objects. We then build a regularity-theoretic framework for actual causation in which various notions of actual causation are defined. Contrary to counterfactual definitions, actual causes are defined directly in terms of the (formal) causal process that causes the possible world.

Introduction

Since the days of Hume (1739), causal reasoning has been an active research domain in philosophy and (later) knowledge representation. With the groundbreaking work of Lewis (1973) and Pearl (2000), the structural equations and counterfactual reasoning approach became mainstream (Halpern and Pearl 2005; Halpern 2016a; Fenton-Glynn 2015; Gerstenberg et al. 2015). But the debate remains intense (Glymour et al. 2010). The counterfactual approach is contested by some (Hall 2004; Baumgartner 2013; Bochman 2018). In many scenarios, there is no agreement of what are the actual causes, and all definitions of actual causation have scenarios where they have been criticized. It shows that the informal notion of actual causation is vague and overloaded with many intuitions; also that many sorts of knowledge influence our judgment of actual causation. Science is not ready yet with unraveling all this.

Of the most striking examples are those where for the same formal causal model, different informal interpretations can be proposed that lead to different actual causes. Such examples are interesting since they are clear cases that *some* relevant knowledge is missing in the causal model. A powerful illustration is given by Halpern (2016b), who discusses 6 causal examples from the literature in which authors had shown (often convincingly) that the actual causation definition of Halpern and Pearl (2005), henceforth called HP, failed to predict the actual causes. He responds by proposing for each example an alternative informal interpretation leading to the *same* structural equation model but to intuitively *different* actual causes which, moreover, are those derived by HP! Halpern concludes that, as far as actual causation goes, the structural equation models are ambiguous. As for what knowledge is missing, he claims:

> "what turns out to arguably be the best way to do the disambiguation is to add [...] extra variables, which [...] capture the **mechanism of causality**".

That is, Halpern argues that it is necessary to make knowledge of causal mechanisms explicit.

That such information is relevant for causal reasoning is not surprising. Almost every causal scenario in the literature comes with an informal specification of causal mechanisms and a *story* specifying which mechanism are active and how they are rigged together in a causal process. As observed before (Glymour et al. 2010; Vennekens 2011), most of this information is abstracted away in structural equations. We illustrate to what problems this may lead with a simple example. Consider two scenarios involving two deadly poisons, arsenic and strychnine. In the first scenario, intake of any of these poisons triggers a specific deadly biochemical process. The structural equation of this scenario is:

$$Dead := Arsenic\_intake \vee Strychnine\_intake$$

If both poisons are taken, this is an instance of overdetermination; HP derives the intuitively correct answer that both poisons are actual causes of death. The second scenario is similar, except that arsenic, in addition to poisoning the victim, also *preempts* the chemical process by which strychnine poisons the victim. Now, the structural equation remains the same (i.e., the victim dies as soon as at least one poison is ingested) and so do the *possible worlds*! However, the judgments of actual causation differ: when both poisons are ingested, only arsenic is a cause of death, since the effects of the strychnine are preempted. The conclusion is that the structural equation correctly predicts the possible worlds but does not contain enough information to explain the actual causes. The missing knowledge is what the separate causal mechanisms are and when they are active.

The following scenario, simplified from **Assassin (Hitchcock 2007)**, illustrates another relevant sort of knowledge that is not expressed in structural equation models. *An assassin may kill a victim by administering deadly poison. A bodyguard may rescue the victim by administering an antidote.* Consider the following structural equation.

$$Dead := Poison\_intake \wedge No\_antidote\_intake$$

While it correctly characterizes the possible worlds of this domain, there is again a problem on the level of actual causes. When only poison is ingested, there is a strong intuition that it is the ingestion of poison that is the actual cause of death, not the absence of antidote. After all, it is the poison that activates the poisoning mechanism, not the absence of antidote. Yet, by the symmetry of the formal model, HP nor any other mathematical method can discover this from the above structural equation. The asymmetry here is that poison *triggers* the causal mechanism, while antidote *preempts* it, i.e., absence of antidote is only a condition to not *preempt* the mechanism. As we will argue below, this distinction plays a role in many controversial causal examples. Such information is missing and should be added to the causal model.

Halperns solution to the above sort of problem is to *reify* the causal mechanism by an auxiliary variable representing whether it *fires*, and incorporating these variables at suitable places in the theory. While his solutions work, there is room here for a complementary approach in which causal mechanisms and their triggers and preemptors are explicit in the causal model.

We proceed as follows. We first define a formal logic to express this knowledge. The logic gives a syntactical and semantical account of causal mechanisms, the causal processes to which they lead, and the possible worlds that these processes produce. Then we define different notions of actual causation in terms of the causal process that causes the possible worlds. We exploit the fact that a causal process gives a precise *explanation* of the possible world that it causes, from which various notions of causation can be "read off". This results in a framework of regularity-theoretic definitions of actual causation. Then, . . . "calculemus!": we evaluate the approach in several examples.

The causal logic: syntax and informal semantics

The logic below is propositional. To represent a causal domain, a *vocabulary* Σ of propositional symbols is to be designed, each expressing a proposition in the domain. As usual, literals over Σ are formulas of the form P or $\neg P$, with $P \in \Sigma$; slightly abusing notation, we use $\neg L$ to denote P if $L = \neg P$ and to denote $\neg P$ if $L = P$.

A causal theory consists of causal laws: abstract representations of causal mechanisms. Each mechanism has triggering conditions, which set the mechanism in operation, no-preemption conditions, which if false, preempt the mechanism, and an effect. This leads to the following definition.

Definition 1. *A* causal law *is a statement of the form:*

$$L \leftarrow T \,||\, C$$

where

- $\leftarrow$ *is the causal operator (not material implication),*
- L *is a literal,*
- T *is a sequence of literals called triggering conditions,*
- C *is a sequence of literals called no-preemption conditions.*

Elements of $T \cup C$ *are called* conditions *of the causal law.*

A causal theory Δ *is a set of causal laws subject to two constraints:*

- Δ *is acyclic, i.e., there exists a strict well-founded order on symbols such that for each causal law, the symbol in the head is strictly larger than each symbol in the body.*
- Δ *does not contain laws* $P \leftarrow \ldots$ *and* $\neg P \leftarrow \ldots$ *for the same symbol* P*.*

The causal logic so far serves to describe causal mechanisms. We extend it to make it suitable to express observations of the world.

Definition 2. *An* extended causal theory *is a set of causal laws and propositional formulas over* Σ*.*

Example 1. Arsenic and Strychnine *The two causal scenarios mentioned in the introduction are represented as follows:*

$$\left\{ \begin{array}{l} Dead \leftarrow Arsenic\_intake \,|| \\ Dead \leftarrow Strychnine\_intake \,|| \end{array} \right\}$$

(both rules have the empty sequence of no-preemption conditions), and

$$\left\{ \begin{array}{l} Dead \leftarrow Arsenic\_intake \,|| \\ Dead \leftarrow Strychnine\_intake \,||\, \neg Arsenic\_intake \end{array} \right\}$$

As for the last rule, strychnine poisoning is triggered by strychnine but preempted by the presence of arsenic.

An extended causal theory is obtained by adding the formula $Dead$ *expressing the observation that the victim died, or the formula* $Arsenic\_intake \vee Strychnine\_intake$ *to express that at least one poison was ingested.*

As usual, we distinguish between *endogenous symbols* (those in the head of laws) and *exogenous* symbols (the rest). A causal theory expresses *all* causal mechanisms affecting an endogenous symbol and it ignores all those of exogenous symbols. The language is not designed for the epistemic state where only part of the causal mechanisms of an endogenous symbol are known.

In a causal law $L \leftarrow T \,||\, C$, T and C represent conjunctions of literals. A causal mechanism that is triggered by a disjunction of events cannot be expressed by a single causal law, but must expressed using multiple causal laws. This is a limitation from a KR point of view, but it was done to simplify the definition of causal process.

A rule $L \leftarrow \,||$ with empty sequences T, C expresses an unconditional causal mechanism causing L.

Definition 3. *A* world W *informally represents a state of affairs; formally, it is a complete and consistent set of literals, i.e., a set of literals such that for each symbol* $P \in \Sigma$*, either* $P \in W$ *or* $\neg P \in W$*, but not both. The* exogenous state *of* W *is the set of its exogenous literals, denoted* $Exo(W)$*.*

Definition 4. *A condition (i.e., a triggering condition or no-preemption condition) K of a causal law r* blocks *r (or is a blocking literal of r) in world W if $\neg K \in W$. We say that r is blocked by K in W. A causal law $L \leftarrow A \,||\, B$ is* active *in world W if $A \subseteq W$, that is, if all its triggering conditions hold in W; otherwise it is inactive. A causal law is* applicable *in W if $A \cup B \subseteq W$. A causal law* is causally preempted *in W if it is active but is blocked by one of its no-preemption conditions in W. A causal law $r = L \leftarrow \ldots$ is* satisfied *in W if it is blocked by some condition, or if it is applicable and its head L holds in W.*

Distinguishing causal mechanisms In many causal domains, properties may be affected by multiple causal mechanisms. E.g., a window may be shattered by one of multiple throws and many more events. A forest fire can be ignited by matches of campers or by lightnings, etc. The modularity principle of knowledge representation suggests to represent such separate objects by separate expressions and so, it is a good thing of the logic that it supports this. Furthermore, this distinction of mechanisms is sometimes needed for determining actual causes, as argued in the introduction.

Even now, before having defined a formal semantics, it is intuitively clear how to transform causal theories to structural equations, namely by *predicate completion* (Clark 1978). E.g., the completion of the first causal theory of **Arsenic and Strychnine** is the propositional logic representation of the structural equation:

$$Dead := Arsenic\_intake \vee Strychnine\_intake$$

The completion of the second theory is syntactically different but logically equivalent.

$$\begin{aligned} Dead := \; & Arsenic\_intake \vee \\ & (Strychnine\_intake \wedge \neg Arsenic\_intake) \end{aligned}$$

The transformation abstracts away the causal mechanisms and the distinction between triggering conditions and no-preemption conditions. We return to this later.

Definition 5. *An endogenous literal L is* deviant *in Δ if it is the head of a causal law. Otherwise L is* default*. A symbol P is in its deviant state in world W if its deviant literal holds in W; otherwise it is in its default state.*

The distinction between deviant and default literal is not made in structural equations but is found in several other formalisms (Hall 2007). The idea is that a symbol is in its default state unless some causal mechanism brings it in its deviant state. A deviant literal L that holds in the world is *caused* by at least one causal mechanism. A default literal L that holds in the world has a *reason*, namely, that every causal mechanism that can cause the deviant literal $\neg L$ is blocked. Either way, the logic implements Leibniz's principle of sufficient reason –that nothing happens in the world without a cause– for deviant as well as for default endogenous literals (but not for exogenous literals).

In many causal domains, causal mechanisms exist that make a property true and others that make it false. E.g., flipping a switch causes the light to be on if the light was not on (off) and vice versa. In our logic, it is not possible to express both mechanisms in the same theory. It is a limitation that, in our opinion, is inherent to the non-temporal nature of the logic. We argue that such combinations of mechanisms are useful mainly in a setting where the truth of propositions fluctuates in time. Like most languages in this area, the logic proposed here is not equipped for modeling such situations.

Triggering conditions versus no-preemption conditions The distinction between triggering conditions and no-preemption conditions of causal mechanisms is a new feature of our logic. Often, a natural distinction can be made between the conditions that set the mechanism in operation and conditions that are necessary for the mechanism to succeed. E.g., to obtain a forest fire, at least three conditions are needed: a forest, a spark igniting a hotbed and absence of extinction operations. It is only the spark (in the form of a lightning or an unsafe camp fire) that triggers the causal mechanism (triggering condition). We argue that this explains the strong intuition shared by many that it is the spark that is the actual cause of the fire, and not the existence of the forest or the absence of fire extinction. We find the same distinction in many examples. The combination of presence of poison in a coffee and drinking it activates a biochemical process that kills the victim (triggering condition) unless blocked by an antidote (no-preemption condition). Suzy's throw at a window activates a process that shatters the window (triggering condition) if her shoulder is not soar (which prevents throwing hard enough), if the stone is not intercepted, if the window is still intact (no-preemption conditions). In each case, we perceive a distinction between triggering conditions and no-preemption conditions. This appears to affect our judgment of the actual causes: in at least one view of actual causation, the triggering conditions are actual causes of the effect, while the no-preemption conditions are not. Hence, to derive this notion of actual causation, the nature of the conditions must be clear from the causal theory.

Example 2. *(***Drinking poisoned coffee***, see (Hitchcock 2007)) Drinking poisoned coffee causes death unless an antidote is administered. There are three conditions here: presence of poison in the coffee ($Poison$), drinking the coffee ($Drink$), and absence of antidote ($\neg Antidote$). It is clear that $\neg Antidote$ is a no-preemption condition but what about $Poison$? The poisoning process is physically triggered by the event of drinking; the poisoning of coffee could have taken place long before. How could $Poison$ be a triggering condition then? On the other hand, it is still the intake of poison that triggers the poisoning process. This condition, in the chosen vocabulary, is best represented by the conjunction of $Drink$ and $Poison$. So, we argue for the following representation:*

$$\neg Alive \leftarrow Drink, Poison \,||\, \neg Antidote$$

Under this representation, it will be derived that $Drink$ and $Poison$ are actual causes of $\neg Alive$, but $\neg Antidote$ is not.

The above example raises a subtle concern. The scientific goal of actual causation research is to find methods to solve

actual causation problems by *deriving* actual causes and preemptions. But in Example 2, one might get the impression that we are directly *encoding* the desired solution of the actual causation problem in the causal model. However, this is not what we do. In the above example, as in many other causal domains, there is a strong and clear intuition of what triggers causal mechanisms and what may preempt them. The distinction is independent of the specific actual causation problem and is relevant in many different actual causation problems. E.g., in **Drinking poisoned coffee** (Ex.2), the fact that $Drink$ and $Poison$ are triggering conditions and $\neg Antidote$ a no-preemption condition is relevant not only for determining the actual causes of death when all conditions hold; it is relevant as well in the seven other exogenous contexts. E.g., in a context where the victim drinks unpoisoned coffee and ingests an antidote, the common intuition is his survival is caused by the absence of poison, not the presence of antidote. In more complex causal models with many variables and causal laws, the information about triggering conditions or no-preemption conditions of one mechanism may influence actual causation of many variables in an exponential number of exogenous states. This can be seen in double preemption examples.

Formal semantics: causal processes and possible worlds

The formal semantics specifies for each causal theory Δ its causal processes and the world that each process leads to. Causal processes can be formalized in multiple ways. Vennekens, Denecker, and Bruynooghe (2009) formalize it as a sequence of states in which at every state one causal laws is applied until all causal laws are satisfied. This representation is precise and gives an account of, e.g., the "stories" in many causal examples. However, for explaining actual causes, it is a bit too detailed. E.g., it fixes the order of application of causal mechanisms which is largely irrelevant for determining actual causes. So, we opt to formalize a process as an acyclic dependency graph of the firing causal laws.

Definition 6. *A possible causal process for Δ is a directed labeled graph $\mathcal{P}$ on a world denoted $World(\mathcal{P})$. Each arc from literal K to literal L is labeled with a rule r and is denoted $L \overset{r}{\leftarrow} K$ (or $K \overset{r}{\rightarrow} L$). The graph satisfies the following conditions:*

- *For each deviant endogenous literal $L \in World(\mathcal{P})$, there exists a nonempty set F_L of applicable rules with head L, called the* firing rules *of L, such that for each condition K of each rule $r \in F_L$, there is an arc $L \overset{r}{\leftarrow} K$. There are no other arcs to L. We call them* active arcs *and distinguish between* trigger arcs *and* no-preemption arcs *depending on the type of the condition K in r.*
- *For each default endogenous literal $L \in World(\mathcal{P})$, for each rule $r = \neg L \leftarrow \ldots$, the set B_r of blocking conditions of r in $World(W)$ is non-empty and there is an arc $\neg L \overset{r}{\leftarrow} \neg K \in \mathcal{P}$ for each $K \in B_r$. There are no other arcs to L. We call such arcs* blocking arcs *and we distinguish between* no-trigger arcs *and* preemption arcs.

The leafs of a causal process are exactly the true exogenous literals of the world; the non-leafs are the true endogenous literals.

We observe that causal processes can have multiple "simultaneous" causal mechanisms causing the same deviant literal L. That is, L's firing set F_L may contain more than one rule. This is needed to model *overdetermination*.

The active arcs in a causal process reflect the conditions that helped to trigger a causal mechanism causing a deviant literal. The blocking arcs for a rule r reflect all conditions that prevented a causal mechanism to be applicable. A false deviant literal L has at least one blocking arc for every causal mechanism that could cause L.

The difference between triggering conditions or no-preemption conditions in a causal theory barely affects the causal processes and is only visible in the classification of the arcs (trigger, no-preemption, no-trigger or preemption arcs). These labels do not play a role in determining the possible world that the process causes but they will play a key role in the definition(s) of actual causation.

Definition 7. *A causal process $\mathcal{P}$* realizes *world W if $W = World(\mathcal{P})$. We call W a possible world of Δ if it is realized by some causal process for Δ.*

The above notions generalize naturally to extended causal theories. A process $\mathcal{P}$ is a possible causal process for an extended causal theory Δ if it is a causal process for the set of causal laws in Δ and $World(\mathcal{P})$ satisfies the propositional formulas of Δ.

A possible world semantics induces the notions of satisfiability and entailment.

Definition 8. *We say that an (extended) causal theory Δ is* satisfiable *if it has at least one possible world. It* logically entails *a propositional formula φ if φ is true in every possible world of Δ.*

Definition 9. *We say that a causal mechanism r is* chronologically preempted *in $\mathcal{P}$ if it is applicable but does not belong to the fire set of its effect.*

Example 3. *(***Drinking poisoned coffee, cont.***) Each of the eight exogenous states of this causal theory determines a unique process. E.g., the context $\{Drink, Poison, \neg Antidote\}$ is the only context in which the victim dies. The causal law is active and fires and $\neg Alive$ has incoming trigger arcs from $Poison$ and $Drink$ and a no-preemption arc from $\neg Antidote$. In context $\{Drink, Poison, Antidote\}$, the law is active but preempted; $Alive$ has an incoming preemption arc from $Antidote$. In $\{\neg Drink, \neg Poison, Antidote\}$, the rule is inactive and $Alive$ has no-trigger arcs from $\neg Drink, \neg Poison$ and a preemption arc from $Antidote$. The latter context corresponds to* **Bogus Prevention** *(Hiddleston 2005; Hall 2007).*

In many preemption examples in the literature, the preemption is due to causal mechanisms that are triggered but fail. In our framework, this corresponds to *causal preemption*. Often, distinction is made between *early* and *late preemption*. E.g., in case of a lightening striking forest ground, the absence of trees (burnt by a previous fire, or cut down by a lumber company) is an early preempter of a forest

fire, while an extinction operation is a late preempter. In our framework, there is no general way to formally distinguish between early and late (causal) preemption, since the process semantics makes abstraction of the order of events.

Beside causal preemption, there is a second sort of preemption. Even when a causal mechanism r with effect L is applicable in world W, that is, all its conditions hold, it is possible that r does not fire. Intuitively, this corresponds to the situation when other causal mechanisms had caused L before r got the chance. We say that r is *chronologically preempted* in $\mathcal{P}$.

Example 4. *(***Window***, see (Hall 2004)) Suzy and Billy throw rocks at a window. Each throw is a separate causal mechanism causing the same deviant state of a broken window. We represent as follows:*

$$\left\{ \begin{array}{l} Broken \leftarrow SuzyT \,||\, \\ Broken \leftarrow BillyT \,||\, \end{array} \right\}$$

Assume that both throw, in which case the window will certainly break. In the overdetermination *scenario, they hit the window simultaneously. It corresponds to the causal process in which the fire set of $Broken$ contains both laws. In the* late preemption *scenario, Suzy's throw arrives first and smashes the window. It corresponds to the process in which only the first law belongs to the fire set of $Broken$. It is called here a case of* chronological preemption*. Observe that for the resulting world, this does not matter: the window is broken. Stated precisely, in the exogenous state $\{SuzyT, BillyT\}$, there are multiple possible causal processes. However, they are confluent: they lead to the same possible world.*

Adding firing information Several other sorts of knowledge have been claimed to influence our judgment of actual causes, e.g., whether a proposition is *normal* (like the presence of oxygen in the air), whether a proposition represents an intentional action of an agent, whether a causal mechanism fires. Such knowledge stands orthogonal to our approach; the language can be extended to express it. We illustrate this for knowledge about firing. We assume that causal laws in a theory Δ have a symbolic name, declared in expressions of the kind:

$$(BillyBreaks :) \quad Broken \leftarrow BillyT \,||$$

An extended causal theory is then a set of named causal laws and Boolean expressions ψ of symbols of Σ and atomic formulas $Fires(r)$ with r a name of one of the causal laws. Given a causal process $\mathcal{P}$, we define $\mathcal{P} \models \psi$ by the standard inductive rules for connectives and by the base rules that $\mathcal{P} \models p$ if $p \in World(\mathcal{P})$ and that $\mathcal{P} \models Fires(r)$ if r fires in $\mathcal{P}$.

Example 5. Window, late preemption, cont. *Suppose Billy's throw is chronologically preempted by Suzy's. In structural equations, it is frequent practice to encode this information by adding auxiliary variables and changing the structural equations (Halpern 2016b). But such knowledge is independent of the workings of causal mechanisms; it should better be expressed separately. In our logic, it amounts to knowledge that Billy's mechanism does not fire. It is expressed as:*

$$Broken \wedge \neg Fires(BillyBreaks)$$

The causal model extended with this proposition logically entails $Fires(SuzyBreaks)$.

(Fundamental) properties of causal knowledge

We first establish the link with structural equations. Recall that *predicate completion* (Clark 1978) transforms a causal theory Δ in a set $compl(\Delta)$ of structural equations.

Theorem 1. *The possible worlds of a causal theory Δ and the solutions of the structural equation model $compl(\Delta)$ are identical.*

The theorem gives an indication for the success of structural equations for causal reasoning even if they do not model informal key concepts of causation such as causal mechanisms and causal processes: the (many) problems that can be solved on the basis of the possible worlds of the theory (and of the variant theories obtained with interventions), can be solved using $compl(\Delta)$.

Proposition 1. *A causal process for Δ is uniquely determined by the set of its exogenous literals and firing rules. That is, two different processes differ on some exogenous literal or on the set of causal mechanisms that fire.*

Theorem 2. *Given a causal theory Δ, each exogenous state W_{exp} can be uniquely extended to a possible world of Δ. Thus, Δ is satisfiable in each exogenous state, and two different possible worlds of Δ differ on some exogenous literals.*

Theorem 3. *All causal processes of Δ in exogenous state W_{exo} realize the same world.*

The latter is a confluence theorem. It is one of these aspects that are brought to the surface by making the causal processes explicit. It tells something important about causal information. For a given exogenous state, it does not matter which of the rules are applied nor in what order they are applied: they will always result in the same world. This point was made in Vennekens, Denecker, and Bruynooghe (2009). A real world case would be that I send one friend the message that I won the lotto, and by the end of the day, I can be certain that all my friends know that I am rich. The process, the details of who tells who, may vary widely and is unknown to me; yet the outcome is predictable. It points to a valuable property of causal information: that it allows to derive much information about the state of the world that is the result of a causal process, even in the absence of almost any information on the process itself.

Proposition 2. *The causal language is non-monotonic: a world that is impossible in a causal theory Δ may be possible in an extension of Δ obtained by adding other causal laws to it.*

For a proof, consider the causal theory $\{ \; P \leftarrow Q \,||\, \}$. An impossible world is $\{P, \neg Q\}$. This world is possible after adding $P \leftarrow \neg Q \,||$. The original theory entails $\neg Q \Rightarrow \neg P$ while its extension does not.

Definitions of actual causation

A causal process $\mathcal{P}$ realizing world W provides a precise causal explanation of W from which different notions of causation can be "read off". Below it is used as a framework to define several notions of actual causation.

Definition 10. *A literal L is an influence of K in a possible causal process $\mathcal{P}$ of Δ if there is a path from K to L in $\mathcal{P}$.*

The concept of influence is useful but weak. We refine it to take the difference between triggering conditions and no-preemption conditions into account. When a causal mechanism fires and causes L, only its triggering conditions are seen as actual causes. E.g., when **Drinking poisoned coffee** without taking an antidote, drinking poisoned coffee is the actual cause of death, not the absence of antidote. Also, one cannot preempt a causal mechanism that has not been triggered, hence, when a causal mechanism to derive L remains inactive by a false triggering condition, its false no-preemption conditions are not actual causes of $\neg L$. E.g., when the victim takes the antidote but does not drink the poisoned coffee, the actual cause for survival is the absence of drinking, not the antidote. Only if the mechanism is active, will a false no-preemption condition be an actual cause of $\neg L$. E.g., the antidote is an actual cause of survival only if the victim drinks poisoned coffee.

Implementing these intuitions is easy: it suffices to discard all causal paths containing an arc $L \stackrel{r}{\rightarrow} K$ that is a no-preemption arc of a firing mechanism r or that is a preemption arc of a non-active causal mechanism r.

Definition 11. *A literal L is an actual P-cause of literal K in process $\mathcal{P}$ if there is a path $K \rightarrow \ldots \rightarrow L$ in $\mathcal{P}$ without no-preemption arcs and without preemption arcs of non-active causal mechanisms. Such path consists of trigger and no-trigger arcs, and preemption arcs of active causal mechanisms.*

The "P" stands for "preemption". Our both notions of causation are defined in the context of a causal process, whereas in most approaches actual causes are defined in the context of a possible world. We bridge this gap.

Definition 12. *A literal K is an influence (actual P-cause) of L in a possible world W of Δ if there is a possible causal process $\mathcal{P}$ realizing W such that K is an influence (actual P-cause) of L in $\mathcal{P}$. We call K a* definite *influence (actual P-cause) of L in W if it is an influence (actual P-cause) in every causal process realizing W. Otherwise it is called* speculative.

As pointed out by Vennekens (2011), even when we know the world, we may not know how it was caused and therefore, we may not be sure about the actual causes. This uncertainty is reflected in the above definition. It is illustrated by the different possible causal processes of **Window** in the exogenous context where both Suzy and Billy throw. It is one of these aspects that are brought to the surface by making the causal processes explicit.

Proposition 3. *The notions of influence and actual P-cause in processes and worlds are anti-symmetric and transitive.*

Now we turn to examples. The ones seen so far (**Arsenic and Strychnine, Drinking poisoning coffee** and **Window**) are modeled by simple causal theories having causal processes of depth 1. It is straightforward to derive the possible causal processes and the influences and actual P-causes of the endogenous literal. Moreover, as can be seen in the discussion preceding Definition 11, the results match the intuitions expressed in the introduction.

Example 6. *(***Backup (Hitchcock 2007) (early preemption versus switch)***)* A crime syndicate hires Assassin to poison victim's coffee who drinks it and dies. The syndicate had hired Backup to watch Assassin and to poison the victim in case Assassin would not poison the coffee. Backup did not have to intervene. *This scenario is a case of early preemption (of the poisoning by Backup). Three causal mechanisms can be discerned. They are represented:*

$$\left\{ \begin{array}{l} Dead \leftarrow APoison \,||\\ Dead \leftarrow BPoison \,||\\ BPoison \leftarrow \neg APoison \,|| \end{array} \right\}$$

The informal scenario corresponds to the context $\{APoison\}$, where the only actual P-cause of $Dead$ is $APoison$. This is the same answer as in (Bochman 2018) but certain counterfactual methods do not return $APoison$ as an actual cause (Lewis 1973; Halpern 2016a). In the context $\{\neg APoison\}$, the actual P-causes of $Dead$ are $BPoison$ and $\neg APoison$. That $\neg APoison$ is an actual P-cause is slightly disconcerting; perhaps this has to do with the longer length of the causal path from $\neg APoison$ to $Dead$. Still, we feel it makes sense, since the fact that Assassin does not poison, sets Backup's mechanism in motion to poison the victims coffee.

Now take an alternative story: the crime syndicate hires both Assassin and Backup, with a similar task: to pickup a poison at (the same) hidden place and poison victim. Assassin is ordered to go to the hiding place on Monday, Backup on Tuesday. The syndicate puts one potion of poison in the location on Sunday. *We argue that in this scenario, the causal laws are the same except for:*

$$BPoison \leftarrow || \neg APoison$$

Here backup has the plan to poison, but may be preempted to do so if Assassin took the poison. In the context $\{APoison\}$, the causes are identical as in the previous story. But in $\{\neg APoison\}$, only $BPoison$ is an actual P-cause and not $\neg APoison$. We feel this makes sense; after all, it was not $\neg APoison$ that triggered Backup to poison the victim, so how could it be a cause for $Dead$?

In both scenarios, $APoison$ is counterfactually irrelevant: whether true or false, the victim dies. The first is an early preemption scenario, the second is more like a **switch** *scenario, with $APoison$ as switch. It has been a challenge to explain the difference between early preemption and switch. This example suggests the underlying problem might be the distinction between triggering conditions and no-preemption conditions.*

Example 7. Double Preemption (Hall 2004) *Double preemption occurs when a potential preempter is preempted. It*

occurs in the following scenario. Suzy fires a missile (SF) to bomb target (B); enemy fires a missile (EF) to hit Suzy's missile (SMH) and Billy fires a missile (BF) to hit Enemy's missile (EMH). *We see three causal mechanisms:*

$$\left\{ \begin{array}{l} B \leftarrow SF \,||\, \neg SMH \\ SMH \leftarrow EF \,||\, \neg EMH \\ EMH \leftarrow BF \,|| \end{array} \right\}$$

In the causal process of context $\{SF, EF, BF\}$, the target is bombed. We find the causal path $BF \rightarrow EMH \rightarrow \neg SMH \rightarrow B$ which in the two last edges display a double preemption: the hit on enemy's missile preempts enemy's attempt at preempting Suzy's bombing.

We broaden the notion of actual P-cause to include double preemption. In our setting, a double preemption path is a causal path $K \stackrel{r}{\rightarrow} L_0 \rightarrow \ldots \rightarrow L_n \stackrel{r'}{\rightarrow} L$ ($n \geq 0$) such that

- $K \stackrel{r}{\rightarrow} L_0$ is a preemption arc of a causally preempted law r. In the example, it is the arc $EMH \rightarrow \neg SMH$.
- The n arcs $L_i \rightarrow L_{i+1}$ are arcs proving that L_0 is an actual P-cause of L_n (or identical to it). In the example, $n = 0$ and $L_0 = L_n$.
- $L_n \stackrel{r'}{\rightarrow} L$ is a no-preemption arc of a firing causal law r'. In the example, it is the arc $\neg SMH \rightarrow B$.

Definition 13. *The DP-causal graph of $\mathcal{P}$ consists of all arcs considered for actual P-causes augmented with* double preemption arcs $K \Rightarrow L$ *for every double-preemption path from K to L in $\mathcal{P}$. A literal K is an actual DP-cause of literal L in process $\mathcal{P}$ if there is a path from K to L in the DP-causal graph of $\mathcal{P}$.*

"DP" stands for double preemption. Billy's fire BF in **Double preemption** is a actual DP-cause of B although it is not an actual P-cause of B.

Proposition 4. *The actual P-causes are actual DP-causes; actual DP-causes are influences. The actual DP-cause relation is anti-symmetric and transitive.*

Example 8. *(***Triple preemption***) The new definition deals with triple preemption and more. Consider* **Double preemption** *extended with Jane who fires at Billy's missile (JF).*

$$\left\{ \begin{array}{l} B \leftarrow SF \,||\, \neg SMH \\ SMH \leftarrow EF \,||\, \neg EMH \\ EMH \leftarrow BF \,||\, \neg BMH \\ BMH \leftarrow JF \end{array} \right\}$$

In state $\{SF, EF, BF, JF\}$, the city is not bombed. The actual DP-causes of $\neg B$ are EF, BMH, JF. The DP-causal graph contains the double preemption arc $BMH \Rightarrow SMH$ which is induced by the double preemption path $BMH \rightarrow \neg EMH \rightarrow SMH$. Since Jane's fire JF is an actual P-cause of BMH, it is an actual DP-cause for the failed bombing $\neg B$.

Counterfactual dependence

In the first causal theory of **Backup**, Example 6, $APoison$ is an actual P-cause of $Dead$ in the exogeneous state $\{APoison\}$. We observed that $APoison$ was counterfactually irrelevant to $Dead$ in the sense that even if $APoison$ would have been false, $Dead$ would have been true all the same. In this section, we formally define this notion of counterfactual relevance.

Ever since the seminal work of Lewis (1973), actual causation has been analyzed using counterfactual reasoning. While our theory of actual causation does not rely on counterfactual reasoning, in many applications, counterfactual questions naturally arise. E.g., in **Window** "would the window have broken had Suzy not thrown?". Or, in **Backup**, "would the victim have died if Assassin had not poisoned him". Below, we define the notions of counterfactual (in)dependency and (ir)relevance using the concept of intervention, as introduced by Pearl (2000). We adapt the definition of intervention from (Vennekens, Denecker, and Bruynooghe 2010), where it was defined in the context of CP-logic.

Definition 14. *We define the* intervention *of causal mechanism r on causal theory Δ (denoted $\Delta[r]$) as the causal theory obtained from Δ by deleting all rules with the same head as r (if any) and adding r.*

Observe that the unique possible world of Δ extending exogeneous state W_{exo} is the unique possible world of $D + W_{exo}$ which is Δ extended with causal rules $L \leftarrow ||$ for each $L \in W_{exo}$.

Definition 15. *Let Δ be a causal theory without exogeneous symbols. Given that K, L are true in the unique possible world W of Δ, we define that L is counterfactually dependent on K according to Δ if L is false in the possible world of $\Delta[K \leftarrow \mathbf{f}]$.*

If Δ has exogeneous symbols and W is a possible world of Δ, we define that L is counterfactually dependent on K in W according to Δ if L is counterfactually dependent on K according to $\Delta + Exo(W)$.

If L counterfactually depends on K, we say also that K is counterfactually relevant for L.

Proposition 5. *If K is counterfactually relevant for L in W according to Δ then K is an influence of L in W.*

The inverse is not true, as can be seen in **Backup**.

Example 9. *(***Backup, cont.***) The causal theory corresponding to* **Backup** *in exogeneous state $W_{exo} = \{APoison\}$ is:*

$$\left\{ \begin{array}{l} Dead \leftarrow APoison \,|| \\ Dead \leftarrow BPoison \,|| \\ BPoison \leftarrow \neg APoison \,|| \\ APoison \leftarrow || \end{array} \right\}$$

According to this theory as well as the intervention by $\neg APoison \leftarrow ||$, $Dead$ is true. It follows that $APoison$ is counterfactually irrelevant for $Dead$. In terms of the original theory Δ, $APoison$ is a counterfactually irrelevant influence of $Dead$ in W_{exo}.

It can be seen that in context $\{SuzyT, BillyT\}$ of **Windows**, Suzy's throw is counterfactually irrelevant for $Broken$. Still, there is difference with the Assassins poisoning: if Suzy does not throw, then she does not actively contribute to breaking the window. But if Assassin does not poison, this causes the Backup to poison the victim, and so,

either way, Assassins choice is an actual cause of the victims death. The following definition expresses this stronger notion of irrelevance.

Definition 16. *Given Δ without exogeneous symbols, we define that K is a strongly irrelevant actual P-cause of L according to Δ if K is an actual P-cause of L in the world of Δ and $\neg K$ is an actual P-cause of L in the world of $\Delta[K \leftarrow \mathbf{f}]$.*

In general, if Δ has exogeneous symbols and W is a possible world of Δ, we define that K is a strongly irrelevant actual P-cause of L in W according to Δ if K is a strongly irrelevant actual P-cause of L according to $\Delta + Exo(W)$.

We see that $APoison$ is a strongly irrelevant actual P-cause of $Dead$ while $SuzyT$ is counterfactually irrelevant but not a strongly irrelevant actual P-cause of $Broken$.

We have a strong intuition not to consider strongly irrelevant actual P-causes such as $APoison$ as actual causes whereas according to our intuition, other actual P-causes such as $SuzyT$ are actual causes even if they are counterfactually irrelevant.

We argue that it is an asset of our approach that it helps distinguishing between *inherently* counterfactual questions about actual causation (such as "would the window have broken if Suzy had not thrown"), and the use of counterfactual analysis to exhume actual causes from a modelling that does not express the causal process.

Related work and conclusions

In the spectrum of counterfactual versus regularity-theoretic approaches to actual causation, our method belongs to the second category since it is based on analysis of causation in the actual world and the actual causal process.

Counterfactual methods originated from Lewis' idea of interpreting "C caused E" as the statement "without C, E would not have been". When counterexamples kept emerging, ever more sophisticated counterfactual strategies were developed. Present-day methods derive actual causes from "blackbox" theories in a way that seems to mimic an empirical scientist who, perhaps without knowledge of the causal mechanisms in the domain, tries to discover actual causes by a strategy of experiments in which the values of well-chosen variables are varied.

Our definition of actual causes defines actual causes in terms of the dependencies shown in the actual process. That leads to a very different definition than the counterfactual definitions of actual causation. Still we do not think that they contradict with each other, but rather than that they point at complementary aspects of the same thing.

Actual causation is an informal concept, which each of us learns through experience and communication with other people. It is not a concept that we acquire by receiving a definition. In such cases, it is well possible that two very different definitions of a concept cover to a large extend the same set of phenomena. We think this is the case with the counterfactual definitions of actual causation, and the dependency-based definitions that we used. In fact, there is a clue to this in the very concept of "dependency": if A depends on B, then we expect that if B is not, then A might not be, and this suggests a counterfactual dependency.

It should not be easy to reconstruct the exact factual dependencies using counterfactual experiments. Sometimes, "nature" puts an effort to hide certain dependencies, and no experiment can bring a dependency to the surface. It suggests that counterfactual definitions and dependency definitions do not perfectly match. Nevertheless, we expect this to be an exception; we expect that in many cases, counterfactual experiments are able to bring a dependency to the surface. This raises a research question: what is the correspondence between actual causes in causal theories Δ defined in terms of the actual causal process, and actual causes derived through counterfactual methods from $comp(\Delta)$. This is a topic for future work.

In this framework, we studied several sorts of knowledge that are important for actual causation but are not or not well expressed in many causal languages: knowledge of causal mechanisms, triggering versus preempting conditions, and whether they fire. We proposed a causal logic suitable for modular expression of such knowledge and equipped with a possible causal process semantics. The explicit modeling of causal processes brought a few fundamental aspects of causation to the surface: e.g., the convergence of causal processes and, paradoxically, theorems explaining why many useful causation problems can be solved without modelling mechanisms and processes. Using causal processes as an explanation of the world, we provided definitions for several notions of actual causation including double preemption. Further analysis is required to corroborate and refine these results, but the method handles a range of problematic examples in causal reasoning.

The aim to study actual causation in the context of causal processes is present in neuron diagrams approaches (Lewis 1986). However, neuron diagrams do not represent individual causal mechanisms (similar to a structural equation) and do not distinguish between triggering and preempting conditions, and fall short for the sort of examples that motivated this paper. The first causal reasoning study in a language that accounts for causal mechanisms, processes and worlds was (Vennekens, Denecker, and Bruynooghe 2009). The language CP-logic was used for different forms of reasoning such as probabilistic reasoning, interventions and actual causation. The logic defined here is related in spirit to CP-logic but differs from it quite considerably. E.g., causal processes are formalized differently, and several sorts of knowledge studied here cannot be expressed in CP-logic (and vice versa). The actual causation method for CP-logic proposed by Vennekens (2011) and refined by Beckers and Vennekens (2012) is based on causal processes as well, but it is intuitively and mathematically completely different. It is a counterfactual method based on analysis of alternative causal processes, in a way related to the approaches of Hall (2004; 2007). The relation with our approach is not obvious and we leave a further analysis of this for future work.

Our formalism is simple and propositional. To make it suitable to express real-world causal domains, it needs to be extended to the predicate case, with quantification, formulas in the body, the possibility to define auxiliary concepts, non-deterministic causation, probabilities, cyclic causation, etc. such that definitions of actual causation still work. This

is for future work. CP-logic covers already most of these extensions, so we expect this to be feasible.

Implementation

We specified the different notions of causality of this paper as a first order logic theory using the knowledge base system IDP (De Cat et al. 2016). Our model is available at `http://adams.cs.kuleuven.be/idp/server.html?chapter=intro/11-AC`. By applying model expansion inference on this specification and on structures encoding causal theories, various notions of actual causation are computed. The webpage contains all examples of the paper and several others. Readers can modify these examples or edit their own and run the system in the web browser to solve causal questions.

Acknowledgements

We are grateful to Alexander Bochman and Sander Beckers for many discussions and valuable feedback. Bart Bogaerts is a postdoctoral fellow of the Research Foundation – Flanders (FWO).

References

Baumgartner, M. 2013. A regularity theoretic approach to actual causation. *Erkenn* 78(Suppl 1:85.

Beckers, S., and Vennekens, J. 2012. Counterfactual dependency and actual causation in CP-logic and structural models: A comparison. In Kersting, K., and Toussaint, M., eds., *Proceedings of the Sixth Starting AI Researchers Symposium, STAIRS, Montpellier, 27-28 August 2012*, volume 241, 35–46.

Bochman, A. 2018. Actual causality in a logical setting. In *IJCAI*.

Clark, K. L. 1978. Negation as failure. In *Logic and Data Bases*, 293–322. Plenum Press.

De Cat, B.; Bogaerts, B.; Bruynooghe, M.; Janssens, G.; and Denecker, M. 2016. Predicate logic as a modelling language: The IDP system. *CoRR* abs/1401.6312v2.

Fenton-Glynn, L. 2015. A proposed probabilistic extension of the halpern and pearl definition of 'actual cause'. *The British Journal for the Philosophy of Science*.

Gerstenberg, T.; Goodman, N. D.; Lagnado, D. A.; and Tenenbaum, J. B. 2015. How, whether, why: Causal judgments as counterfactual contrasts. In *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, 782–787.

Glymour, C.; Danks, D.; Glymour, B.; Eberhardt, F.; Ramsey, J.; Scheines, R.; Spirtes, P.; Teng, C. M.; and Zhang, J. 2010. Actual causation: a stone soup essay. *Synthese* 175(2):169–192.

Hall, N. 2004. Two concepts of causation. In *Causation and Counterfactuals*.

Hall, N. 2007. Structural equations and causation. *Philosophical Studies* 132(1):109–136.

Halpern, J., and Pearl, J. 2005. Causes and explanations: A structural-model approach. part i: Causes. *The British Journal for the Philosophy of Science* 56:843–87.

Halpern, J. 2016a. *Actual causality*. MIT Press.

Halpern, J. Y. 2016b. Appropriate causal models and the stability of causation. *Rew. Symb. Logic* 9(1):76–102.

Hiddleston, E. 2005. A causal theory of counterfactuals. *Noûs* 39(4):632–657.

Hitchcock, C. 2007. Prevention, preemption, and the principle of sufficient reason. *Philosophical Review* 116(4):495–532.

Hume, D. 1739. *A Treatise of Human Nature*. John Noon.

Lewis, D. 1973. Causation. *Journal of Philosophy* 70:113–126.

Lewis, D. 1986. Postscripts to 'causation'. In Lewis, D., ed., *Philosophical Papers Vol. Ii*. Oxford University Press.

Pearl, J. 2000. *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Vennekens, J.; Denecker, M.; and Bruynooghe, M. 2009. CP-logic: A language of causal probabilistic events and its relation to logic programming. *TPLP* 9(3):245–308.

Vennekens, J.; Denecker, M.; and Bruynooghe, M. 2010. Embracing events in causal modelling: Interventions and counterfactuals in CP-logic. In *JELIA*, 313–325.

Vennekens, J. 2011. Actual causation in cp-logic. *Theory and Practice of Logic Programming* 11:647–662.

Relating Two Dialects of Answer Set Programming

Amelia Harrison and **Vladimir Lifschitz**
University of Texas at Austin
{ameliaj,vl}@cs.utexas.edu

Abstract

The input language of the answer set solver CLINGO is based on the definition of a stable model proposed by Paolo Ferraris. The semantics of the ASP-Core language, developed by the ASP Standardization Working Group, uses the approach to stable models due to Wolfgang Faber, Nicola Leone, and Gerald Pfeifer. The two languages are based on different versions of the stable model semantics, and the ASP-Core document requires, "for the sake of an uncontroversial semantics," that programs avoid the use of recursion through aggregates. In this paper we prove that the absence of recursion through aggregates does indeed guarantee the equivalence between the two versions of the stable model semantics, and show how that requirement can be relaxed without violating the equivalence property.

Introduction

Early work on autoepistemic logic and default logic has led to the development of the stable model semantics of logic programs, which serves as the semantic basis of answer set programming (ASP). The ASP-Core document[1], produced in 2012–2015 by the ASP Standardization Working Group, was intended as a specification for the behavior of answer set programming systems. The existence of such a specification enables system comparisons and competitions to evaluate such systems.

The semantics of ASP programs described in that document differs from that of the input language of the widely used answer set solver CLINGO.[2] The two languages are based on different versions of the stable model semantics: the former on the FLP-semantics, proposed by Faber, Leone, and Pfeifer (2004) and generalized to arbitrary propositional formulas by Truszczynski (2010), and the latter on the approach of Ferraris (2005).

In view of this discrepancy, the ASP-Core document includes a warning: "For the sake of an uncontroversial semantics, we require [the use of] aggregates to be non-recursive" (Section 6.3 of Version 2.03c). Including this warning was apparently motivated by the belief that in the absence of recursion through aggregates the functionality of CLINGO conforms with the ASP-Core semantics.

In this paper, that belief is turned into a theorem: for a programming language that is essentially a large subset of ASP-Core,[3] we prove that the absence of recursion through aggregates guarantees the equivalence between ASP-Core and CLINGO. Our theorem is actually stronger, in two ways. First, it shows that the view of recursion through aggregates adopted in the ASP-Core document is unnecessarily broad when applied to disjunctive programs (see Footnote 9). Second, it shows that aggregates that do not contain negation as failure can be used recursively without violating that property. For example, the rule

```
val(W,0) :- gate(G,and), output(W,G),
    card{W: val(W,0), input(W,G)} > 0
```

which describes the propagation of binary signals through an and-gate (Gelfond and Zhang, 2014, Example 9) has the same meaning in both languages.

A few years ago it was difficult not only to prove such a theorem, but even to state it properly, because a mathematically precise semantics of the language of CLINGO became available only with the publication by Gebser et al. (2015). The concept of a stable model for CLINGO programs is defined in that paper in two steps: first a transformation τ is introduced,[4] which turns a CLINGO program into a set of infinitary propositional formulas, and then the definition of a stable model due to Ferraris (2005), extended to the infinitary case by Truszczynski (2012), is invoked. Infinite conjunctions and disjunctions are needed when the program uses local variables. We will refer to stable models in the sense of this two-step definition as "FT-stable."

The semantics of ASP-Core programs is precisely defined in Section 2 of the ASP-Core document, but that definition is not completely satisfactory: it is not applicable to programs

[1] https://www.mat.unical.it/aspcomp2013/ASPStandardization.

[2] http://potassco.org/clingo.

[3] This language does not include classical negation, weak constraints, optimize statements, and queries, and it does not allow multiple aggregate elements within the same aggregate atom. On the other hand, it includes the symbols *inf* and *sup* from the CLINGO language.

[4] An oversight in the definiton of τ in that publication is corrected in the arXiv version of the paper, http://arXiv.org/abs/1507.06576v2.

with local variables. The problem is that the definition of a ground instance in Section 2.2 of the document includes replacing the list $e_1; \dots; e_n$ of aggregate elements in an aggregate atom by its instantiation $\text{inst}(\{e_1; \dots; e_n\})$; the instantiation, as defined in the document, is an infinite object, because the set of symbols that can be substituted for local variables includes arbitrary integers and arbitrarily long symbolic constants. So the result of the replacement is not an ASP-Core program. Prior to addressing the main topic of this note, we propose a way to correct this defect. We use a two-step procedure, similar to the one employed by Gebser et al. (2015): after applying a transformation τ_1, almost identical to τ,[5] it refers to a straightforward generalization of the definition of a stable model due to Faber, Leone, and Pfeifer (2004) to the infinitary case. In the absence of local variables, this semantics is consistent with the ASP-Core document (Harrison, 2017, Chapter 12). Stable models in the sense of this two-step definition will be called "FLP-stable."

We start by defining the syntax of programs, two versions of the stable model semantics of infinitary formulas, and two versions of the semantics of programs. The main theorem asserts that if the aggregates used in a program recursively do not contain negation then the FLP-stable models of the program are the same as its FT-stable models. To prove the theorem we investigate under what conditions the models of a set of infinitary propositional formulas that are stable in the sense of Faber et al. are identical to the models stable in the sense of Ferraris and Truszczynski.

Syntax of Programs

The syntax of programs is described here in an abstract fashion, in the spirit of Gebser et al. (2015), so as to avoid inessential details related to the use of ASCII characters.

We assume that three pairwise disjoint sets of symbols are selected: *numerals*, *symbolic constants*, and *variables*. Further, we assume that these sets do not contain the symbols

$$+ \quad - \quad \times \quad / \tag{1}$$

$$\textit{inf} \quad \textit{sup} \tag{2}$$

$$= \quad \neq \quad < \quad > \quad \leq \quad \geq \tag{3}$$

$$\textit{not} \quad \wedge \quad \vee \quad \leftarrow \tag{4}$$

$$, \quad : \quad (\quad) \quad \{ \quad \} \tag{5}$$

and are different from the *aggregate names* *count*, *sum*, *max*, *min*. All these symbols together form the alphabet of programs, and rules will be defined as strings over this alphabet.

We assume that a 1–1 correspondence between the set of numerals and the set $\mathbf{Z}$ of integers is chosen. For every integer n, the corresponding numeral will be denoted by $\overline{n}$.

Terms are defined recursively, as follows:

- all numerals, symbolic constants, and variables, as well as symbols (2) are terms;
- if f is a symbolic constant and $\mathbf{t}$ is a non-empty tuple of terms (separated by commas) then $f(\mathbf{t})$ is a term;
- if t_1 and t_2 are terms and $\star$ is one of the symbols (1) then $(t_1 \star t_2)$ is a term.

A term, or a tuple of terms, is *ground* if it does not contain variables. A term, or a tuple of terms, is *precomputed* if it contains neither variables nor symbols (1). We assume a total order on precomputed terms such that *inf* is its least element, *sup* is its greatest element, and, for any integers m and n, $\overline{m} \leq \overline{n}$ iff $m \leq n$.

For each aggregate name we define a function that maps every set of non-empty tuples of precomputed terms to a precomputed term. Functions corresponding to each of the aggregate names are defined below using the following terminology. If the first member of a tuple $\mathbf{t}$ of precomputed terms is a numeral $\overline{n}$ then we say that the integer n is the *weight* of $\mathbf{t}$; if $\mathbf{t}$ is empty or its first member is not an numeral then the weight of $\mathbf{t}$ is 0. For any set T of tuples of precomputed terms,

- $\widehat{\textit{count}}(T)$ is the numeral corresponding to the cardinality of T if T is finite, and *sup* otherwise;
- $\widehat{\textit{sum}}(T)$ is the numeral corresponding to the sum of the weights of all tuples in T if T contains finitely many tuples with non-zero weights, and 0 otherwise;
- $\widehat{\textit{min}}(T)$ is *sup* if T is empty, the least element of the set consisting of the first elements of the tuples in T if T is a finite non-empty set, and *inf* if T is infinite;
- $\widehat{\textit{max}}(T)$ is *inf* if T is empty, the greatest element of the set consisting of the first elements of the tuples in T if T is a finite non-empty set, and *sup* if T is infinite.

An *atom* is a string of the form $p(\mathbf{t})$ where p is a symbolic constant and $\mathbf{t}$ is a tuple of terms. For any atom A, the strings

$$A \qquad \textit{not } A \tag{6}$$

are *symbolic literals*. An *arithmetic literal* is a string of the form $t_1 \prec t_2$ where t_1, t_2 are terms and $\prec$ is one of the symbols (3). A *literal* is a symbolic or arithmetic literal.[6]

An *aggregate atom* is a string of the form

$$\alpha\{\mathbf{t} : \mathbf{L}\} \prec s, \tag{7}$$

where

- α is an aggregate name,
- $\mathbf{t}$ is a tuple of terms,
- $\mathbf{L}$ is a tuple of literals called the "conditions" (if $\mathbf{L}$ is empty then the preceding colon may be dropped),
- $\prec$ is one of the symbols (3),
- and s is a term.

For any aggregate atom A, the strings (6) are *aggregate literals*; the former is called *positive*, and the latter is called *negative*.

A *rule* is a string of the form

$$H_1 \vee \dots \vee H_k \leftarrow B_1 \wedge \dots \wedge B_n \tag{8}$$

[5] The original translation τ could be used for this purpose as well. However, the definition of τ_1 seems more natural.

[6] In the parlance of the ASP-Core document, atoms are "classical atoms," arithmetic literals are "built-in atoms," and literals are "naf-literals."

$(k, n \geq 0)$, where each H_i is an atom, and each B_j is a literal or aggregate literal. The expression $B_1 \wedge \cdots \wedge B_n$ is the *body* of the rule, and $H_1 \vee \cdots \vee H_k$ is the *head*. A *program* is a finite set of rules.

About a variable we say that it is *global*

- in a symbolic or arithmetic literal L, if it occurs in L;
- in an aggregate atom (7) or its negation, if it occurs in s;
- in a rule (8), if it is global in at least one of the expressions H_i, B_j.

A variable that is not global is called *local*. A literal or a rule is *closed* if it has no global variables.

Stable Models of Infinitary Formulas

Formulas

Let σ be a propositional signature, that is, a set of propositional atoms. The sets $\mathcal{F}_0, \mathcal{F}_1, \ldots$ are defined as follows:

- $\mathcal{F}_0 = \sigma$,
- $\mathcal{F}_{i+1}$ is obtained from $\mathcal{F}_i$ by adding expressions $\mathcal{H}^\wedge$ and $\mathcal{H}^\vee$ for all subsets $\mathcal{H}$ of $\mathcal{F}_i$, and expressions $F \to G$ for all $F, G \in \mathcal{F}_i$.

The elements of $\bigcup_{i=0}^{\infty} \mathcal{F}_i$ are called *(infinitary propositional) formulas* over σ.

In an infinitary formula, $F \wedge G$ and $F \vee G$ are abbreviations for $\{F, G\}^\wedge$ and $\{F, G\}^\vee$ respectively; $\top$ and $\bot$ are abbreviations for $\emptyset^\wedge$ and $\emptyset^\vee$; $\neg F$ stands for $F \to \bot$, and $F \leftrightarrow G$ stands for $(F \to G) \wedge (G \to F)$. *Literals* over σ are atoms from σ and their negations. If $\langle F_\iota \rangle_{\iota \in I}$ is a family of formulas from one of the sets $\mathcal{F}_i$ then the expression $\bigwedge_\iota F_\iota$ stands for the formula $\{F_\iota : \iota \in I\}^\wedge$, and $\bigvee_\iota F_\iota$ stands for $\{F_\iota : \iota \in I\}^\vee$.

Subsets of a propositional signature σ will be called its *interpretations*. The satisfaction relation between an interpretation and a formula is defined recursively as follows:

- For every atom p from σ, $I \models p$ if $p \in I$.
- $I \models \mathcal{H}^\wedge$ if for every formula F in $\mathcal{H}$, $I \models F$.
- $I \models \mathcal{H}^\vee$ if there is a formula F in $\mathcal{H}$ such that $I \models F$.
- $I \models F \to G$ if $I \not\models F$ or $I \models G$.

We say that an interpretation satisfies a set $\mathcal{H}$ of formulas, or is a *model* of $\mathcal{H}$, if it satisfies every formula in $\mathcal{H}$. We say that $\mathcal{H}$ *entails* a formula F if every model of $\mathcal{H}$ satisfies F. Two sets of formulas are *equivalent* if they have the same models.

FLP-Stable Models

Let $\mathcal{H}$ be a set of infinitary formulas of the form $G \to H$, where H is a disjunction of atoms from σ. The *FLP-reduct* $FLP(\mathcal{H}, I)$ of $\mathcal{H}$ w.r.t. an interpretation I of σ is the set of all formulas $G \to H$ from σ such that I satisfies G. We say that I is an *FLP-stable model* of $\mathcal{H}$ if it is minimal w.r.t. set inclusion among the models of $FLP(\mathcal{H}, I)$.

It is clear that I satisfies $FLP(\mathcal{H}, I)$ iff I satisfies $\mathcal{H}$. Consequently every FLP-stable model of $\mathcal{H}$ is a model of $\mathcal{H}$.

FT-Stable Models

The *FT-reduct* $FT(F, I)$ of an infinitary formula F w.r.t. an interpretation I is defined as follows:

- For any atom p from σ, $FT(p, I) = \bot$ if $I \not\models p$; otherwise $FT(p, I) = p$.
- $FT(\mathcal{H}^\wedge, I) = \{FT(G, I) \mid G \in \mathcal{H}\}^\wedge$.
- $FT(\mathcal{H}^\vee, I) = \{FT(G, I) \mid G \in \mathcal{H}\}^\vee$.
- $FT(G \to H, I) = \bot$ if $I \not\models G \to H$; otherwise $FT(G \to H, I) = FT(G, I) \to FT(H, I)$.

The FT-reduct $FT(\mathcal{H}, I)$ of a set $\mathcal{H}$ of formulas is defined as the set of the reducts $FT(F, I)$ of all formulas F from $\mathcal{H}$. An interpretation I is an *FT-stable model* of $\mathcal{H}$ if it is minimal w.r.t. set inclusion among the models of $FT(\mathcal{H}, I)$.

It is easy to show by induction that I satisfies $FT(F, I)$ iff I satisfies F. Consequently every FT-stable model of a set of formulas is a model of that set.

It is easy to check also that if I does not satisfy F then $FT(F, I)$ is equivalent to $\bot$.

Comparison

An FLP-stable model of a set of formulas is not necessarily FT-stable, and an FT-stable model is not necessarily FLP-stable. For example, consider (the singleton set containing) the formula

$$p \vee \neg p \to p. \tag{9}$$

It has no FT-stable models, but the interpretation $\{p\}$ is its FLP-stable model. On the other hand, the formula

$$\neg\neg p \to p \tag{10}$$

has two FT-stable models, $\emptyset$ and $\{p\}$, but latter is not FLP-stable.

It is clear that replacing the antecedent of an implication by an equivalent formula within any set of formulas does not affect its FLP-stable models. For instance, from the perspective of the FLP semantics, formula (9) has the same meaning as $\top \to p$, and (10) has the same meaning as $p \to p$. On the other hand, the FLP-stable models may change if we break an implication of the form $F \vee G \to H$ into $F \to H$ and $G \to H$. For instance, breaking (9) into $p \to p$ and $\neg p \to p$ gives a set without FLP-stable models.

With the FT semantics, it is the other way around: it does matter, generally, whether we write $\neg\neg p$ or p in the antecedent of an implication, but breaking $F \vee G \to H$ into two implications cannot affect the set of stable models.

Transformations of infinitary formulas that do not affect their FT-stable models were studied by Harrison et al. (2017). These authors extended, in particular, the logic of here-and-there introduced by Heyting (1930) to infinitary propositional formulas and showed that any two sets of infinitary formulas that have the same models in the infinitary logic of here-and-there have also the same FT-stable models.

Semantics of Programs

In this section, we define two very similar translations, τ_1 and τ. Each of them transforms any program into a set of infinitary formulas over the signature σ_0 consisting of all

atoms of the form $p(\mathbf{t})$, where p is a symbolic constant and $\mathbf{t}$ is a tuple of precomputed terms. The definition of τ follows Gebser et al. (2015), and examples of using τ can be found in that paper.

Given these translations, the two versions of the semantics of programs are defined as follows. The *FLP-stable models* of a program Π are the FLP-stable models of $\tau_1\Pi$. The *FT-stable models* of Π are the FT-stable models of $\tau\Pi$.

Semantics of Terms

The semantics of terms tells us, for every ground term t, whether it is *well-formed*, and if it is, which precomputed term is considered its *value*:[7]

- If t is a numeral, symbolic constant, or one of the symbols *inf* or *sup* then t is well-formed, and its value $val(t)$ is t itself.
- If t is $f(t_1, \dots, t_n)$ and the terms $t_1, \dots, t_n$ are well-formed, then t is well-formed also, and $val(t)$ is $f(val(t_1), \dots, val(t_n))$.
- If t is $(t_1 + t_2)$ and the values of t_1 and t_2 are numerals $\overline{n_1}$, $\overline{n_2}$ then t is well-formed, and $val(t)$ is $\overline{n_1 + n_2}$; similarly when t is $(t_1 - t_2)$ or $(t_1 \times t_2)$.
- If t is (t_1/t_2), the values of t_1 and t_2 are numerals $\overline{n_1}$, $\overline{n_2}$, and $n_2 \neq 0$ then t is well-formed, and $val(t)$ is $\overline{\lfloor n_1/n_2 \rfloor}$.

If $\mathbf{t}$ is a tuple $t_1, \dots, t_n$ of well-formed ground terms then we say that $\mathbf{t}$ is well-formed, and its value $val(\mathbf{t})$ is the tuple $val(t_1), \dots, val(t_n)$.

A closed arithmetic literal $t_1 \prec t_2$ is well-formed if t_1 and t_2 are well-formed. A closed symbolic literal $p(\mathbf{t})$ or *not* $p(\mathbf{t})$ is well-formed if $\mathbf{t}$ is well-formed. A closed aggregate literal E or *not* E, where E is (7), is well-formed if s is well-formed.

Semantics of Arithmetic and Symbolic Literals

A well-formed arithmetic literal $t_1 \prec t_2$ is *true* if $val(t_1) \prec val(t_2)$, and *false* otherwise.

The result of applying the transformation τ_1 to a well-formed symbolic literal is defined as follows:

$$\tau_1(p(\mathbf{t})) \text{ is } p(val(\mathbf{t})); \quad \tau_1(\mathit{not}\ \mathbf{p}(\mathbf{t})) \text{ is } \neg\mathbf{p}(val(\mathbf{t})).$$

About a tuple of well-formed literals we say that it is *nontrivial* if all arithmetic literals in it are true. If $\mathbf{L}$ is a nontrivial tuple of well-formed arithmetic and symbolic literals then $\tau_1\mathbf{L}$ stands for the conjunction of the formulas $\tau_1 L$ for all symbolic literals L in $\mathbf{L}$.

Semantics of Aggregate Literals

Let E be a well-formed aggregate atom (7), and let $\mathbf{x}$ be the list of variables occurring in $\mathbf{t} : \mathbf{L}$. By A we denote the set of all tuples $\mathbf{r}$ of precomputed terms of the same length as $\mathbf{x}$ such that

(i) $\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}$ is well-formed, and

(ii) $\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}$ is well-formed and nontrivial.[8]

For any subset Δ of A, by $val(\Delta)$ we denote the set of tuples $\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}$ for all $\mathbf{r} \in \Delta$. We say that Δ *justifies* E if the relation $\prec$ holds between $\widehat{\alpha}(val(\Delta))$ and $val(s)$. We define $\tau_1 E$ to be the disjunction of formulas

$$\bigwedge_{\mathbf{r}\in\Delta} \tau_1(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \wedge \bigwedge_{\mathbf{r}\in A\setminus\Delta} \neg\tau_1(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \tag{11}$$

over the subsets Δ of A that justify E.

Assume, for example, that E is

$$\mathit{count}\{X : p(X)\} = \overline{0}. \tag{12}$$

Then

- $\mathbf{t}$ is X, $\mathbf{L}$ is $p(X)$, $\mathbf{x}$ is X, and A is the set of all precomputed terms, $val(\Delta)$ is Δ;
- $\widehat{\alpha}(val(\Delta))$ is the cardinality of Δ if Δ is finite and *sup* otherwise;
- Δ justifies (12) iff $\Delta = \emptyset$;
- $\tau_1 E$ is the conjunction of the formulas $\neg p(r)$ over all precomputed terms r.

The result of applying τ_1 to a negative aggregate literal *not* E is $\neg\tau_1 E$.

The definition of $\tau_1\mathbf{L}$ given earlier can be extended now to nontrivial tuples that may include well-formed literals of all three kinds: for any such tuple $\mathbf{L}$, $\tau_1\mathbf{L}$ stands for the conjunction of the formulas $\tau_1 L$ for all symbolic literals and aggregate literals L in $\mathbf{L}$.

Applying τ_1 to Rules and Programs

The result of applying τ_1 to a rule (8) is defined as the set of all formulas of the form

$$\tau_1((B_1, \dots, B_n)^{\mathbf{x}}_{\mathbf{r}}) \to \tau_1(H_1)^{\mathbf{x}}_{\mathbf{r}} \vee \cdots \vee \tau_1(H_k)^{\mathbf{x}}_{\mathbf{r}} \tag{13}$$

where $\mathbf{x}$ is the list of all global variables of the rule, and $\mathbf{r}$ is any tuple of precomputed terms of the same length as $\mathbf{x}$ such that all literals $(H_i)^{\mathbf{x}}_{\mathbf{r}}$, $(B_j)^{\mathbf{x}}_{\mathbf{r}}$ are well-formed.

For any program Π, $\tau_1\Pi$ stands for the union of the sets $\tau_1 R$ for all rules R of Π.

Transformation τ

The definition of τ differs from the definition of τ_1 in only one place: in the treatment of aggregate atoms In the spirit of Ferraris (2005), we define τE to be the conjunction of the implications

$$\bigwedge_{\mathbf{r}\in\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \to \bigvee_{\mathbf{r}\in A\setminus\Delta} \tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}) \tag{14}$$

over the subsets Δ of A that do not justify E.

For example, if E is (12) then τE is

$$\bigwedge_{\Delta\subseteq A,\ \Delta\neq\emptyset} \left(\bigwedge_{r\in\Delta} p(r) \to \bigvee_{r\in A\setminus\Delta} p(r) \right).$$

[7] In the input language of CLINGO, a term may contain "intervals", such as 1..3, and in that more general setting a ground term may have several values.

[8] Here $\mathbf{t}^{\mathbf{x}}_{\mathbf{r}}$ stands for the result of substituting $\mathbf{r}$ for $\mathbf{x}$ in $\mathbf{t}$. The meaning of $\mathbf{L}^{\mathbf{x}}_{\mathbf{r}}$ is similar.

It is easy to show that τE is equivalent to $\tau_1 E$. Consider the disjunction D of formulas (11) over all subsets Δ of A that do not justify E. It is to see that every interpretation satisfies either $\tau_1 E$ or D. On the other hand, no interpretation satisfies both D and $\tau_1 E$, because in every disjunctive term of $\tau_1 E$ and every disjunctive term of D there is a pair of conflicting conjunctive terms. It follows that D is equivalent to $\neg\tau_1 E$. It is clear that D is also equivalent to $\neg\tau E$.

Since all occurrences of translations $\tau_1 E$ in implication (13) belong to its antecedent, it follows that τ could be used instead of τ_1 in the definition of an FLP-stable model of a program. For the definition of an FT-stable model of a program, however, the difference between τ_1 and τ is essential. Although the translation τ_1 will not be used in the statement or proof of the main theorem, we introduce it here because it is simpler than τ in the sense that it in application to aggregate literals it does not produce implications. We anticipate that for establishing other properties of FLP-stable models it may be a useful tool.

Main Theorem

To see that the FLP and FT semantics of programs are generally not equivalent, consider the one-rule program

$$p \leftarrow \mathit{count}\{\overline{1} : \mathit{not}\ p\} < \overline{1}. \tag{15}$$

The result of applying τ to this program is $\neg\neg p \rightarrow p$. The FT-stable models are $\emptyset$ and $\{p\}$; the first of them is an FLP-stable model, and the second is not.

Our main theorem gives a condition ensuring that the FLP-stable models and FT-stable models of a program are the same. To state it, we need to describe the precise meaning of "recursion through aggregates."

The *predicate symbol* of an atom $p(t_1, \dots, t_n)$ is the pair p/n. The *predicate dependency graph* of a program Π is the directed graph that

- has the predicate symbols of atoms occurring in Π as its vertices, and
- has and edge from p/n to q/m if there is a rule R in Π such that p/n is the predicate symbol of an atom occurring in the head of R, and q/m is the predicate symbol of an atom occurring in the body of R.[9]

We say that an occurrence of an aggregate literal L in a rule R is *recursive* with respect to a program Π containing R if for some predicate symbol p/n occurring in L and some predicate symbol q/m occurring in the head of R there exists a path from p/n to q/m in the predicate dependency graph of Π.

For example, the predicate dependency graph of program (15) has a single vertex $p/0$ and an edge from $p/0$ to itself. The aggregate literal in the body of this program is recursive. Consider, on the other hand, the one-rule program

$$q \leftarrow \mathit{not}\ \mathit{count}\{\overline{1} : p\} < \overline{1}.$$

Its predicate dependency graph has the vertices $p/0$ and $q/0$, and an edge from $q/0$ to $p/0$. Since there is no path from $p/0$ to $q/0$ in this graph, the aggregate literal in the body of this rule is not recursive.

We say that an aggregate literal is *positive* if it is an aggregate atom and all symbolic literals occurring in it are positive.

Main Theorem *If every aggregate literal that is recursive with respect to a program Π is positive then the FLP-stable models of Π are the same as the FT-stable models of Π.*

In particular, if all aggregate literals in Π are positive then Π has the same FLP- and FT-stable models. For example, consider the one-rule program

$$p \leftarrow \mathit{count}\{\overline{1} : p\} > \overline{0}.$$

The only aggregate literal in this program is positive; according to the main theorem, the program has the same FLP- and FT-stable models. Indeed, it is easy to verify that $\emptyset$ is the only FLP-stable model of this program and also its only FT-stable model.

Main Lemma

In this section we talk about infinitary formulas over an arbitrary propositional signature σ.

Formulas p, $\neg p$, $\neg\neg p$, where p is an atom from σ, will be called *extended literals*. A *simple disjunction* is a disjunction of extended literals. A *simple implication* is an implication $\mathcal{A}^\wedge \rightarrow \mathcal{L}^\vee$ such that its antecedent $\mathcal{A}^\wedge$ is a conjunction of atoms and its consequent $\mathcal{L}^\vee$ is a simple disjunction. A conjunction of simple implications will be called a *simple formula*. Formulas of the form $G \rightarrow H$, where G is a simple formula and H is a disjunction of atoms, will be called *simple rules*.[10]

For example, (9), (10) can be rewritten as simple rules

$$(\top \rightarrow p \vee \neg p) \rightarrow p, \tag{16}$$

$$(\top \rightarrow \neg\neg p) \rightarrow p. \tag{17}$$

In the proof of Main Theorem we will show how any formula obtained by applying transformation τ to a program can be transformed into a simple rule with the same meaning.

A *simple program* is a set of simple rules.

In the statement of Main Lemma below, we refer to simple programs that are "FT-tight" and "FLP-tight." The lemma asserts that if a program is FT-tight then its FLP-stable models are FT-stable; if a program is FLP-tight then its FT-stable models are FLP-stable. To describe these two classes of simple programs we need the following preliminary definitions.

An atom p *occurs strictly positively* in a simple formula F if there is a conjunctive term $\mathcal{A}^\wedge \rightarrow \mathcal{L}^\vee$ in F such that p belongs to $\mathcal{L}$. An atom p *occurs positively* in a simple formula F if there is a conjunctive term $\mathcal{A}^\wedge \rightarrow \mathcal{L}^\vee$ in F such that p or $\neg\neg p$ belongs to $\mathcal{L}$.

We define the *(extended positive) dependency graph* of a simple program $\mathcal{H}$ to be the graph that has

[9] The definition of the predicate dependency graph in the ASP-Core document includes also edges between predicate symbols of atoms occurring in the head of the same rule. Dropping these edges from the graph makes the assertion of the main theorem stronger.

[10] Note that a simple rule is not a rule in the sense of the programming language described above; it is an infinitary propositional formula of a special syntactic form.

- all atoms occurring in $\mathcal{H}$ as its vertices, and
- an edge from p to q if for some formula $G \to H$ in $\mathcal{H}$, p is a disjunctive term in H and q occurs positively in G.

For example, the simple programs (16), (17) have the same dependency graph: a self-loop at p.[11]

A simple implication $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ will be called *positive* if $\mathcal{L}$ is a set of atoms, and *non-positive* otherwise. An edge from p to q in the dependency graph of a simple program $\mathcal{H}$ will be called *FT-critical* if for some formula $G \to H$ in $\mathcal{H}$, p is a disjunctive term in H and q occurs strictly positively in some non-positive conjunctive term D of G. We call a simple program *FT-tight* if its dependency graph has no path containing infinitely many FT-critical edges.[12]

Consider, for example, the dependency graph of program (16). Its only edge—the self-loop at p—is FT-critical, because the implication $\top \to p \vee \neg p$ is non-positive, and p occurs strictly positively in it. It follows that the program is not FT-tight: consider the path consisting of infinitely many repetitions of this self-loop. On the other hand, in the dependency graph of program (17) the same edge is not FT-critical, because p does not occur strictly positively in the implication $\top \to \neg\neg p$. Program (17) is FT-tight.

An edge from p to q in the dependency graph of a simple program $\mathcal{H}$ will be called *FLP-critical* if for some simple rule $G \to H$ in $\mathcal{H}$, p is a disjunctive term in H and, for some conjunctive term $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ of G, $\neg\neg q$ belongs to $\mathcal{L}$. We call a simple program *FLP-tight* if its dependency graph has no path containing infinitely many FLP-critical edges.

It is clear that if there are no extended literals of the form $\neg\neg p$ in a simple program then there are no FLP-critical edges in its dependency graph, so that the program is FLP-tight. For example, (16) is a simple program of this kind. On the other hand, in the dependency graph of program (17) the self-loop at p is FLP-critical, so that the program is not FLP-tight.

Main Lemma *For any simple program $\mathcal{H}$,*

(a) if $\mathcal{H}$ is FT-tight then all FLP-stable models of $\mathcal{H}$ are FT-stable;

(b) if $\mathcal{H}$ is FLP-tight then all FT-stable models of $\mathcal{H}$ are FLP-stable.

Some parts of the proof of the lemma below are inspired by results from Ferraris, Lee, and Lifschitz (2006).

Proof of Main Lemma

If F is a simple disjunction and X is a set of atoms, by $F^X_\perp$ we denote the simple disjunction obtained from F by removing all disjunctive terms that belong to X.[13] If F is a simple implication $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ then by $F^X_\perp$ we denote F itself if $\mathcal{A} \cap X$ is non-empty, and $\mathcal{A}^\wedge \to (\mathcal{L}^\vee)^X_\perp$ otherwise.[14] If F is a simple formula then $F^X_\perp$ stands for the simple formula obtained by applying this transformation to all conjunctive terms of F. It is clear that $F^X_\perp$ entails F.

For any simple program $\mathcal{H}$, by $\mathcal{H}^X_\perp$ we denote the simple program obtained from $\mathcal{H}$ by applying this transformation to G and H for each simple rule $G \to H$ in $\mathcal{H}$.

Lemma 1 *Let I be a model of a simple program $\mathcal{H}$, X be a set of atoms, and K be a subset of X such that the dependency graph of $\mathcal{H}$ has no edges from atoms in K to atoms in $X \setminus K$. If I satisfies $\mathcal{H}^X_\perp$, then I satisfies $\mathcal{H}^K_\perp$.*

Proof. Assume on the contrary that I does not satisfy $\mathcal{H}^K_\perp$. Then there is a simple rule $G \to H$ in $\mathcal{H}$ such that I satisfies $G^K_\perp$ but does not satisfy $H^K_\perp$. Further, since I satisfies $G^K_\perp$ and $G^K_\perp$ entails G, I satisfies G as well. Then since I is a model of $\mathcal{H}$, I satisfies H. Since I satisfies H but does not satisfy $H^K_\perp$, there is some atom p in H that is also in K. Now, since I satisfies $G^K_\perp$ it must also satisfy $G^X_\perp$. Indeed, if this were not the case, there would be some atom q occurring positively in G and also occurring in $X \setminus K$. Then there would be an edge from $p \in K$ to $q \in X \setminus K$, contradicting the assumption of the lemma. On the other hand, I does not satisfy $H^X_\perp$, since I does not satisfy $H^K_\perp$ and K is a subset of X. We may conclude that I does not satisfy $G^X_\perp \to H^X_\perp$ and therefore does not satisfy $\mathcal{H}^X_\perp$.

Lemma 2 *Let I be a model of a simple program $\mathcal{H}$ and let K be a subset of I such that there are no FT-critical edges in the subgraph of the dependency graph of $\mathcal{H}$ induced by K. If $I \models \mathcal{H}^K_\perp$ then $I \setminus K$ satisfies the FLP-reduct of $\mathcal{H}$ with respect to I.*

Proof. Consider a simple rule $G \to H$ in $\mathcal{H}$ such that $I \models G$, so that $G \to H$ is in the FLP-reduct of $\mathcal{H}$. We will show that $I \setminus K$ satisfies $G \to H$. Since $I \models \mathcal{H}^K_\perp$, either $I \not\models G^K_\perp$ or $I \models H^K_\perp$.

Case 1: $I \models H^K_\perp$. Then H has a disjunctive term that belongs to I but not to K, so that $I \setminus K \models H$. We conclude that $I \setminus K \models G \to H$.

Case 2: $I \not\models G^K_\perp$. Consider a conjunctive term $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ in G such that $I \not\models (\mathcal{A}^\wedge \to \mathcal{L}^\vee)^K_\perp$. Since $I \models G$, $I \models \mathcal{A}^\wedge \to \mathcal{L}^\vee$. It follows that $\mathcal{A} \cap K$ is empty and that I satisfies both $\mathcal{A}^\wedge$ and $\mathcal{L}^\vee$ but does not satisfy $(\mathcal{L}^\vee)^K_\perp$.

Case 2.1: $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ is positive. Then $\mathcal{L}$ is a set of atoms. Since $I \not\models (\mathcal{L}^\vee)^K_\perp$, all atoms from I that are in $\mathcal{L}$ are also in K. So $I \setminus K \not\models \mathcal{L}^\vee$. Since $\mathcal{A} \cap K$ is empty and I satisfies $\mathcal{A}^\wedge$, $I \setminus K$ also satisfies $\mathcal{A}^\wedge$. We may conclude that $I \setminus K \not\models G$ so that $I \setminus K \models G \to H$.

Case 2.2: $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ is non-positive. Since I satisfies $\mathcal{L}^\vee$ but not $(\mathcal{L}^\vee)^K_\perp$, there is an atomic disjunctive term p in $\mathcal{L}$ that belongs to $I \cap K$. Then p occurs positively in G. It follows that no disjunctive term in H occurs in K. (If there were

[11] We call the graph *extended* positive to emphasize the fact that the definition reqires q to occur positively in G, but not strictly positively.

[12] In the case of a finite dependency graph, this condition is equivalent to requiring that no cycle contains an FT-critical edge.

[13] This notation is motivated by the fact that $F^X_\perp$ is the result of substituting $\perp$ for the disjunctive members of F that belong to X, rewritten as a simple disjunction.

[14] This operation is a special case of the NES operation defined by Ferraris, Lee, and Lifschitz (2006). Distinguishing between the two cases in the definition is crucial for Lemmas 5 and 6.

such a disjunctive term q in H then, since $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ is non-positive, there would be an FT-critical edge from q to p in the subgraph of the dependency graph of $\mathcal{H}$ induced by K. But the condition of the lemma stipulates that there are no FT-critical edges in that graph.) Since I satisfies G and is a model of the program, I satisfies H as well. Since no atoms from K occur in H, it follows that $I \setminus K$ satisfies H, so that $I \setminus K$ satisfies $G \to H$.

Lemma 3 *If $\mathcal{H}$ is an FT-tight simple program and X is a non-empty set of atoms, then there exists a non-empty subset K of X such that in the subgraph of the dependency graph of $\mathcal{H}$ induced by X*

(i) there are no edges from K to atoms in $X \setminus K$, and

(ii) no atom in K has outgoing FT-critical edges.

Proof. Consider the subgraph of the dependency graph of $\mathcal{H}$ induced by X. It contains some vertex b such that there is no path starting at b that contains an FT-critical edge. (If there were no such vertex b, then there would be a path containing infinitely many FT-critical edges and $\mathcal{H}$ would not be FT-tight.) Take K to be the set of all vertices reachable from b. It is clear that condition (i) is satisfied. Furthermore, since all atoms in K are reachable from b, and no path starting at b contains an FT-critical edge, none of the atoms in K have outgoing FT-critical edges in the subgraph of the dependency graph of $\mathcal{H}$ induced by X. So condition (ii) is satisfied as well.

Lemma 4 *If $\mathcal{H}$ is an FT-tight simple program and I is an FLP-stable model of $\mathcal{H}$, then for every non-empty subset X of I, $I \not\models \mathcal{H}^X_\perp$.*

Proof. Assume on the contrary that there is some non-empty subset X of I such that $I \models \mathcal{H}^X_\perp$. By Lemma 3, there is a non-empty subset K of X meeting conditions (i) and (ii). Since $I \models \mathcal{H}^X_\perp$ and K satisfies (i), by Lemma 1 we may conclude that $I \models \mathcal{H}^K_\perp$. Since K satisfies condition (ii) and is a subset of X, it is clear that there are no FT-critical edges in the subgraph of the dependency graph of $\mathcal{H}$ induced by K. So by Lemma 2, $I \setminus K$ satisfies the FLP-reduct of $\mathcal{H}$, contradicting the assumption that I is FLP-stable.

Lemma 5 *Let G be a simple disjunction or a simple formula, and let X be a set of atoms. An interpretation I satisfies $G^X_\perp$ iff it satisfies $FT(G,I)^X_\perp$.*

Proof. To prove the assertion for a simple disjunction, it is sufficient to consider the case when G is a single extended literal. If G is an atom p,

$$I \models p^X_\perp \quad \text{iff} \quad p \in I \text{ and } p \notin X \quad \text{iff} \quad I \models FT(p,I)^X_\perp.$$

If G is either $\neg p$ or $\neg\neg p$, then $G^X_\perp$ is G and $FT(G,I)^X_\perp$ is $FT(G,I)$. It is clear that I satisfies G iff it satisfies $FT(G,I)$.

To prove the assertion of the lemma for simple formulas, it is sufficient to consider the case when G is a single simple implication $\mathcal{A}^\wedge \to \mathcal{L}^\vee$. If I does not satisfy G then it does not satisfy $G^X_\perp$ either; on the other hand, in this case $FT(G,I)$ is $\perp$, and so is $FT(G,I)^X_\perp$. Otherwise, $FT(G,I)^X_\perp$ is

$$(FT(\mathcal{A}^\wedge, I) \to FT(\mathcal{L}^\vee, I))^X_\perp. \tag{18}$$

We consider two cases corresponding to whether or not $\mathcal{A} \cap X \cap I$ is empty. If $\mathcal{A} \cap X \cap I$ is empty, I does not satisfy (18) iff

$$\mathcal{A} \subseteq I \quad \text{and} \quad I \not\models FT(\mathcal{L}^\vee, I)^X_\perp,$$

or equivalently,

$$I \models \mathcal{A}^\wedge \quad \text{and} \quad I \not\models (\mathcal{L}^\vee)^X_\perp.$$

If on the other hand, $\mathcal{A} \cap X \cap I$ is non-empty, then (18) is $FT(G,I)$ and $G^X_\perp$ is G.

Lemma 6 *For any simple disjunction G and any interpretations I and J, $J \models FT(G,I)$ iff $I \models FT(G,I)^{I\setminus J}_\perp$.*

Proof. It is sufficient to prove the lemma for the case when G is a single extended literal. If G is an atom p then

$$J \models FT(p,I) \quad \text{iff} \quad p \in I \text{ and } p \in J \quad \text{iff} \quad I \models FT(p,I)^{I\setminus J}_\perp.$$

If G is $\neg p$ then both

$$J \models FT(G,I)$$

and

$$I \models FT(G,I)^{I\setminus J}_\perp$$

are equivalent to $p \notin I$. If G is $\neg\neg p$ then both

$$J \models FT(G,I)$$

and

$$I \models FT(G,I)^{I\setminus J}_\perp$$

are equivalent to $p \in I$.

Lemma 7 *For any simple formula G and any interpretations I and J, if $I \models FT(G,I)^{I\setminus J}_\perp$ then $J \models FT(G,I)$.*

Proof. It is sufficient to consider the case when G is a single simple implication $\mathcal{A}^\wedge \to \mathcal{L}^\vee$. If I does not satisfy G then both $FT(G,I)$ and $FT(G,I)^{I\setminus J}_\perp$ are $\perp$. Assume I satisfies G. Then $FT(G,I)$ is

$$FT(\mathcal{A}^\wedge, I) \to FT(\mathcal{L}^\vee, I).$$

We consider two cases corresponding to whether or not $\mathcal{A} \cap I \setminus J$ is empty. If $\mathcal{A} \cap I \setminus J$ is non-empty, then $FT(G,I)^{I\setminus J}_\perp$ is $FT(G,I)$. Furthermore, J does not satisfy $FT(\mathcal{A}^\wedge, I)$. Indeed, if it did, $\mathcal{A}$ would be a subset of both I and J, contradicting the assumption that $\mathcal{A} \cap I \setminus J$ is non-empty. It follows that $J \models FT(G,I)$. If, on the other hand, $\mathcal{A} \cap I \setminus J$ is empty, then $FT(G,I)^{I\setminus J}_\perp$ is

$$FT(\mathcal{A}^\wedge, I) \to (FT(\mathcal{L}^\vee, I))^{I\setminus J}_\perp.$$

Assume that J does not satisfy $FT(G,I)$. Then

$$J \models FT(\mathcal{A}^\wedge, I) \quad \text{and} \quad J \not\models FT(\mathcal{L}^\vee, I).$$

From the first condition we may conclude that $I \models FT(\mathcal{A}^\wedge, I)$. (Indeed, if $J \models FT(\mathcal{A}^\wedge, I)$ then $\mathcal{A}$ must be a subset both of I and of J.) From the last condition using Lemma 6 it follows that $I \not\models FT(\mathcal{L}^\vee, I)^{I\setminus J}_\perp$. We may conclude that $I \not\models FT(G,I)^{I\setminus J}_\perp$.

Proof of Part (a) of Main Lemma. Let I be an FLP-stable model of an FT-tight simple program $\mathcal{H}$. Then $I \models \mathcal{H}$, so that $I \models FT(\mathcal{H}, I)$. We need to show that no proper subset J of I satisfies $FT(\mathcal{H}, I)$. Take a proper subset J of I, and let X be $I \setminus J$. By Lemma 4, I does not satisfy $\mathcal{H}^X_\perp$. Then there is a simple rule $G \to H$ in $\mathcal{H}$ such that I satisfies $G^X_\perp$ and does not satisfy $H^X_\perp$. By Lemma 5, it follows that I satisfies $FT(G, I)^X_\perp$ and does not satisfy $FT(H, I)^X_\perp$. Since $X = I \setminus J$, it follows that J satisfies $FT(G, I)$ (Lemma 7) but does not satisfy $FT(H, I)$ (Lemma 6). So J does not satisfy $FT(\mathcal{H}, I)$. It follows that I is FT-stable.

We turn now to the proof of part (b) of Main Lemma. If F is a simple disjunction then by F^+ we denote the result of replacing each extended literal $\neg\neg p$ in F by p, and similarly for simple implications, formulas, rules, and programs.

Lemma 8 *Let I be a model of a simple program $\mathcal{H}$, and let K be a set of atoms such that there are no FLP-critical edges in the subgraph of the dependency graph of $\mathcal{H}$ induced by K. If $I \models (\mathcal{H}^+)^K_\perp$ then $I \setminus K$ satisfies the FT-reduct of $\mathcal{H}$ with respect to I.*

Proof. We need to show that $I \setminus K$ satisfies the FT-reduct of every simple rule $G \to H$ in $\mathcal{H}$. Since I is a model of $\mathcal{H}$, that reduct is $FT(G, I) \to FT(H, I)$. If $I \not\models G$ then the antecedent of this implication is equivalent to $\perp$, and the assertion that the implication is satisified by $I \setminus K$ is trivial.

Assume then that $I \models G$. Since I is a model of $\mathcal{H}$, it follows that $I \models H$. Since $I \models (\mathcal{H}^+)^K_\perp$, either $I \not\models (G^+)^K_\perp$ or $I \models H^K_\perp$.

Case 1: $I \models H^K_\perp$. Then H has a disjunctive term p that belongs to I but not to K. Then p is also a disjunctive term in $FT(H, I)$, so that $I \setminus K \models FT(H, I)$. We conclude that $I \setminus K$ satisfies $FT(G \to H, I)$.

Case 2: $I \not\models (G^+)^K_\perp$. Consider a conjunctive term

$$\mathcal{A}^\wedge \to \mathcal{L}^\vee$$

in G such that I does not satisfy $(\mathcal{A}^\wedge \to (\mathcal{L}^\vee)^+)^K_\perp$. Since $I \models G$, $I \models \mathcal{A}^\wedge \to \mathcal{L}^\vee$. It follows that $\mathcal{A} \cap K$ is empty and that I satisfies $\mathcal{A}^\wedge$, $\mathcal{L}^\vee$ and $(\mathcal{L}^\vee)^+$ but does not satisfy $((\mathcal{L}^\vee)^+)^K_\perp$.

Case 2.1: $\mathcal{L}^\vee$ does not contain any extended literal $\neg\neg p$ such that $p \in K$. Since I satisfies $(\mathcal{L}^\vee)^+$ but not $((\mathcal{L}^\vee)^+)^K_\perp$, each atomic disjunctive term p in $(\mathcal{L}^\vee)^+$ that is in I must also be in K. Furthermore, I cannot satisfy any literal $\neg p$ in $\mathcal{L}$. (If it did, then that literal would also be in $((\mathcal{L}^\vee)^+)^K_\perp$, and this disjunction would be satisfied by I.) Since $\mathcal{L}$ does not contain any extended literal $\neg\neg p$ such that p is in K, I does not satisfy any extended literal $\neg\neg p$ in $\mathcal{L}$. (For each extended literal $\neg\neg p$ in $\mathcal{L}$, p is a disjunctive term in $((\mathcal{L}^\vee)^+)^K_\perp$. If I satisfied some extended literal $\neg\neg p \in \mathcal{L}$, then I would satisfy p and therefore also satisfy $((\mathcal{L}^\vee)^+)^K_\perp$.) We conclude that every extended literal in $\mathcal{L}$ that is satisfied by I is an atom from K. It follows that $FT(\mathcal{L}^\vee, I)$ is equivalent to a disjunction of atoms from K. So $I \setminus K \not\models FT(\mathcal{L}^\vee, I)$. Since $I \models \mathcal{A}^\wedge$, $I \models FT(\mathcal{A}^\wedge, I)$. Since $\mathcal{A} \cap K$ is empty, $I \setminus K$ also satisfies $FT(\mathcal{A}^\wedge)$. We may conclude that $I \setminus K \not\models FT(G, I)$ so that $I \setminus K \models FT(G \to H, I)$.

Case 2.2: $\mathcal{L}^\vee$ contains an extended literal $\neg\neg p$ such that $p \in K$. Then no disjunctive term in H occurs in K. (If there were such a disjunctive term q in H then there would be an FLP-critical edge from q to p in the subgraph of the dependency graph of $\mathcal{H}$ induced by K. But the condition of the lemma stipulates that there are no FLP-critical edges in that graph.) Since I satisfies H, I satisfies $FT(H, I)$ as well. Since no atoms from K occur in H, it follows that $I \setminus K$ satisfies $FT(H, I)$, so that $I \setminus K$ satisfies $FT(G \to H, I)$.

Lemma 9 *If $\mathcal{H}$ is an FLP-tight simple program and X is a non-empty set of atoms, then there exists a non-empty subset K of X such that in the subgraph of the dependency graph of $\mathcal{H}$ induced by X*

(i) there are no edges from K to atoms in $X \setminus K$, and

(ii) no atom in K has outgoing FLP-critical edges.

The proof is similar to the proof of Lemma 3.

Lemma 10 *If $\mathcal{H}$ is an FLP-tight simple program and I is an FT-stable model of $\mathcal{H}$, then for every non-empty subset X of I, $I \not\models (\mathcal{H}^+)^X_\perp$.*

Proof. Assume on the contrary that $I \models (\mathcal{H}^+)^X_\perp$ for some non-empty subset X of I. Consider a non-empty subset K of X meeting conditions (i) and (ii) from Lemma 9. Since $I \models (\mathcal{H}^+)^X_\perp$ and K satisfies (i), by Lemma 1 we may conclude that $I \models (\mathcal{H}^+)^K_\perp$. Since K satisfies (ii) and is a subset of X, there are no FLP-critical edges in the subgraph of the dependency graph of $\mathcal{H}$ induced by K. So by Lemma 8, $I \setminus K$ satisfies the FT-reduct of $\mathcal{H}$, contradicting the assumption that I is FT-stable.

Lemma 11 *Let G be a simple disjunction or a simple formula. For any interpretations I and J such that $J \subseteq I$, if $I \models (G^+)^{I \setminus J}_\perp$ then $J \models G$.*

Proof. To prove the assertion of the lemma for simple disjunctions, it is sufficient to consider the case when G is a single extended literal. If G is p or $\neg\neg p$ then $(G^+)^{I \setminus J}_\perp$ is $p^{I \setminus J}_\perp$. Since I satisfies this formula, $p \in J$, so that $J \models G$. If G is $\neg p$ then $(G^+)^{I \setminus J}_\perp$ is $\neg p$. Since $I \models \neg p$ and $J \subseteq I$, $J \models \neg p$.

To prove the assertion of the lemma for simple formulas, it is sufficient to consider the case when G is a single simple implication $\mathcal{A}^\wedge \to \mathcal{L}^\vee$. If $\mathcal{A} \cap I \setminus J$ is non-empty then

$$J \not\models \mathcal{A}^\wedge,$$

so that $J \models G$. If, on the other hand, $\mathcal{A} \cap I \setminus J$ is empty then $(G^+)^{I \setminus J}_\perp$ is $\mathcal{A}^\wedge \to ((\mathcal{L}^\vee)^+)^{I \setminus J}_\perp$. Assume that J does not satisfy G. Then

$$J \models \mathcal{A}^\wedge \quad \text{and} \quad J \not\models \mathcal{L}^\vee.$$

From the first condition and the fact that $J \subseteq I$ we may conclude that $I \models \mathcal{A}^\wedge$. From the second condition it follows, by the part of the lemma proved above, that $I \not\models ((\mathcal{L}^\vee)^+)^{I \setminus J}_\perp$. Consequently $I \not\models (G^+)^{I \setminus J}_\perp$.

Proof of Part (b) of Main Lemma. Let I be an FT-stable model of an FLP-tight simple program $\mathcal{H}$. Then I is a model

of $\mathcal{H}$, and consequently a model of the reduct $FLP(\mathcal{H}, I)$. We need to show that no proper subset J of I is a model of this reduct. Consider a proper subset J of I, and let X be $I \setminus J$. By Lemma 10, I does not satisfy $(\mathcal{H}^+)^X_\perp$. Then there is a simple rule $G \to H$ in $\mathcal{H}$ such that I satisfies $(G^+)^X_\perp$ and does not satisfy $H^X_\perp$. Since $(G^+)^X_\perp$ entails G^+, and G^+ is equivalent to G, we can conclude that I satisfies G, so that $G \to H$ belongs to the reduct $FLP(\mathcal{H}, I)$. On the other hand, by Lemma 11, J satisfies G. Since I does not satisfy $H^X_\perp$ and H is a disjunction of atoms, J does not satisfy H. So J does not satisfy $G \to H$, and consequently is not a model $FLP(\mathcal{H}, I)$.

Proof of Main Theorem

Consider a program Π in the programming language described at the beginning of this paper. Every formula in $\tau\Pi$ corresponds to one of the rules (8) of Π and has the form

$$\tau((B_1, \dots, B_n)^{\mathbf{x}}_{\mathbf{r}}) \to \tau(H_1)^{\mathbf{x}}_{\mathbf{r}} \vee \cdots \vee \tau(H_k)^{\mathbf{x}}_{\mathbf{r}} \qquad (19)$$

where $\mathbf{x}$ is the list of all global variables of the rule, and $\mathbf{r}$ is a tuple of precomputed terms such that all literals $(H_i)^{\mathbf{x}}_{\mathbf{r}}$, $(B_j)^{\mathbf{x}}_{\mathbf{r}}$ are well-formed. The consequent of (19) is a disjunction of atoms over the signature σ_0—the set of atoms of the form $p(\mathbf{t})$, where p is a symbolic constant and $\mathbf{t}$ is a tuple of precomputed terms. The antecedent of (19) is a conjunction of formulas of three types:

(i) literals over σ_0—each of them is $\tau(L^{\mathbf{x}}_{\mathbf{r}})$ for some symbolic literal L from the body of the rule;

(ii) implications of form (14)—each of them is $\tau(E^{\mathbf{x}}_{\mathbf{r}})$ for some aggregate atom E from the body of the rule;

(iii) negations of such implications—each of them is $\neg\tau(E^{\mathbf{x}}_{\mathbf{r}})$ for some aggregate literal *not* E from the body of the rule.

Each of the formulas $\tau(\mathbf{L}^{\mathbf{x}}_{\mathbf{r}})$ in (14) is a conjunction of literals over σ_0. It follows that (14) can be represented in the form

$$(\mathcal{A}_1)^\wedge \wedge \bigwedge_{p \in \mathcal{A}_2} \neg p \to \mathcal{C}^\vee, \qquad (20)$$

where $\mathcal{A}_1$ and and $\mathcal{A}_2$ are sets of atoms from σ_0, and $\mathcal{C}$ is a set of conjunctions of literals over σ_0.

Consider the simple program $\mathcal{H}$ obtained from $\tau\Pi$ by transforming the conjunctive terms of the antecedents of its formulas (19) as follows:

- Every literal L is replaced by the simple implication

$$\top \to L. \qquad (21)$$

- Every implication (20) is replaced by the simple formula

$$\bigwedge_{\phi} \left((\mathcal{A}_1)^\wedge \to \bigvee_{p \in \mathcal{A}_2} \neg\neg p \vee \bigvee_{C \in \mathcal{C},\ C \text{ is non-empty}} \phi(C) \right), \qquad (22)$$

where the big conjunction extends over all functions ϕ that map every non-empty conjunction from $\mathcal{C}$ to one of its conjunctive terms.

- Every negated implication (20) is replaced by the simple formula

$$\bigwedge_{p \in \mathcal{A}_1} (\top \to p) \wedge \bigwedge_{p \in \mathcal{A}_2} (\top \to \neg p) \wedge \bigwedge_{C \in \mathcal{C}} \bigvee_{L \text{ is a conjunctive term of } C} (\top \to \neg L). \qquad (23)$$

Each conjunctive term of the antecedent of (19) is equivalent to the simple formula that replaces it in $\mathcal{H}$. It follows that $\tau\Pi$ and $\mathcal{H}$ have the same FLP-stable models. On the other hand, $\tau\Pi$ and $\mathcal{H}$ have the same models in the infinitary logic of here-and-there, and consequently the same FT-stable models. Consequently, the FLP-stable models of Π can be characterized as the FLP-stable models of $\mathcal{H}$, and the FT-stable models of Π can be characterized as the FT-stable models of $\mathcal{H}$.

To derive the main theorem from the main lemma, we will establish two claims that relate the predicate dependency graph of Π to the dependency graph of $\mathcal{H}$:

Claim 1. If there is an edge from an atom $p(t_1, \dots, t_k)$ to an atom $q(s_1, \dots, s_l)$ in the dependency graph of $\mathcal{H}$ then there is an edge from p/k to q/l in the predicate dependency graph of Π.

Claim 2. If the edge from $p(t_1, \dots, t_k)$ to $q(s_1, \dots, s_l)$ in the dependency graph $\mathcal{H}$ is FT-critical or FLP-critical then Π contains a rule (8) such that

- p/k is the predicate symbol of one of the atoms H_i, and
- q/l is the predicate symbol of an atom occurring in one of the non-positive aggregate literals B_j.

Using these claims, we will show that if the dependency graph of $\mathcal{H}$ has a path with infinitely many FT-critical edges or infinitely many FLP-critical edges then we can find a non-positive aggregate literal recursive with respect to Π. The assertion of the theorem will immediately follow then by Main Lemma.

Assume that $p_1(\mathbf{t}^1), p_2(\mathbf{t}^2), \dots$ is a path in the dependency graph of $\mathcal{H}$ that contains infinitely many FT-critical edges (for FLP-critical edges, the reasoning is the same). By Claim 1, the sequence p_1/k_1, p_2/k_2, $\dots$, where k_i is the length of $\mathbf{t}^i$, is a path in the predicate dependency graph of Π. Since that graph is finite, there exists a positive integer a such that all vertices p_a/k_a, p_{a+1}/k_{a+1}, $\dots$ belong to the same strongly connected component. Since the path $p_1(\mathbf{t}^1), p_2(\mathbf{t}^2), \dots$ contains infinitely many FT-critical edges, there exists a $b \geq a$ such that the edge from $p_b(\mathbf{t}^b)$ to $p_{b+1}(\mathbf{t}^{b+1})$ is FT-critical. By Claim 2, it follows that Π contains a rule (8) such that p_b/k_b is the predicate symbol of one of the atoms H_i, and p_{b+1}/k_{b+1} is the predicate symbol of an atom occurring in one of the non-positive aggregate literals B_j. Since p_b/k_b and p_{b+1}/k_{b+1} belong to the same strongly connected component, there exists a path from p_{b+1}/k_{b+1} to p_b/k_b. It follows that B_i is recursive with respect to Π.

Proof of Claim 1. If there is an edge from $p(t_1, \dots, t_k)$ to $q(s_1, \dots, s_l)$ in the dependency graph of $\mathcal{H}$ then Π contains

a rule (8) such that $p(t_1, \ldots, t_k)$ occurs in the consequent of one of the implications (19) corresponding to this rule, and $q(s_1, \ldots, s_l)$ occurs in one of the formulas (21)–(23). Then $q(s_1, \ldots, s_l)$ occurs also in the antecedent of (19). It follows that p/k is the predicate symbol of one of the atoms occuring in the head of the rule, and q/l is the predicate symbol of one of the atoms occurring in its body.

Proof of Claim 2. If the edge from $p(t_1, \ldots, t_k)$ to $q(s_1, \ldots, s_l)$ in the dependency graph of $\mathcal{H}$ is FT-critical then Π contains a rule (8) such that $p(t_1, \ldots, t_k)$ occurs in the consequent of one of the implications (19) corresponding to this rule, and $q(s_1, \ldots, s_l)$ occurs strictly positively in one of the non-positive conjunctive terms $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ of one of the simple conjunctions (21)–(23). If a formula of form (21) is non-positive then no atoms occur in it strictly positively. Consequently $\mathcal{A}^\wedge \to \mathcal{L}^\vee$ is a conjunctive term of one of the formulas (22) or (23), and it corresponds to an aggregate literal from the body of the rule. That aggregate literal is not positive, because for any positive literal E no conjunctive term of the corresponding simple conjunction (22) is non-positive. It follows that p/k is the predicate symbol of one of the atoms in the head of the rule, and q/l is the predicate symbol of an atom from a non-positive aggregate literal in the body.

For FLP-critical edges the reasoning is similar, using the fact that formulas of form (21) do not contain double negations, and neither do formulas of form (22) corresponding to positive aggregate literals.

Related Work

The equivalence between the FLP and FT approaches to defining stable models for programs without aggregates was established by Faber, Leone, and Pfeifer (2004), Theorem 3. The fact that this equivalence is not destroyed by the use of positive aggregates was proved by Ferraris (2005), Theorem 3. That result is further generalized by Bartholomew, Lee, and Meng (2011), Theorem 7.

The program

$$q(\overline{1}),$$
$$r \leftarrow \mathit{count}\{X : \mathit{not}\ p(X),\ q(X)\} = \overline{1}$$

has no recursive aggregates but is not covered by any of the results quoted above because it contains a negative literal in the conditions of an aggregate atom.

Conclusion

An oversight in the semantics proposed in the ASP-Core document can be corrected using a translation into the language of infinitary propositional formulas. The main theorem of this paper describes conditions when stable models in the sense of the (corrected) ASP-Core definition are identical to stable models in the sense of the input language of CLINGO.

The main lemma asserts that if a set of infinitary propositional formulas is FT-tight then its FLP-stable models are FT-stable, and if it is FLP-tight then its FT-stable models are FLP-stable.

Acknowledgements

Martin Gebser made a valuable contribution to our work by pointing out an oversight in an earlier version of the proof and suggesting a way to correct it. We are grateful to Wolfgang Faber, Jorge Fandiño, Michael Gelfond, and Yuanlin Zhang for useful discussions related to the topic of this paper, and to the anonymous referees for their comments. This research was partially supported by the National Science Foundation under Grant IIS-1422455.

References

Bartholomew, M.; Lee, J.; and Meng, Y. 2011. First-order extension of the FLP stable model semantics via modified circumscription. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 724–730.

Faber, W.; Leone, N.; and Pfeifer, G. 2004. Recursive aggregates in disjunctive logic programs: Semantics and complexity. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*.

Ferraris, P.; Lee, J.; and Lifschitz, V. 2006. A generalization of the Lin-Zhao theorem. *Annals of Mathematics and Artificial Intelligence* 47:79–101.

Ferraris, P. 2005. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 119–131.

Gebser, M.; Harrison, A.; Kaminski, R.; Lifschitz, V.; and Schaub, T. 2015. Abstract Gringo. *Theory and Practice of Logic Programming* 15:449–463.

Gelfond, M., and Zhang, Y. 2014. Vicious circle principle and logic programs with aggregates. *Theory and Practice of Logic Programming* 14(4-5):587–601.

Harrison, A.; Lifschitz, V.; Pearce, D.; and Valverde, A. 2017. Infinitary equilibrium logic and strongly equivalent logic programs. *Arificial Intelligence* 246.

Harrison, A. 2017. *Formal Methods for Answer Set Programming*[15]. Ph.D. Dissertation, University of Texas at Austin.

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie von Wissenschaften. Physikalisch-mathematische Klasse* 42–56.

Truszczynski, M. 2010. Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs. *Artificial Intelligence* 174(16):1285 – 1306.

Truszczynski, M. 2012. Connecting first-order ASP and the logic FO(ID) through reducts. In Erdem, E.; Lee, J.; Lierler, Y.; and Pearce, D., eds., *Correct Reasoning: Essays on Logic-Based AI in Honor of Vladimir Lifschitz*. Springer. 543–559.

[15] http://www.cs.utexas.edu/users/ameliaj/pubs/ajh_thesis.pdf

Defeasible Entailment: from Rational Closure to Lexicographic Closure and Beyond

Giovanni Casini
CSC, Université du Luxembourg
Luxembourg
giovanni.casini@uni.lu

Thomas Meyer
CAIR & University of Cape Town
South Africa
tmeyer@cs.uct.ac.za

Ivan Varzinczak
CRIL, Univ. Artois & CNRS
France
varzinczak@cril.fr

Abstract

In this paper we present what we believe to be the first systematic approach for extending the framework for defeasible entailment first presented by Kraus, Lehmann, and Magidor—the so-called KLM approach. Drawing on the properties for KLM, we first propose a class of basic defeasible entailment relations. We characterise this basic framework in three ways: (*i*) semantically, (*ii*) in terms of a class of properties, and (*iii*) in terms of ranks on statements in a knowlege base. We also provide an algorithm for computing the basic framework. These results are proved through various representation results. We then refine this framework by defining the class of rational defeasible entailment relations. This refined framework is also characterised in thee ways: semantically, in terms of a class of properties, and in terms of ranks on statements. We also provide an algorithm for computing the refined framework. Again, these results are proved through various representation results.

We argue that the class of rational defeasible entailment relations—a strengthening of basic defeasible entailment which is itself a strengthening of the original KLM proposal—is worthy of the term *rational* in the sense that all of them can be viewed as appropriate forms of defeasible entailment. We show that the two well-known forms of defeasible entailment, *rational closure* and *lexicographic closure*, fall within our rational defeasible framework. We show that rational closure is the most conservative of the defeasible entailment relations within the framework (with respect to subset inclusion), but that there are forms of defeasible entailment within our framework that are more "adventurous" than lexicographic closure.

1 Introduction

The approach by Kraus, Lehmann and Magidor (1990) (a.k.a. KLM, or System P plus Rational Monotony) is a well-known framework for defeasible reasoning. The KLM properties can be viewed as constraints on appropriate forms of defeasible entailment. Currently two are the most known forms of defeasible entailment satisfying those properties: rational closure (Lehmann and Magidor 1992) and lexicographic closure (Lehmann 1995). Both forms of defeasible entailment can be characterised in three ways: semantically, in terms of ranks, and algorithmically.

Here we present what we believe to be the first systematic approach for extending the framework for defeasible entailment originally proposed by Kraus, Lehmann, and Magidor. Our first proposal for doing so is referred to as *basic defeasible entailment*. This framework can be obtained by a strengthening of the KLM properties: adding additional properties to those initially proposed by KLM. We then proceed to characterise basic defeasible entailment in two other ways. The first is a semantic characterisation in terms of a class of *ranked interpretations*. The second is a characterisation in terms of a class of functions that rank propositional (and defeasible) statements in a knowledge base according to their level of typicality. We then provide an algorithm for computing the framework. Given a rank function, we present an algorithm for computing basic defeasible entailment. The algorithm is a generalisation of one that has been proposed for computing rational closure (Freund 1998).

Having defined basic defeasible entailment, we propose a further strengthening via an additional property; one that requires any defeasible entailment relation to extend rational closure. In doing so we ensure that rational closure is viewed as the most conservative form of defeasible entailment. We refer to the resulting class of defeasible entailment relations as *rational* and argue that the class as a whole is worthy of further investigation. Part of this justification is that both rational and lexicographic closure are rational defeasible entailment relations. But while rational closure is the most conservative form of rational defeasible entailment, it turns out there are forms of rational defeasible entailment that are "bolder" than lexicographic closure. We show that rational defeasible entailment can also be characterised in two other ways: a semantic characterisation in terms of a further restricted class of ranked interpretations, and a characterisation in terms of a further restricted class of ranks. We also provide an algorithm for computing rational defeasible entailment. Given one of the restricted rank functions, the algorithm computes rational defeasible entailment.

We argue that the framework for rational defeasible entailment is reminiscent of the AGM framework for belief change (Alchourrón, Gärdenfors, and Makinson 1985) when viewed from the semantic perspective. Rational closure is analogous to full-meet revision (and contraction); there are defeasible entailment relations analogous to (linear) maxichoice revision (and contraction), and the semantic construction of the class of rational defeasible entailment relations bears a close resemblance to transitively relation partial-

meet revision (and contraction) (Gärdenfors 1988).

The remainder of the paper is structured as follows. Section 2 fixes the notation and terminology we use, and contains a summary of the necessary technical background: an introduction to KLM-style defeasible implication (Section 2.1), defeasible entailment for propositional languages enriched with a defeasible implication connective (Section 2.2), and an overview of rational closure (Section 2.3). Then, in Section 3 we present and discuss our notion of basic defeasible entailment. This is followed in Section 4 with an investigation into rational defeasible entailment. Section 5 is devoted to lexicographic closure and its relation to rational defeasible entailment, while Section 6 provides an overview of related work. Finally, in Section 7 we conclude and briefly point to future directions for this line of research.

2 Background

Let $\mathcal{P}$ be a finite set of propositional *atoms*. We use $p, q, \ldots$ as meta-variables for atoms. Propositional sentences are denoted by $\alpha, \beta, \ldots$, and are recursively defined in the usual way: $\alpha ::= \top \mid \bot \mid p \mid \neg\alpha \mid (\alpha \wedge \alpha) \mid (\alpha \vee \alpha) \mid (\alpha \rightarrow \alpha) \mid (\alpha \leftrightarrow \alpha)$. With $\mathcal{L}$ we denote the set of all propositional sentences.

With $\mathscr{U} \equiv_{\text{def}} \{0,1\}^{\mathcal{P}}$ we denote the (finite) set of all propositional *valuations*, with 1 representing truth and 0 representing falsity. We use $u, v \ldots$ to denote valuations. Whenever it eases the presentation, we represent valuations as sequences of atoms (e.g., p) and barred atoms (e.g., $\overline{p}$), with the understanding that the presence of a non-barred atom indicates that the atom is true (has the value 1) in the valuation, while the presence of a barred atom indicates that the atom is false (has the value 0) in the valuation. Thus, for the logic generated from $\mathcal{P} = \{p, q\}$, the valuation in which p is true and q is false will be represented as $p\overline{q}$. Satisfaction of a sentence $\alpha \in \mathcal{L}$ by $v \in \mathscr{U}$ is defined in the usual truth-functional way and is denoted by $v \Vdash \alpha$. The *models* of a set of sentences X is defined as: $[\![X]\!] \equiv_{\text{def}} \{v \in \mathscr{U} \mid v \Vdash \alpha \text{ for every } \alpha \in X\}$.

2.1 KLM-style defeasible implication

In the logic proposed by Kraus et al. (1990), often referred to as the *KLM approach*, we are interested in defeasible implications (or DIs) of the form $\alpha \mathrel{|\!\sim} \beta$, read as "typically, if α, then β". For instance, if $\mathcal{P} = \{\mathsf{b}, \mathsf{f}, \mathsf{p}\}$, where b, f and p stand for, respectively, "being a bird", "being able to fly", and "being a penguin", the following are examples of defeasible implications: $\mathsf{b} \mathrel{|\!\sim} \mathsf{f}$ (birds typically fly), $\mathsf{p} \wedge \mathsf{b} \mathrel{|\!\sim} \neg\mathsf{f}$ (penguins that are birds typically do not fly).

The semantics of KLM-style rational defeasible implications is given by structures referred to as *ranked interpretations* (Lehmann and Magidor 1992). In this work we adopt the following alternative representation thereof:

Definition 1 A ranked interpretation $\mathscr{R}$ is a function from $\mathscr{U}$ to $\mathbb{N} \cup \{\infty\}$ such that $\mathscr{R}(u) = 0$ for some $u \in \mathscr{U}$, and satisfying the following *convexity* property: for every $i \in \mathbb{N}$, if $\mathscr{R}(v) = i$, then, for every j s.t. $0 \leq j < i$, there is a $u \in \mathscr{U}$ for which $\mathscr{R}(u) = j$.

In a ranked interpretation, we call $\mathscr{R}(v)$ the *rank of* v w.r.t. $\mathscr{R}$. The intuition is that valuations with a lower rank are deemed more normal (or typical) than those with a higher rank, while those with an infinite rank are regarded as so atypical as to be impossible. With $\mathscr{U}^{\mathscr{R}} \equiv_{\text{def}} \{v \in \mathscr{U} \mid \mathscr{R}(v) < \infty\}$ we denote the *possible* valuations in $\mathscr{R}$. Given $\alpha \in \mathcal{L}$, we let $[\![\alpha]\!]^{\mathscr{R}} \equiv_{\text{def}} \{v \in \mathscr{U}^{\mathscr{R}} \mid v \Vdash \alpha\}$. $\mathscr{R}$ satisfies (is a ranked model of) α (denoted $\mathscr{R} \Vdash \alpha$) if $\mathscr{U}^{\mathscr{R}} \subseteq [\![\alpha]\!]^{\mathscr{R}}$.

Note that $\mathscr{R}$ generates a total preorder (a connected, transitive ordering) $\preceq_{\mathscr{R}}$ on $\mathscr{U}$ as follows: $v \preceq_{\mathscr{R}} u$ if and only if $\mathscr{R}(v) \leq \mathscr{R}(u)$. Moreover, given any total preorder $\preceq$ on $V \subseteq \mathscr{U}$, we can use its strict version $\prec$ to generate a ranked interpretation. To see how, first let the *height* $h(v)$ of $v \in V$ be the length of the $\prec$-path between any one of the $\prec$-minimal elements of V and v (where the length of the $\prec$-path between any of the $\prec$-minimal elements and a $\prec$-minimal element is 0).

Definition 2 For $V \subseteq \mathscr{U}$ and a total preorder $\preceq$ on V, the ranked interpretation $\mathscr{R}^{\preceq}$ generated from $\preceq$ is defined as follows: for every $v \in \mathscr{U}$, $\mathscr{R}^{\preceq}(v) = h(v)$ if $v \in V$, and $\mathscr{R}^{\preceq}(v) = \infty$ otherwise.

Given a ranked interpretation $\mathscr{R}$ and $\alpha, \beta \in \mathcal{L}$, we say $\mathscr{R}$ satisfies (is a ranked model of) the conditional $\alpha \mathrel{|\!\sim} \beta$ (denoted $\mathscr{R} \Vdash \alpha \mathrel{|\!\sim} \beta$) if all the $\prec$-minimal α-valuations also satisfy β, i.e., if $\min_{\prec}[\![\alpha]\!]^{\mathscr{R}} \subseteq [\![\beta]\!]^{\mathscr{R}}$. We say $\mathscr{R}$ satisfies a set of conditionals $\mathcal{K}$ if $\mathscr{R} \Vdash \alpha \mathrel{|\!\sim} \beta$ for every $\alpha \mathrel{|\!\sim} \beta \in \mathcal{K}$.

Figure 1 depicts an example of a ranked interpretation for $\mathcal{P} = \{\mathsf{b}, \mathsf{f}, \mathsf{p}\}$ satisfying $\mathcal{K} = \{\mathsf{p} \rightarrow \mathsf{b}, \mathsf{b} \mathrel{|\!\sim} \mathsf{f}, \mathsf{p} \mathrel{|\!\sim} \neg\mathsf{f}\}$.[1]

2	pbf
1	$\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}$ $\mathsf{pb}\overline{\mathsf{f}}$
0	$\overline{\mathsf{p}}\overline{\mathsf{b}\mathsf{f}}$ $\overline{\mathsf{p}}\overline{\mathsf{b}}\mathsf{f}$ $\overline{\mathsf{p}}\mathsf{bf}$

Figure 1: A ranked interpretation for $\mathcal{P} = \{\mathsf{b}, \mathsf{f}, \mathsf{p}\}$.

An important property of ranked interpretations is that *all* classical propositional sentences can be expressed as DIs. More precisely, we have the following result: for every $\mathscr{R}$ and every $\alpha \in \mathcal{L}$, $\mathscr{R} \Vdash \alpha$ if and only if $\mathscr{R} \Vdash \neg\alpha \mathrel{|\!\sim} \bot$. The logic of defeasible implications can therefore be viewed as an extension of propositional logic.

2.2 Defeasible Entailment

Let a *knowledge base* $\mathcal{K}$ be a finite set of defeasible implications. One of the central questions is to determine what *entailment* means in this context. That is, we aim to specify what it means for a defeasible implication to be entailed by a fixed knowledge $\mathcal{K}$. This is the main question with which we concern ourselves in this paper. We refer to this type of reasoning as *defeasible entailment* and denote it by $\mathrel{|\!\approx}$. It is

[1] For brevity, we shall omit the valuations with rank ∞ in our graphical representations of ranked interpretations.

important to note that, for the purposes of this paper, we assume $\mathcal{K}$ to be fixed.[2] Strictly speaking, we should therefore refer to defeasible $\mathcal{K}$-entailment. However, where there is no ambiguity, we frequently drop the $\mathcal{K}$ prefix, and just refer to defeasible entailment.

It is well-accepted that defeasible entailment (unlike classical entailment) is not unique. Lehmann and Magidor (1992) put forward *rational closure* as an appropriate form of defeasible entailment, while Lehmann (1995) proposed *lexicographic closure* as an alternative. We consider both of these in more detail below.

More generally, in studying different forms of defeasible entailment, the position advocated by Lehmann and Magidor (1992), and one we adopt here as well, is to consider a number of *rationality properties*, referred to as the KLM properties, for defeasible entailment.

(Ref) $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \alpha$

(LLE) $\dfrac{\alpha \equiv \beta,\ \mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \gamma}{\mathcal{K} \mathrel{|\!\approx} \beta \mathrel{|\!\sim} \gamma}$

(RW) $\dfrac{\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta,\ \beta \models \gamma}{\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \gamma}$

(And) $\dfrac{\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta,\ \mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \gamma}{\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta \wedge \gamma}$

(Or) $\dfrac{\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \gamma,\ \mathcal{K} \mathrel{|\!\approx} \beta \mathrel{|\!\sim} \gamma}{\mathcal{K} \mathrel{|\!\approx} \alpha \vee \beta \mathrel{|\!\sim} \gamma}$

(CM) $\dfrac{\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta,\ \mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \gamma}{\mathcal{K} \mathrel{|\!\approx} \alpha \wedge \beta \mathrel{|\!\sim} \gamma}$

(RM) $\dfrac{\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \gamma,\ \mathcal{K} \not\mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \neg\beta}{\mathcal{K} \mathrel{|\!\approx} \alpha \wedge \beta \mathrel{|\!\sim} \gamma}$

Lehmann and Magidor argue that defeasible entailment ought to satisfy all the above KLM properties. We refer to this as *LM-rationality*.

We can refer to defeasible entailment as being generated from single ranked interpretations.

Definition 3 A ranked interpretation $\mathscr{R}$ is said to generate a defeasible $\mathcal{K}$-entailment relation $\mathrel{|\!\approx}_{\mathscr{R}}$ by setting $\mathcal{K} \mathrel{|\!\approx}_{\mathscr{R}} \alpha \mathrel{|\!\sim} \beta$ if and only if $\mathscr{R} \Vdash \alpha \mathrel{|\!\sim} \beta$. (If there isn't any ambiguity, we drop the subscript $\mathscr{R}$.)

Lehmann and Magidor proved the following useful result.

Observation 1 (Lehman & Magidor (1992)) *A defeasible entailment relation is LM-rational if and only if it can be generated from a ranked interpretation.*[3]

Using LM-rationality as a starting point, it is easy to see that the most obvious attempt at defining defeasible entailment does not make the grade (Lehmann and Magidor 1992, Sect. 4.2).

[2]For an investigation of the case where $\mathcal{K}$ may vary, the reader is invited to consult the work of Casini and Meyer (2017).

[3]This result was originally phrased as a representation result about non-monotonic consequence relations on propositional statements (Kraus, Lehmann, and Magidor 1990), but were subsequently applied to defeasible entailment relations for propositional logic enriched with defeasible implication. Section 6 explains this in more detail.

Definition 4 A defeasible implication $\alpha \mathrel{|\!\sim} \beta$ is rank entailed by a knowledge base $\mathcal{K}$ (denoted as $\mathcal{K} \mathrel{|\!\approx}_R \alpha \mathrel{|\!\sim} \beta$) if every ranked model of $\mathcal{K}$ is also a ranked model of $\alpha \mathrel{|\!\sim} \beta$.

Rank entailment is an application of the classical Tarskian tradition of entailment to ranked interpretations, but it does not satisfy rational monotonicity (RM) and is therefore not rational (Lehmann and Magidor 1992, Th. 4.2). Despite this, rank entailment plays an important part in defining acceptable versions of defeasible entailment, since it can be viewed as the *monotonic core* of any appropriate form of defeasible entailment (Casini and Meyer 2017).

2.3 Rational Closure

The first version of defeasible entailment satisfying LM-rationality that we consider is *rational closure* (Lehmann and Magidor 1992). Given a knowledge base $\mathcal{K}$, consider the ordering $\preceq_{\mathcal{K}}$ on all ranked models of $\mathcal{K}$, which is defined as follows: $\mathscr{R}_1 \preceq_{\mathcal{K}} \mathscr{R}_2$ if for every $v \in \mathscr{U}$, $\mathscr{R}_1(v) \leq \mathscr{R}_2(v)$. Intuitively, ranked models lower down in the ordering are more typical. It is easy to see that $\preceq_{\mathcal{K}}$ is a weak partial order. Giordano et al. (2015) showed that there is a unique $\preceq_{\mathcal{K}}$-minimal element. The rational closure of $\mathcal{K}$ is defined in terms of this minimum ranked model (Giordano et al. 2015, Theorem 2).

Definition 5 Consider a knowledge base $\mathcal{K}$, and let $\mathscr{R}^{RC}_{\mathcal{K}}$ be the minimum element of the ordering $\preceq_{\mathcal{K}}$ on ranked models of $\mathcal{K}$. A defeasible implication $\alpha \mathrel{|\!\sim} \beta$ is in the rational closure of $\mathcal{K}$ (denoted as $\mathcal{K} \mathrel{|\!\approx}_{RC} \alpha \mathrel{|\!\sim} \beta$) if $\mathscr{R}^{RC}_{\mathcal{K}} \Vdash \alpha \mathrel{|\!\sim} \beta$.

Observe that there are two levels of typicality at work for rational closure, namely *within* ranked models of $\mathcal{K}$, where valuations lower down are viewed as more typical, and *between* ranked models of $\mathcal{K}$, where ranked models lower down in the ordering are viewed as more typical. Essentially, the most typical ranked model $\mathscr{R}^{RC}_{\mathcal{K}}$ is the one in which valuations are as typical as $\mathcal{K}$ allows them to be.

Since rational closure is defined in terms of a single ranked interpretation, it follows from Observation 1 that it is LM-rational (it satisfies all the KLM properties).

It will prove useful to be able to refer to the possible valuations w.r.t. a knowledge base.

Definition 6 We refer to $\mathscr{U}^{\mathcal{K}}_R \equiv_{\text{def}} \mathscr{U} \setminus \{u \in \mathscr{U} \mid u \in [\![\alpha]\!]$ for some α s.t. $\mathcal{K} \mathrel{|\!\approx}_R \neg\alpha \mathrel{|\!\sim} \bot\}$ as the set of possible valuations w.r.t. $\mathcal{K}$.

Informally $\mathscr{U}^{\mathcal{K}}_R$ refers to all the valuations not in conflict with rank entailment w.r.t. $\mathcal{K}$. From results by Lehmann and Magidor (1992) (Lemmas 24 and 30, to be precise) it follows that the possible valuations in the minimal model $\mathscr{R}^{RC}_{\mathcal{K}}$ are precisely the possible valuations w.r.t. $\mathcal{K}$: $\mathscr{U}^{\mathcal{K}}_R = \mathscr{U}^{\mathscr{R}^{RC}_{\mathcal{K}}}$.

Rational closure can also be defined in terms of the *base rank* of a statement w.r.t. $\mathcal{K}$.[4] Given a knowledge base $\mathcal{K}$, a propositional sentence α is said to be *exceptional* w.r.t. $\mathcal{K}$ if $\mathcal{K} \mathrel{|\!\approx}_R \top \mathrel{|\!\sim} \neg\alpha$ (i.e., α is false in all the most typical valuations in every ranked model of $\mathcal{K}$). Let $\varepsilon(\mathcal{K}) =$

[4]In the literature, the base rank of a sentence is just referred to as its rank, but for reasons that will become clear we have opted for the term *base rank* here.

$\{\alpha \mid\sim \beta \mid \mathcal{K} \not\approx_R \top \mid\sim \neg\alpha\}$. Now define a sequence of knowledge bases $\mathcal{E}_0^{\mathcal{K}}, \ldots, \mathcal{E}_\infty^{\mathcal{K}}$ as follows: $\mathcal{E}_0^{\mathcal{K}} \equiv_{\text{def}} \mathcal{K}$, $\mathcal{E}_i^{\mathcal{K}} \equiv_{\text{def}} \varepsilon(\mathcal{E}_{i-1}^{\mathcal{K}})$, for $0 < i < n$, and $\mathcal{E}_\infty \equiv_{\text{def}} \mathcal{E}_n^{\mathcal{K}}$, where n is the smallest k for which $\mathcal{E}_k^{\mathcal{K}} = \mathcal{E}_{k+1}^{\mathcal{K}}$ (since $\mathcal{K}$ is finite, n must exist). The *base rank* $br_{\mathcal{K}}(\alpha)$ of a propositional statement α w.r.t. a knowledge base $\mathcal{K}$ is defined to be the smallest r for which α is *not* exceptional w.r.t. $\mathcal{E}_r^{\mathcal{K}}$. $br_{\mathcal{K}}(\alpha) \equiv_{\text{def}} \min\{r \mid \mathcal{E}_r^{\mathcal{K}} \not\approx_R \top \mid\sim \neg\alpha\}$.

Observation 2 (Lehmann and Magidor (1992)) *$\mathcal{K} \not\approx_{RC} \alpha \mid\sim \beta$ if and only if $br_{\mathcal{K}}(\alpha) < br_{\mathcal{K}}(\alpha \wedge \neg\beta)$ or $br_{\mathcal{K}}(\alpha) = \infty$.*

It turns out that there is a fundamental connection between the base ranks of propositional statements w.r.t. a knowledge base $\mathcal{K}$ and the ranks of valuations in the minimal ranked model $\mathscr{R}_{\mathcal{K}}^{RC}$.

Observation 3 (Lehmann and Magidor (1992)) *For every knowledge base $\mathcal{K}$ and $\alpha \in \mathcal{L}$, $br_{\mathcal{K}}(\alpha) = \min\{i \mid$ there is a $v \in [\![\alpha]\!]$ s.t. $\mathscr{R}_{\mathcal{K}}^{RC}(v) = i\}$.*

From Observation 3 it also follows immediately that a classical statement α (or its defeasible representation $\neg\alpha \mid\sim \bot$) is in the rational closure of $\mathcal{K}$ if and only if the base rank of $\neg\alpha$ w.r.t. $\mathcal{K}$ is ∞, as intuitively expected.

The definition of base rank can be extended to defeasible implications as follows: $br_{\mathcal{K}}(\alpha \mid\sim \beta) \equiv_{\text{def}} br_{\mathcal{K}}(\alpha)$.

Assigning base ranks to defeasible implications in this way forms the basis of an algorithm for computing rational closure; an algorithm that can be reduced to a number of classical entailment checks. Define the *materialisation* of a knowledge base $\mathcal{K}$ as $\overrightarrow{\mathcal{K}} \equiv_{\text{def}} \{\alpha \rightarrow \beta \mid \alpha \mid\sim \beta \in \mathcal{K}\}$. It can be shown (Lehmann and Magidor 1992) that a sentence α is exceptional w.r.t. $\mathcal{K}$ if and only if $\overrightarrow{\mathcal{K}} \models \neg\alpha$. From this we can define a procedure `BaseRank` which partitions the materialisation of $\mathcal{K}$ into $n+1$ equivalence classes according to base rank: $i = 0, \ldots n-1, \infty, R_i \equiv_{\text{def}} \{\alpha \rightarrow \beta \mid \alpha \mid\sim \beta \in \mathcal{K}, br_{\mathcal{K}}(\alpha) = i\}$.

Algorithm 1: `BaseRank`

Input: A knowledge base $\mathcal{K}$
Output: An ordered tuple $(R_0, \ldots, R_{n-1}, R_\infty, n)$

$i := 0$;
$E_0 := \overrightarrow{\mathcal{K}}$;
repeat
  $E_{i+1} := \{\alpha \rightarrow \beta \in E_i \mid E_i \models \neg\alpha\}$;
  $R_i := E_i \setminus E_{i+1}$;
  $i := i + 1$;
until $E_{i-1} = E_i$;
$R_\infty := E_{i-1}$;
if $E_{i-1} = \emptyset$ **then**
  $n := i - 1$;
else
  $n := i$;
return $(R_0, \ldots, R_{n-1}, R_\infty, n)$

We can use the `BaseRank` procedure to define an algorithm for computing rational closure. It takes as input a knowledge base $\mathcal{K}$ and a DI $\alpha \mid\sim \beta$, and returns **true** if and only if $\alpha \mid\sim \beta$ is in the rational closure of $\mathcal{K}$.

Algorithm 2: `RationalClosure`

Input: A knowledge base $\mathcal{K}$ and a DI $\alpha \mid\sim \beta$
Output: true, if $\mathcal{K} \not\approx \alpha \mid\sim \beta$, and **false**, otherwise

$(R_0, \ldots, R_{n-1}, R_\infty, n) :=$ `BaseRank`$(\mathcal{K})$;
$i := 0$;
$R := \bigcup_{j=0}^{j<n} R_j$;
while $R_\infty \cup R \models \neg\alpha$ **and** $R \neq \emptyset$ **do**
  $R := R \setminus R_i$;
  $i := i + 1$;
return $R_\infty \cup R \models \alpha \rightarrow \beta$;

Informally, the algorithm keeps on removing (materialisations of) defeasible implications from (the materialisation of) $\mathcal{K}$, starting with the lowest base rank, and proceeding base rank by base rank, until it finds the first R which is classically consistent with α (and therefore α is not exceptional w.r.t. the defeasible version of R). $\alpha \mid\sim \beta$ is then taken to be in the rational closure of $\mathcal{K}$ if and only if R classically entails the materialisation of $\alpha \mid\sim \beta$.

Observation 4 (Freund (1998)) *Given a knowledge base $\mathcal{K}$ and a defeasible implication $\alpha \mid\sim \beta$, the algorithm* `RationalClosure` *returns* **true** *if and only if $\mathcal{K} \not\approx_{RC} \alpha \mid\sim \beta$.*

To conclude this section, we observe that algorithm `RationalClosure` involves a number of calls to a classical-entailment checker that is polynomial in the size of $\mathcal{K}$. Computing rational closure is therefore no harder than checking classical entailment.

3 Basic Defeasible Entailment

As discussed in the previous section, the departure point for defining defeasible entailment is that it ought to be LM-rational. The central question we address in this paper is whether LM-rationality is sufficient. That is, is it justifiable to regard any form of defeasible entailment that is LM-rational as appropriate? The immediate answer to this question is that it is not. For starters, we require $\not\approx$ to satisfy Inclusion and Classic Preservation.

(Inclusion) $\mathcal{K} \not\approx \alpha \mid\sim \beta$ for every $\alpha \mid\sim \beta \in \mathcal{K}$

(Classic Preservation) $\mathcal{K} \not\approx \alpha \mid\sim \bot$ if and only if $\mathcal{K} \not\approx_R \alpha \mid\sim \bot$

Inclusion simply requires all elements of $\mathcal{K}$ to be defeasibly entailed by $\mathcal{K}$. Classic Preservation states that the classical defeasible implications (those corresponding to classical sentences) defeasibly entailed by $\mathcal{K}$ should correspond exactly to those in the monotonic core of $\mathcal{K}$ (i.e., those that are rank entailed by $\mathcal{K}$). An easy corollary of Classic Preservation is Classic Consistency, requiring that a knowledge base is consistent if and only if it is consistent w.r.t. rank entailment.

(Classic Consistency) $\mathcal{K} \not\approx \top \mid\sim \bot$ if and only if $\mathcal{K} \not\approx_R \top \mid\sim \bot$

We refer to a defeasible entailment relation satisfying LM-rationality, Inclusion, and Classic Preservation as a *basic defeasible entailment relation*.

We shall see below (using Theorem 1) that rational closure is a basic defeasible entailment relation. However, since ranked entailment does not satisfy RM, it is not LM-rational, and is therefore not a basic defeasible entailment relation.

We now proceed with presenting our first fundamental result, a semantic characterisation of basic defeasible entailment relations in which we consider a class of ranked models we refer to as $\mathcal{K}$-faithful ranked.

Definition 7 A ranked model $\mathscr{R}$ of $\mathcal{K}$ is said to be $\mathcal{K}$-faithful if the possible valuations in $\mathscr{R}$ are precisely the possible valuations w.r.t. $\mathcal{K}$: $\mathscr{U}^{\mathscr{R}} = \mathscr{U}_R^{\mathcal{K}}$.

Note that the minimal model $\mathscr{R}_{\mathcal{K}}^{RC}$ is $\mathcal{K}$-faithful. This brings us to our first representation result.

Theorem 1 *Every basic defeasible $\mathcal{K}$-entailment relation can be generated from a $\mathcal{K}$-faithful ranked model. Conversely, every $\mathcal{K}$-faithful ranked model generates a defeasible $\mathcal{K}$-entailment relation.*

From this it follows immediately that basic defeasible entailment satisfies the following property.

(Rank Extension) If $\mathcal{K} \mathrel{|\!\approx}_R \alpha \mathrel{|\!\sim} \beta$, then $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$

To see why, note that if $\mathcal{K} \mathrel{|\!\approx}_R \alpha \mathrel{|\!\sim} \beta$ then $\alpha \mathrel{|\!\sim} \beta$ is satisfied by every ranked model of $\mathcal{K}$, and in particular, by the ranked model used to generate $\mathrel{|\!\approx}$.

Rank Extension requires $\mathrel{|\!\approx}$ to extend its monotonic core (i.e., it is required to extend the rank entailment of $\mathcal{K}$).

We can also characterise basic defeasible entailment by generalising the notion of base rank.

Definition 8 Let $r : \mathcal{L} \longrightarrow \mathbb{N} \cup \{\infty\}$ be a rank function such that $r(\top) = 0$ and satisfying the following *convexity* property: for every $i \in \mathbb{N}$, if $r(\alpha) = i$ then, for every j such that $0 \leq j < i$, there is a $\beta \in \mathcal{L}$ for which $r(\beta) = j$. r is referred to as entailment preserving if $\alpha \models \beta$ implies that $r(\alpha) \geq r(\beta)$. Given a knowledge base $\mathcal{K}$, r is said to be $\mathcal{K}$-faithful if (*i*) it is entailment preserving; (*ii*) $r(\alpha) < r(\alpha \wedge \neg\beta)$ or $r(\alpha) = \infty$, for every $\alpha \mathrel{|\!\sim} \beta \in \mathcal{K}$, and (*iii*) $r(\alpha) = \infty$ if and only if $\mathcal{K} \mathrel{|\!\approx}_R \alpha \mathrel{|\!\sim} \bot$.

Observe that the base rank $br_{\mathcal{K}}(\cdot)$ is $\mathcal{K}$-faithful.

Definition 9 A rank function r generates a defeasible entailment relation $\mathrel{|\!\approx}$ whenever $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$ if $r(\alpha) < r(\alpha \wedge \neg\beta)$ or $r(\alpha) = \infty$.

We can now present our second representation result.

Theorem 2 *Every basic defeasible $\mathcal{K}$-entailment relation can be generated by a $\mathcal{K}$-faithful rank function r. Conversely, every $\mathcal{K}$-faithful rank function r generates a basic defeasible $\mathcal{K}$-entailment relation.*

Next, we present algorithm `DefeasibleEntailment` that computes the defeasible entailment relation generated by a $\mathcal{K}$-faithful rank function. It is a modified version of the `RationalClosure` algorithm presented earlier, differing from that algorithm in that the call to the `BaseRank` algorithm is replaced with a call to the `Rank` algorithm described below. As for the `Rank` algorithm, it receives as input, not just a knowledge base $\mathcal{K}$ as `BaseRank` does, but also a $\mathcal{K}$-faithful rank function r, and then produces as output a sequence $(R_0, \ldots, R_{n-1}, R_\infty, n)$ where the R_is are *sentences*, unlike the `BaseRank` algorithm, which produces sets of sentences. `DefeasibleEntailment` is then adjusted accordingly.

Algorithm 3: `DefeasibleEntailment`

Input: A knowledge base $\mathcal{K}$, a $\mathcal{K}$-faithful rank function r, and a DI $\alpha \mathrel{|\!\sim} \beta$
Output: **true**, if $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$, and **false**, otherwise

$(R_0, \ldots, R_{n-1}, R_\infty, n)$:= `Rank`($\mathcal{K}$,r);
$i := 0$;
$R := \bigcup_{j=0}^{j<n} \{R_j\}$;
while $\{R_\infty\} \cup R \models \neg\alpha$ **and** $R \neq \emptyset$ **do**
  $R := R \setminus \{R_i\}$;
  $i := i + 1$;
return $\{R_\infty\} \cup R \models \alpha \rightarrow \beta$;

Like the `RationalClosure` algorithm, the `DefeasibleEntailment` algorithm keeps on removing statements, starting with the lowest rank, and proceeding rank by rank, until it finds the first R which is classically consistent with α. $\alpha \mathrel{|\!\sim} \beta$ is then taken to be defeasibly entailed by $\mathcal{K}$ if and only if R classically entails the materialisation of $\alpha \mathrel{|\!\sim} \beta$. Intuitively, the R_is correspond to classical representations of defeasible information, with different R_is representing information with different levels of typicality, and with R_∞ corresponding to information that is classical, rather than defeasible. In fact, the set containing all the R_is is equivalent to the materialisation of $\mathcal{K}$.

For $\alpha \in \mathcal{L}$, let $[\alpha]$ be a canonical representative of the set $\{\beta \mid \beta \equiv \alpha\}$. The `Rank` algorithm receives as input a knowledge base $\mathcal{K}$ and a $\mathcal{K}$-faithful rank function r and, as mentioned above, produces as output an ordered tuple of *sentences* $(R_0, \ldots, R_{n-1}, R_\infty, n)$.

Algorithm 4: `Rank`

Input: A knowledge base $\mathcal{K}$ and a $\mathcal{K}$-faithful rank function r
Output: An ordered tuple $(R_0, \ldots, R_{n-1}, R_\infty, n)$

$R_\infty := \neg\left(\bigvee_{r([\alpha])=\infty}[\alpha]\right)$;
$n := \max\{i \in \mathbb{N} \mid \text{there is an } \alpha \in \mathcal{L} \text{ s.t. } r(\alpha) = i\}$;
if $n = 0$ **then**
  $R_0 := \top$; $n := 1$;
else
  for $i := 0$ **to** $n - 1$ **do**
    $R_i \equiv_{\text{def}} \neg\left(\bigvee_{r([\alpha])=i+1}[\alpha]\right)$
return $(R_0, \ldots, R_{n-1}, R_\infty, n)$

Note that if there is no α such that $r(\alpha) = \infty$, then R_∞ will be set to $\top$. This corresponds to the case where all information is defeasible. Note also that if $n = 0$, it corresponds

to the case where there is no defeasible information. In this case we set n to 1 and set R_0 to $\top$. Also, as mentioned above, the set consisting of all the R_is is equivalent to $\mathcal{K}$.

Lemma 1 *Let $(R_0, \ldots, R_{n-1}, R_\infty, n)$ be the output obtained from the* `Rank` *algorithm, given a knowledge base $\mathcal{K}$ and a $\mathcal{K}$-faithful ranking function r. Then $\{R_\infty\} \cup \bigcup_{j=0}^{j<n} \{R_j\} \equiv \overrightarrow{\mathcal{K}}$.*

To get a sense of how the algorithm works, consider the following examples.

Example 1 Let $\mathcal{K} = \{\mathsf{p} \rightarrow \mathsf{b}, \mathsf{b} \mathrel{|\!\sim} \mathsf{f}, \mathsf{p} \mathrel{|\!\sim} \neg\mathsf{f}\}$. It can be shown that there is only one ranking function r for which $r((\mathsf{b} \rightarrow \mathsf{f}) \rightarrow \mathsf{p}) = 1$, $r(\mathsf{p} \wedge (\mathsf{b} \rightarrow \mathsf{f})) = 2$, and $r(\neg(\mathsf{p} \rightarrow \mathsf{b})) = \infty$. Moreover, for r it will be the case that for every $\alpha \in \mathcal{L}$, $r(\alpha) = \infty$ or $r(\alpha) \leq 2$. Given $\mathcal{K}$ and r, the `Rank` algorithm will output the ordered tuple $(R_0, R_1, R_\infty, 2)$, where $R_\infty \equiv \mathsf{p} \rightarrow \mathsf{b}$,

$$R_1 \equiv \neg(\mathsf{p} \wedge (\mathsf{b} \rightarrow \mathsf{f})) \equiv \mathsf{p} \rightarrow (\mathsf{b} \wedge \neg\mathsf{f}), \text{ and}$$

$$R_0 \equiv \neg((\mathsf{b} \rightarrow \mathsf{f}) \rightarrow \mathsf{p}) \equiv (\mathsf{b} \rightarrow \mathsf{f}) \wedge \neg\mathsf{p}.$$

Given $\mathcal{K}$, r, and $(\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f}) \mathrel{|\!\sim} \neg\mathsf{f}$, `DefeasibleEntailment` will return **true**. It will do so by first verifying that $\{R_0, R_1, R_\infty\} \not\models \neg((\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f}))$ and then checking whether $\{R_0, R_1, R_\infty\} \models ((\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f})) \rightarrow \neg\mathsf{f}$ (which it does). It is worth noting that, given this r, algorithm `DefeasibleEntailment` computes the rational closure of $\mathcal{K}$. ■

Example 2 Consider again $\mathcal{K} = \{\mathsf{p} \rightarrow \mathsf{b}, \mathsf{b} \mathrel{|\!\sim} \mathsf{f}, \mathsf{p} \mathrel{|\!\sim} \neg\mathsf{f}\}$. It can be shown that there is only one ranking function r s.t. $r(\mathsf{f} \rightarrow \mathsf{p}) = 1$, $r((\mathsf{b} \vee \mathsf{f}) \rightarrow (\mathsf{p} \wedge \mathsf{f})) = 2$, and $r(\neg(\mathsf{p} \rightarrow \mathsf{b})) = \infty$, and that r is $\mathcal{K}$-faithful. Moreover, for r it will be the case that for every $\alpha \in \mathcal{L}$, $r(\alpha = \infty)$ or $r(\alpha) \leq 2$. Given $\mathcal{K}$ and r, the `Rank` algorithm will output the ordered tuple $(R_0, R_1, R_\infty, 2)$ where $R_\infty \equiv \mathsf{p} \rightarrow \mathsf{b}$,

$$R_1 \equiv \neg((\mathsf{b} \vee \mathsf{f}) \rightarrow (\mathsf{p} \wedge \mathsf{f})) \equiv (\neg\mathsf{b} \rightarrow \mathsf{f}) \wedge (\mathsf{p} \rightarrow \neg\mathsf{f}), \text{ and}$$

$$R_0 \equiv \neg(\mathsf{f} \rightarrow \mathsf{p}) \equiv \mathsf{f} \wedge \neg\mathsf{p}.$$

Given $\mathcal{K}$, r, and the DI $(\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f}) \mathrel{|\!\sim} \neg\mathsf{f}$, algorithm `DefeasibleEntailment` will return **false**. It will do so by first removing R_0 (since $\{R_0, R_1, R_\infty\} \models \neg((\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f}))$), then removing R_1 (since $\{R_1, R_\infty\} \models \neg((\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f}))$), and then, since $\{R_\infty\} \not\models \neg((\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f}))$, it will check whether $\{R_\infty\} \models ((\mathsf{p} \leftrightarrow \mathsf{b}) \wedge (\mathsf{b} \leftrightarrow \mathsf{f})) \rightarrow \neg\mathsf{f}$ (which it does not). ■

Definition 10 Algorithm `DefeasibleEntailment` is said to compute a defeasible entailment relation $\mathrel{|\!\approx}$ for a knowledge base $\mathcal{K}$ and a rank functon r whenever $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$ if `DefeasibleEntailment`, when presented with $\mathcal{K}$, r, and $\alpha \mathrel{|\!\sim} \beta$, returns **true**.

This provides us with the material for our third representation result.

Theorem 3 *Given a $\mathcal{K}$-faithful rank function r, the basic defeasible entailment relation generated by r is exactly the defeasible entailment relation computed by the* `DefeasibleEntailment` *algorithm when given $\mathcal{K}$ and r as input.*

The results obtained for basic defeasible entailment can therefore be summarised in the following theorem.

Theorem 4 *The following statements are equivalent.*

- $\mathrel{|\!\approx}$ *is a basic defeasible $\mathcal{K}$-entailment relation.*
- *There is a $\mathcal{K}$-faithful ranked model $\mathscr{R}$ and a $\mathcal{K}$-faithful rank function r such that:*
  1. $r(\alpha) = \min\{i \mid$ *there is a* $v \in [\![\alpha]\!]$ *s.t.* $\mathscr{R}(v) = i\}$*;*
  2. $\mathrel{|\!\approx}$ *can be generated from $\mathscr{R}$;*
  3. $\mathrel{|\!\approx}$ *can be generated from r;*
  4. $\mathrel{|\!\approx}$ *can be computed by algorithm* `Defeasible Entailment`*, given $\mathcal{K}$ and r as input.*

Note that point 1 in Theorem 4 establishes a connection between $\mathscr{R}$ and r via a result that is a generalisation of Observation 3.

Finally, observe that `DefeasibleEntailment` involves a number of calls to a classic entailment checker that is linear in n times the size of $\mathcal{K}$ (where n is the number returned by the `Rank` algorithm). But note also that n may be exponential in the size of $\mathcal{K}$.

4 Rational Defeasible Entailment

Having analysed basic defeasible entailment in the previous section, we now proceed by contending that it is too permissive. In particular, we first show that it does not satisfy the following property.

(RC Extension) If $\mathcal{K} \mathrel{|\!\approx}_{RC} \alpha \mathrel{|\!\sim} \beta$, then $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$

RC Extension requires of $\mathrel{|\!\approx}$ to extend the rational closure of $\mathcal{K}$. To see that basic defeasible entailment does not satisfy RC Extension, consider the following example.

Example 3 Figure 2 depicts the ($\mathcal{K}$-faithful) minimal ranked model $\mathscr{R}_{\mathcal{K}}^{RC}$ of $\mathcal{K} = \{\mathsf{p} \rightarrow \mathsf{b}, \mathsf{b} \mathrel{|\!\sim} \mathsf{f}, \mathsf{p} \mathrel{|\!\sim} \neg\mathsf{f}\}$. Note that $\mathscr{R}_{\mathcal{K}}^{RC} \Vdash \neg\mathsf{p} \wedge \neg\mathsf{f} \mathrel{|\!\sim} \neg\mathsf{b}$. From Definition 5 it then follows that $\mathcal{K} \mathrel{|\!\approx}_{RC} \neg\mathsf{p} \wedge \neg\mathsf{f} \mathrel{|\!\sim} \neg\mathsf{b}$. But also note that for the $\mathcal{K}$-faithful ranked model $\mathscr{R}$ in Figure 3 below it follows that $\mathscr{R} \not\Vdash \neg\mathsf{p} \wedge \neg\mathsf{f} \mathrel{|\!\sim} \neg\mathsf{b}$. And from Theorem 4 it follows that for the basic defeasible $\mathcal{K}$-entailment relation $\mathrel{|\!\approx}$ generated from $\mathscr{R}$, $\mathcal{K} \not\mathrel{|\!\approx} \neg\mathsf{p} \wedge \neg\mathsf{f} \mathrel{|\!\sim} \neg\mathsf{b}$. So RC Extension does not hold. ■

2	pbf
1	$\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}$ $\mathsf{p}\mathsf{b}\overline{\mathsf{f}}$
0	$\overline{\mathsf{p}}\overline{\mathsf{b}}\overline{\mathsf{f}}$ $\overline{\mathsf{p}}\overline{\mathsf{b}}\mathsf{f}$ $\overline{\mathsf{p}}\mathsf{b}\mathsf{f}$

Figure 2: The minimal $\mathcal{K}$-faithful ranked model $\mathscr{R}_{\mathcal{K}}^{RC}$

If a basic defeasible entailment relation satisfies RC Extension as well, we refer to it as *rational* defeasible entailment. We propose the class of rational defeasible entailment relations as those worthy of the term *rational* and analyse them further in the remainder of this section.

We start by showing that rational defeasible entailment can be characterised semantically in terms of a subset of the $\mathcal{K}$-faithful ranked models.

2	pbf
1	$\overline{\mathsf{pbf}}$ $\mathsf{p}\overline{\mathsf{bf}}$ $\mathsf{pb}\overline{\mathsf{f}}$
0	$\overline{\mathsf{p}}\mathsf{bf}$ $\overline{\mathsf{pbf}}$

Figure 3: The $\mathcal{K}$-faithful ranked model $\mathscr{R}$

Definition 11 A $\mathcal{K}$-faithful ranked model $\mathscr{R}$ is said to be rank preserving if the following condition holds: for all $v, u \in \mathscr{U}$, if $\mathscr{R}_{\mathcal{K}}^{RC}(v) < \mathscr{R}_{\mathcal{K}}^{RC}(u)$, then $\mathscr{R}(v) < \mathscr{R}(u)$.

Informally, rank preservation requires the total preorder $\preceq_{\mathscr{R}}$ generated from $\mathscr{R}$ to respect the relative positions assigned to valuations in the minimal model $\mathscr{R}_{\mathcal{K}}^{RC}$ of $\mathcal{K}$.

Theorem 5 *Every rational defeasible $\mathcal{K}$-entailment relation can be generated by a rank preserving $\mathcal{K}$-faithful model. Conversely, every rank preserving $\mathcal{K}$-faithful model generates a rational defeasible $\mathcal{K}$-entailment relation.*

We can also characterise rational defeasible entailment using a subclass of $\mathcal{K}$-faithful rank functions.

Definition 12 A $\mathcal{K}$-faithful rank function r is said to be base rank preserving if the following condition holds: for all $\alpha, \beta \in \mathcal{L}$, if $br_{\mathcal{K}}(\alpha) < br_{\mathcal{K}}(\beta)$, then $r(\alpha) < r(\beta)$.

As the name indicates, base rank preserving rank functions (or rather, the relations $<$ derivable from base rank preserving rank functions) respect the base rank (or rather, the relation $<$ derivable from the base rank).

Theorem 6 *Every rational defeasible $\mathcal{K}$-entailment relation can be generated by a $\mathcal{K}$-faithful base rank preserving rank function. Conversely, every $\mathcal{K}$-faithful base rank preserving rank function generates a rational defeasible $\mathcal{K}$-entailment relation.*

The following result shows that the algorithm `DefeasibleEntailment` described in the previous section can also be used to compute rational defeasible entailment, provided it receives base rank preserving rank functions as input.

Theorem 7 *The defeasible entailment relation computed from algorithm* `DefeasibleEntailment`*, given a knowledge base $\mathcal{K}$ and a $\mathcal{K}$-faithful base rank preserving rank function, is a rational defeasible $\mathcal{K}$-entailment relation. Conversely, every rational defeasible $\mathcal{K}$-entailment relation can be computed from algorithm* `DefeasibleEntailment` *when given $\mathcal{K}$ and a $\mathcal{K}$-faithful base rank preserving rank function as input.*

The results obtained for rational defeasible entailment can therefore be summarised in the following theorem.

Theorem 8 *The following statements are equivalent.*

- $\mathrel{|\!\approx}$ *is a rational defeasible $\mathcal{K}$-entailment relation.*
- *There is a rank preserving $\mathcal{K}$-faithful ranked model $\mathscr{R}$ and a $\mathcal{K}$-faithful base rank preserving rank function r s.t.:*
  1. $r(\alpha) = \min\{i \mid v \in [\![\alpha]\!] \text{ and } \mathscr{R}(v) = i\}$*;*
  2. $\mathrel{|\!\approx}$ *can be generated from $\mathscr{R}$;*
  3. $\mathrel{|\!\approx}$ *can be generated from r;*
  4. $\mathrel{|\!\approx}$ *can be computed from algorithm* `Defeasible Entailment`*, given $\mathcal{K}$ and r as input.*

Analogous to the case for basic defeasible entailment, Point 1 of Theorem 8 establishes a connection between $\mathscr{R}$ and r via a result that is a generalisation of Observation 3.

5 Lexicographic Closure

In this section we turn our attention to *lexicographic closure*, a second form of defeasible entailment, other than rational closure, that has been studied in the literature (Lehmann 1995). Our central result is that lexicographic closure is a rational defeasible entailment relation, confirming our contention that rational defeasible entailment is a class of defeasible relations worth investigating. We also show that lexicographic closure can be characterised in three different ways: semantically via a rank preserving $\mathcal{K}$-faithful ranked model, in terms of a base preserving $\mathcal{K}$-faithful rank function r, and via the `DefeasibleEntailment` algorithm when it is presented with r (and a knowledge base $\mathcal{K}$) as input. While the semantic construction of lexicographic closure is known (Lehmann 1995), the other two constructions are new. Finally, we show that there are rational defeasible entailment relations that extend lexicographic closure, which means that lexicographic closure is not the "boldest" form of rational defeasible entailment, as has been the conjecture in the literature.

For a knowledge base $\mathcal{K}$, let $\mathcal{C}^{\mathcal{K}}$ be a function from $\mathscr{U}$ to $\mathbb{N}$ s.t. $\mathcal{C}^{\mathcal{K}}(v) = \#\{\alpha \mathrel{|\!\sim} \beta \in \mathcal{K} \mid v \Vdash \alpha \rightarrow \beta\}$.[5] So $\mathcal{C}^{\mathcal{K}}(v)$ is the number of DIs in $\mathcal{K}$ whose materialisations are satisfied by v. In defining lexicographic closure, the goal is to refine the ordering on $\mathscr{U}$ obtained from the minimal model $\mathscr{R}_{\mathcal{K}}^{RC}$ with $\mathcal{C}^{\mathcal{K}}$: in comparing two valuations with the same rank w.r.t. $\mathscr{R}_{\mathcal{K}}^{RC}$, the one with a higher number will be viewed as more typical.

Given a knowledge base $\mathcal{K}$, we define an ordering $\preceq_{LC}^{\mathcal{K}}$ on $\mathscr{U}$ as follows: $v \preceq_{LC}^{\mathcal{K}} u$ if $\mathscr{R}_{\mathcal{K}}^{RC}(u) = \infty$, or $\mathscr{R}_{\mathcal{K}}^{RC}(v) < \mathscr{R}_{\mathcal{K}}^{RC}(u)$, or $\mathscr{R}_{\mathcal{K}}^{RC}(v) = \mathscr{R}_{\mathcal{K}}^{RC}(u)$ and $\mathcal{C}^{\mathcal{K}}(v) \geq \mathcal{C}^{\mathcal{K}}(u)$. Then we let $\mathscr{R}_{\mathcal{K}}^{LC}$ be the ranked interpretation generated from $\preceq_{LC}^{\mathcal{K}}$. We refer to $\mathscr{R}_{\mathcal{K}}^{LC}$ as the lexicographic ranked model of $\mathcal{K}$.

Definition 13 The lexicographic closure $\mathrel{|\!\approx}_{LC}$ of $\mathcal{K}$ is defined as follows: $\mathcal{K} \mathrel{|\!\approx}_{LC} \alpha \mathrel{|\!\sim} \beta$ if $\mathscr{R}_{\mathcal{K}}^{LC} \Vdash \alpha \mathrel{|\!\sim} \beta$.

The next result shows that the lexicographic ranked model of $\mathcal{K}$ is $\mathcal{K}$-faithful and rank preserving.

Proposition 1 *$\mathscr{R}_{\mathcal{K}}^{LC}$ is a $\mathcal{K}$-faithful and rank preserving ranked model.*

From this result it follows immediately from Theorems 8 and 4 that lexicographic closure is a rational and basic defeasible entailment relation. In fact, Lehmann (1995, Theorem 3) already showed that lexicographic closure satisfies RC Extension.

To appreciate some of the differences between rational and lexicographic closure, consider the following example.

[5] $\#X$ denotes the cardinality of the set X.

Example 4 Figure 4 depicts the minimal ranked model $\mathscr{R}^{RC}_{\mathcal{K}}$ of $\mathcal{K} = \{\mathsf{p} \rightarrow \mathsf{b}, \mathsf{b} \mathrel{|\!\sim} \mathsf{f}, \mathsf{p} \mathrel{|\!\sim} \neg\mathsf{f}, \mathsf{b} \mathrel{|\!\sim} \mathsf{w}\}$, while Figure 5 depicts the lexicographic ranked model $\mathscr{R}^{LC}_{\mathcal{K}}$ of $\mathcal{K}$. From these two models we can see that $\mathsf{p} \mathrel{|\!\sim} \mathsf{w}$ (penguins usually have wings) is not in the rational closure of $\mathcal{K}$, but is in the lexicographic closure of $\mathcal{K}$. This is indicative of the difference between, what Lehmann refers to as Prototypical Reasoning and Presumptive Reasoning (1995). Presumptive Reasoning states that properties of a class are presumed to hold for all members of that class unless we have knowledge to the contrary. So, because birds usually have wings we assume that penguins, being birds, usually have wings as well, since we don't have information to the contrary. Contrast this with Prototypical Reasoning which states that, while typical members of a class are presumed to inherit the properties of that class, the same does not hold for atypical members. According to Prototypical Reasoning, since penguins are atypical members of the class of birds (they usually don't fly), they do not inherit the property of having wings. Rational closure operates according to Prototypical Reasoning, while lexicographic closure adheres to Presumptive Reasoning. ■

2	$\mathsf{pbf}\overline{\mathsf{w}}$ pbfw
1	$\mathsf{p}\overline{\mathsf{b}}\mathsf{f}\overline{\mathsf{w}}$ $\mathsf{p}\overline{\mathsf{b}}\mathsf{fw}$ $\mathsf{pb}\overline{\mathsf{f}}\mathsf{w}$ $\mathsf{pb}\overline{\mathsf{f}}\overline{\mathsf{w}}$
0	$\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\mathsf{w}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\mathsf{w}$ $\overline{\mathsf{p}}\mathsf{bf}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{bfw}$

Figure 4: The minimal model $\mathcal{K}$-faithful ranked model $\mathscr{R}^{RC}_{\mathcal{K}}$

5	$\mathsf{pbf}\overline{\mathsf{w}}$
4	pbfw
3	$\mathsf{pb}\overline{\mathsf{f}}\overline{\mathsf{w}}$ $\mathsf{p}\overline{\mathsf{b}}\mathsf{f}\overline{\mathsf{w}}$
2	$\mathsf{pb}\overline{\mathsf{f}}\mathsf{w}$ $\mathsf{p}\overline{\mathsf{b}}\mathsf{f}\mathsf{w}$
1	$\overline{\mathsf{p}}\mathsf{bf}\overline{\mathsf{w}}$
0	$\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\mathsf{w}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\mathsf{w}$ $\overline{\mathsf{p}}\mathsf{bf}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{bfw}$

Figure 5: The lexicographic ranked model $\mathscr{R}^{LC}_{\mathcal{K}}$

We have already seen that lexicographic closure ($\mathrel{|\!\approx}_{LC}$) can be generated from a $\mathcal{K}$-faithful rank preserving model. From Theorem 8 it then follows that there is a $\mathcal{K}$-faithful base rank preserving rank function r from which $\mathrel{|\!\approx}_{LC}$ can be generated. Furthermore, it can be generated by algorithm `DefeasibleEntailment`, given $\mathcal{K}$ and r as input. We now show how to construct the $\mathcal{K}$-faithful base rank preserving rank function r mentioned above.

Definition 14 The lexicographic rank w.r.t. a knowledge base $\mathcal{K}$ is defined as $r^{LC}_{\mathcal{K}}(\alpha) \equiv_{\text{def}} \min\{\mathscr{R}^{LC}_{\mathcal{K}}(v) \mid v \in [\![\alpha]\!]\}$.

First we show that $r^{LC}_{\mathcal{K}}$ is $\mathcal{K}$-faithful and base rank preserving.

Proposition 2 *The lexicographic rank $r^{LC}_{\mathcal{K}}$ w.r.t. a knowledge base $\mathcal{K}$ is $\mathcal{K}$-faithful and base rank preserving.*

Next, we show that $r^{LC}_{\mathcal{K}}$ generates the same rational defeasible entailment relation as $\mathscr{R}^{LC}_{\mathcal{K}}$.

Proposition 3 *$\mathscr{R}^{LC}_{\mathcal{K}} \Vdash \alpha \mathrel{|\!\sim} \beta$ if and only if $r^{LC}_{\mathcal{K}}(\alpha) < r^{LC}_{\mathcal{K}}(\alpha \wedge \neg\beta)$ or $r^{LC}_{\mathcal{K}}(\alpha) = \infty$.*

Finally, we show that the `DefeasibleEntailment` algorithm computes the same (rational) defeasible entailment relation as $\mathscr{R}^{LC}_{\mathcal{K}}$ does when given the input $\mathcal{K}$ and $r^{LC}_{\mathcal{K}}$.

Proposition 4 *Given a knowledge base $\mathcal{K}$ and a defeasible implication $\alpha \mathrel{|\!\sim} \beta$, the `DefeasibleEntailment` algorithm returns* **true** *when given the input $\mathcal{K}$, $r^{LC}_{\mathcal{K}}$, and $\alpha \mathrel{|\!\sim} \beta$ if and only if $r^{LC}_{\mathcal{K}}(\alpha) < r^{LC}_{\mathcal{K}}(\alpha \wedge \neg\beta)$, or $r^{LC}_{\mathcal{K}}(\alpha) = \infty$.*

We conclude this section by showing that, while lexicographic closure extends rational closure, it is not (always) the "boldest" form of rational defeasible entailment. To do so, we give an example of a knowledge base for which there is a rational defeasible entailment relation that extends lexicographic closure.

Example 5 Consider the knowledge base $\mathcal{K}$ in Example 4 and let a $\mathcal{K}$-faithful ranked model $\mathscr{R}$ be as depicted in Figure 5 below. It is easy to see that $\mathscr{R}$ is a refinement of the lexicographic ranked model $\mathscr{R}^{LC}_{\mathcal{K}}$ in Figure 6. It can be shown that $\mathscr{R}$ is rank base preserving, and therefore it generates a rational defeasible $\mathcal{K}$-entailment relation $\mathrel{|\!\approx}$, and that $\mathrel{|\!\approx}$ strictly extends lexicographic closure: If $\mathcal{K} \mathrel{|\!\approx}_{LC} \alpha \mathrel{|\!\sim} \beta$, then $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$, and there is at least one defeasible implication $\alpha \mathrel{|\!\sim} \beta$ such that $\mathcal{K} \mathrel{|\!\approx} \alpha \mathrel{|\!\sim} \beta$, but $\mathcal{K} \not\mathrel{|\!\approx}_{LC} \alpha \mathrel{|\!\sim} \beta$. For example, observe that $\mathcal{K} \mathrel{|\!\approx} \mathsf{b} \wedge \neg\mathsf{f} \wedge \mathsf{w} \mathrel{|\!\sim} \neg\mathsf{p}$, but $\mathcal{K} \not\mathrel{|\!\approx}_{LC} \mathsf{b} \wedge \neg\mathsf{f} \wedge \mathsf{w} \mathrel{|\!\sim} \neg\mathsf{p}$ ■

7	$\mathsf{pbf}\overline{\mathsf{w}}$
6	pbfw
5	$\mathsf{pb}\overline{\mathsf{f}}\overline{\mathsf{w}}$
4	$\mathsf{p}\overline{\mathsf{b}}\mathsf{f}\overline{\mathsf{w}}$
3	$\mathsf{pb}\overline{\mathsf{f}}\mathsf{w}$
2	$\mathsf{p}\overline{\mathsf{b}}\mathsf{f}\mathsf{w}$
1	$\mathsf{p}\overline{\mathsf{b}}\mathsf{f}\overline{\mathsf{w}}$
0	$\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\mathsf{w}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{b}\overline{\mathsf{f}}\mathsf{w}$ $\overline{\mathsf{p}}\mathsf{bf}\overline{\mathsf{w}}$ $\overline{\mathsf{p}}\mathsf{bfw}$

Figure 6: The ranked model $\mathscr{R}$ of Example 5.

6 Related Work

The original work in the KLM style (Kraus, Lehmann, and Magidor 1990) was inspired by the work of Shoham (1988),

and investigated a class of non-monotonic consequence relations, where the defeasible implication $\mid\sim$ was viewed as the (non-monotonic) form of entailment. This approach was subsequently adapted by Lehmann and Magidor (1992) to the case where $\mid\sim$ is viewed as an object-level connective for defeasible implication, and where the focus then shifts to defeasible entailment (i.e., $\mid\approx$) for a logic language that extends propositional logic with the defeasible implication connective $\mid\sim$.

When it comes to defeasible entailment, while there has been some work in this regard (Goldszmidt and Pearl 1996; Bezzazi, Makinson, and Perez 1997; Casini et al. 2014; Beierle et al. 2016; Kern-Isberner 2018), we are aware of three instances that have been studied in detail: ranked entailment (Lehmann and Magidor 1992) which is not LM-rational and is judged to be too weak, rational closure (Lehmann and Magidor 1992) and lexicographic closure (Lehmann 1995) which are both regarded as appropriate forms of defeasible entailment. Of these, rational closure is by far the best studied form of defeasible entailment (Booth et al. 2015; Booth and Paris 1998; Giordano et al. 2015).

Despite the work mentioned above, the present paper is, to the best of our knowledge, the first systematic attempt to characterise appropriate classes of defeasible entailment relations (to be distinguished from the original work by Kraus et al. (1990), which is a study of $\mid\sim$ as a form of non-monotonic consequence).

Our work is reminiscent of the AGM framework for belief change (Alchourrón, Gärdenfors, and Makinson 1985; Gärdenfors 1988), where belief change operators are studied. Taking the analogy further, rational closure can be viewed as the defeasible entailment equivalent of *full-meet belief contraction or revision* since, by virtue of the property of RC Extension, it is the most conservative of those defeasible entailment relations we regard as appropriate. Taking it even further, the boldest forms of rational defeasible entailment can be regarded as analogous to *maxichoice* belief contraction and revision. To see this, observe that maxichoice operators are obtained by imposing a linear ordering on the propositional valuations that are counter-models of a *belief set*. Similarly, the boldest forms of rational defeasible entailment are obtained by imposing a linear ordering on $\mathscr{U}_R^{\mathcal{K}}$, the set of possible valuations w.r.t. a knowledge base $\mathcal{K}$ and then considering the defeasible entailment relations generated from the base rank preserving $\mathcal{K}$-faithful ranked models obtained from such linear orderings.

Studies of defeasible entailment have also started to move beyond the propositional case, and now includes cases involving versions of defeasible implication in more expressive languages, most notably description logics (Bonatti et al. 2015; Bonatti and Sauro 2017; Britz, Meyer, and Varzinczak 2011b; Britz and Varzinczak 2018b; Casini and Straccia 2013; Giordano et al. 2013; Quantz and Royer 1992; Pensel and Turhan 2018; Casini, Straccia, and Meyer 2018) and modal logics (Boutilier 1994; Britz, Meyer, and Varzinczak 2011a; 2012). A slightly different type of extension is one in which defeasible implication is enriched by either introducing an explicit notion of typicality in propositional logic (Booth, Meyer, and Varzinczak 2012; 2013; Booth et al. 2015) or a notion of defeasible modality (Britz and Varzinczak 2017; 2018a).

7 Conclusion

The central focus of this paper is the question of determining what (defeasible) entailment means for propositional logic enriched with a defeasible implication connective. The short answer to this question provided here is that a defeasible entailment relation needs to be *rational* in the technical sense described in Section 4. In arriving at this conclusion we have made a detour through the more permissive class of *basic* defeasible entailment relations defined in Section 3. Both basic and rational defeasible entailment are characterised in four different ways, through sets of properties, semantically via ranked interpretations, in terms of ranks assigned to (propositional and defeasible) statements, and algorithmically. While basic defeasible entailment tightens the requirements imposed by KLM-style defeasible entailment somewhat, rational defeasible entailment goes further by requiring that the form of defeasible entailment known as rational closure ought to be viewed as the most basic form of defeasible entailment. Part of the argument in favour of rational defeasible entailment is that, as is the case for rational closure, lexicographic closure (the other well-known form of defeasible entailment) is also rational.

There are at least three important lines of research to which the work in this paper can lead. First on the agenda is an analysis of concrete forms of rational defeasible entailment other than rational and lexicographic closure.

Secondly, the description of both basic and rational defeasible entailment in this paper can be viewed as being on the *knowledge level* (Gärdenfors 1988) in the sense that the syntactic form of knowledge bases are, for the most part, irrelevant. But there is a strong case to be made for defining defeasible implication where syntax matters. Roughly speaking, this is analogous to the distinction between belief change on belief sets (sets closed under classical consequence) and base change (Hansson 1999), where the structure of the set of beliefs of an agent plays a role in determining how change ought to occur. In fact, although lexicographic closure is an instance of rational defeasible entailment, it is an example of a form of entailment where the structure of the knowledge base matters. Our current conjecture is that a syntax-based class of defeasible entailment will form a strict subclass of the class of rational defeasible entailment relations, and that lexicographic closure will be the strongest form of syntax-based rational defeasible entailment.

Finally, syntax-based defeasible entailment opens the door for studying the computation of defeasible entailment in more detail. We have presented an algorithm for computing any rational defeasible entailment relation, but the algorithm depends on the provision of a knowledge base $\mathcal{K}$, as well as a function that ranks all propositional (and therefore all defeasible implication) statements. With a syntax-based approach, it is possible to use the structure of $\mathcal{K}$ to rank statements, in the way that the `BaseRank` algorithm in Section 2.3 does in the process of computing rational closure.

Acknowledgements

The work of Giovanni Casini and Thomas Meyer have received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 690974 (MIREL project). The work of Thomas Meyer has been supported in part by the National Research Foundation of South Africa (grant No. UID 98019).

References

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic* 50:510–530.

Beierle, C.; Eichhorn, C.; Kern-Isberner, G.; and Kutsch, S. 2016. Skeptical, weakly skeptical, and credulous inference based on preferred ranking functions. In *ECAI 2016*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 1149–1157. IOS Press.

Bezzazi, H.; Makinson, D.; and Perez, R. P. 1997. Beyond rational monotony: Some strong non-horn rules for nonmonotonic inference relations. *J. Logic Computat.* 7(5):605–631.

Bonatti, P., and Sauro, L. 2017. On the logical properties of the nonmonotonic description logic DL^N. *Artificial Intelligence* 248:85–111.

Bonatti, P.; Faella, M.; Petrova, I.; and Sauro, L. 2015. A new semantics for overriding in description logics. *Artificial Intelligence* 222:1–48.

Booth, R., and Paris, J. 1998. A note on the rational closure of knowledge bases with both positive and negative knowledge. *Journal of Logic, Language and Information* 7(2):165–190.

Booth, R.; Casini, G.; Meyer, T.; and Varzinczak, I. 2015. On the entailment problem for a logic of typicality. In *IJCAI 2015*, 2805–2811.

Booth, R.; Meyer, T.; and Varzinczak, I. 2012. PTL: A propositional typicality logic. In *Proceedings of the 13th European Conference on Logics in Artificial Intelligence (JELIA)*, number 7519 in LNCS, 107–119. Springer.

Booth, R.; Meyer, T.; and Varzinczak, I. 2013. A propositional typicality logic for extending rational consequence. In Fermé, E.; Gabbay, D.; and Simari, G., eds., *Trends in Belief Revision and Argumentation Dynamics*, volume 48 of *Studies in Logic – Logic and Cognitive Systems*. King's College Publications. 123–154.

Boutilier, C. 1994. Conditional logics of normality: A modal approach. *Artificial Intelligence* 68(1):87–154.

Britz, K., and Varzinczak, I. 2017. From KLM-style conditionals to defeasible modalities, and back. *Journal of Applied Non-Classical Logics (JANCL)*.

Britz, K., and Varzinczak, I. 2018a. Preferential accessibility and preferred worlds. *Journal of Logic, Language and Information (JoLLI)* 27(2):133–155.

Britz, K., and Varzinczak, I. 2018b. Rationality and context in defeasible subsumption. In Woltran, S., and Ferrarotti, F., eds., *FoIKS 2018*, LNCS. Springer.

Britz, K.; Meyer, T.; and Varzinczak, I. 2011a. Preferential reasoning for modal logics. *Electronic Notes in Theoretical Computer Science* 278:55–69.

Britz, K.; Meyer, T.; and Varzinczak, I. 2011b. Semantic foundation for preferential description logics. In Wang, D., and Reynolds, M., eds., *AI 2011*, number 7106 in LNAI, 491–500. Springer.

Britz, K.; Meyer, T.; and Varzinczak, I. 2012. Normal modal preferential consequence. In Thielscher, M., and Zhang, D., eds., *AI 2012*, number 7691 in LNAI, 505–516. Springer.

Casini, G., and Meyer, T. 2017. Belief change in a preferential non-monotonic framework. In *IJCAI 2017*, 929–935.

Casini, G., and Straccia, U. 2013. Defeasible inheritance-based description logics. *JAIR* 48:415–473.

Casini, G.; Meyer, T.; Moodley, K.; and Nortje, R. 2014. Relevant closure: A new form of defeasible reasoning for description logics. In *JELIA 2014*, 92–106.

Casini, G.; Straccia, U.; and Meyer, T. 2018. A polynomial time subsumption algorithm for nominal safe $\mathcal{ELO}_\bot$ under rational closure. *Information Sciences,* https://doi.org/10.1016/j.ins.2018.09.037.

Freund, M. 1998. Preferential reasoning in the perspective of Poole default logic. *Artificial Intelligence* 98:209–235.

Gärdenfors, P. 1988. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press.

Giordano, L.; Gliozzi, V.; Olivetti, N.; and Pozzato, G. 2013. A non-monotonic description logic for reasoning about typicality. *Artificial Intelligence* 195:165–202.

Giordano, L.; Gliozzi, V.; Olivetti, N.; and Pozzato, G. 2015. Semantic characterization of rational closure: From propositional logic to description logics. *Art. Int.* 226:1–33.

Goldszmidt, M., and Pearl, J. 1996. Qualitative probabilities for default reasoning, belief revision, and causal modeling. *Artificial Intelligence* 85:57–112.

Hansson, S. 1999. *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer.

Kern-Isberner, G. 2018. Axiomatizing a qualitative principle of conditional preservation for iterated belief change. In *KR 2018*, accepted.

Kraus, S.; Lehmann, D.; and Magidor, M. 1990. Nonmonotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence* 44:167–207.

Lehmann, D., and Magidor, M. 1992. What does a conditional knowledge base entail? *Art. Int.* 55:1–60.

Lehmann, D. 1995. Another perspective on default reasoning. *Annals of Math. and Art. Int.* 15(1):61–82.

Pensel, M., and Turhan, A.-Y. 2018. Reasoning in the defeasible description logic $\mathcal{EL}_\bot$ - computing standard inferences under rational and relevant semantics. *International Journal of Approximate Reasoning* 103:28 – 70.

Quantz, J., and Royer, V. 1992. A preference semantics for defaults in terminological logics. In *KR 1992*, 294–305.

Shoham, Y. 1988. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence*. MIT Press.

How to construct Remainder Sets for Paraconsistent Revisions: Preliminary Report

Rafael Testa[1,2], **Eduardo Fermé**[1], **Marco Garapa**[1], **Maurício Reis**[1]
[1]Faculty of Exact Sciences and Engineering, University of Madeira (UMa),
Funchal, Madeira, Portugal
[2]Centre for Logic, Epistemology and the History of Science (CLE), University of Campinas (Unicamp),
Campinas, SP, Brazil

Abstract

Revision operation is the consistent expansion of a theory by a new belief-representing sentence. We consider that in a paraconsistent setting this desideratum can be accomplished in at least three distinct ways: the output of a revision operation should be either non-trivial or non-contradictory (in general or relative to the new belief). In this paper those distinctions will be explored in the constructive level by showing how the *remainder sets* could be refined, capturing the key concepts of paraconsistency in a dynamical scenario. These are preliminaries results of a wider project on Paraconsistent Belief Change conduced by the authors.

Introduction

In a working group at BRAON'17 (Third Madeira Workshop on Belief Revision, Argumentation, Ontologies, and Norms), the very definition of revision was discussed in the context of an inconsistent-tolerant setting: given the logical possibility of contradictory but non-trivial belief sets (a direct consequence of considering an underlying paraconsistent logic), some authors propose that the revision could be understood as a plain expansion (cf. for instance (Priest 2001; Girard and Tanaka 2016)). The questions adduced by the referred working group were: could it still be rationally and even logically possible for the reasoner to demand from a revision operator a non-contradictory output in a paraconsistent scenario? If so, which definition of revision should be considered? Is it really necessary to equate revision with a plain expansion?

Paraconsistent logics are based on the study of contradictory yet non-trivial theories, exposing a clear distinction between *triviality* and *contradictoriness*. As we understand it, the classical desideratum of consistency, in a paraconsistent setting, splits itself into two distinct ones: non-triviality and non-contradiction. More: since contradictions are distinct, this last concept can be considered with respect to a specific belief-representing sentence (namely, the input). In this paper the relation between those will be constructively explored. We suggest new constructions for remainder sets that fulfill the above desiderata and also circumvent some issues advanced by the literature, as the failure of *extensionality* in general.

On AGM

AGM-style belief revision describes an idealized agent, with a potentially infinite set of belief-representing sentences closed under logical consequence. To express the closure, we are going to use the consequence operator Cn: for a given underlying logic $\mathbf{L}$, $K \vdash_{\mathbf{L}} \alpha$ if and only if $\alpha \in Cn(K)$. Hence the criterion that K is closed under logical consequence can be formally expressed by:

$$K = Cn(K)$$

The agent's dynamics is given by operations that describe the change from one belief set to another. These operations are:

Expansion. An expansion occurs when new information is simply added to the set of the beliefs of an agent. As a result of an expansion, the belief set can become inconsistent. The outcome of an expansion of a belief set K by a sentence α will be denoted by $K + \alpha$.

Contraction. A contraction occurs when information is removed from the set of beliefs of an agent. The result of a contraction of K by a sentence α will be denoted by $K - \alpha$.

Revision: A revision occurs when new information is added to the agent's belief set. When performing a revision some beliefs can be removed in order to ensure consistency. Contrary to expansion, revision preserves consistency (unless the new information is itself inconsistent). The result of a revision of a belief set K by a sentence α will be denoted by $K * \alpha$.

Formally we have the following:

Definition 1. *The expansion of K by α ($K + \alpha$) is given by*

$$K + \alpha = Cn(K \cup \{\alpha\})$$

The explicit construction for contraction adopted is the *partial meet contraction*, constructed as follows (the results of this section are from (Alchourrón, Gärdenfors, and Makinson 1985)):

1. Choose some maximal subsets of K (with respect the inclusion) that do not entail α.
2. Take the intersection of such sets.

The *remainder* of K and α is the set of all maximal subsets of K that do not entail α. Formally the definition is the following:

Definition 2 (Remainder)**.** *The set of all the maximal subsets of K that do not entail α is called the* remainder set *of K by α and is denoted by $K\perp\alpha$, that is, $K' \in K\perp\alpha$ iff:*

(i) $K' \subseteq K$.

(ii) $\alpha \notin Cn(K')$.

(iii) *If* $K' \subset K'' \subseteq K$ *then* $\alpha \in Cn(K'')$.

Typically $K\perp\alpha$ may contain more than one maximal subset. The main idea constructing a contraction function is to apply a *selection function* γ which intuitively selects the sets in $K\perp\alpha$ containing the beliefs that the agent holds in higher regard (those beliefs that are more entrenched).

Definition 3 (selection function)**.** *A selection function for K is a function γ such that, for every α:*

1. $\gamma(K\perp\alpha) \subseteq K\perp\alpha$ *if* $K\perp\alpha \neq \emptyset$.
2. $\gamma(K\perp\alpha) = \{K\}$ *otherwise.*

The *partial meet contraction* is the intersection of the sets of $K\perp\alpha$ selected by γ.

Definition 4 (partial meet contraction)**.** *Let K be a belief set, and γ a selection function for K. The partial meet contraction on K that is generated by γ is the operation $-_\gamma$ such that for all sentences α:*

$$K -_\gamma \alpha = \bigcap \gamma(K\perp\alpha).$$

By the Levi identity, revision $K * \alpha$ is defined as a prior contraction by $\neg\alpha$ followed by a expansion by α. As it can be easily understood, the prior contraction assures the consistency of the result.

The partial meet revision (the construction for revision defined over the partial meet contraction) is defined as follows.

Definition 5 (partial meet revision)**.** *Let K be a belief set, and γ a selection function for K. The partial meet revision on K that is generated by γ is the operation $*_\gamma$ such that for all sentences α:*

$$K *_\gamma \alpha = \big(\bigcap \gamma(K\perp \neg\alpha)\big) + \alpha$$

An operation $*$ on K is a partial meet revision if and only if there is a selection function γ for K such that for all sentences $\alpha : K * \alpha = K *_\gamma \alpha$.

Partial meet revision is axiomatically characterized as follows:

Observation 6. *The operator $*$ is an operator of partial meet revision for a belief set K if and only if it satisfies the following postulates:*

(K∗1) $K * \alpha = Cn(K * \alpha)$. **(Closure)**

(K∗2) $\alpha \in K * \alpha$. **(Success)**

(K∗3) $K * \alpha \subseteq K + \alpha$. **(Inclusion)**

(K∗4) *If* $K + \alpha$ *is consistent, then* $K * \alpha = K + \alpha$. **(Vacuity)**

(K∗5) *If* α *is consistent, then* $K * \alpha$ *is consistent* **(Consistency)**

(K∗6) *If* $Cn(\alpha) = Cn(\beta)$*, then* $K * \alpha = K * \beta$. **(Extensionality)**

On Paraconsistent Belief Revision

Some approachs on Paraconsistent Belief can be found, for instance, in (Restall and Slaney 1995), (Chopra and Parikh 1999), (Tamminga 2001), (Priest 2001), (Mares 2002), (Girard and Tanaka 2016) and (Testa, Coniglio, and Ribeiro 2017). A brief overview on some of these inquiries can be found in (Fermé and Hansson 2018). The main objective of this work is to refine some results of the so-called AGMp system, following directly the original AGM model (with suitable adjustments), advanced in (Testa, Coniglio, and Ribeiro 2017). This system is designed over a class of paraconsistent logics called **LFI**s to be further introduced.

Paraconsistent Logics and LFIs

The Logics of Formal Inconsistency (**LFIs**), advanced by (Carnielli and Marcos 2002) and further developed mainly in (Carnielli, Coniglio, and Marcos 2007) are a family of paraconsistent logics that encompasses most of paraconsistent systems with a supraclassical character, where it is possible to re-encode classical reasoning within it (cf. (Carnielli and Coniglio 2016) by a comprehensive textbook).

Roughly speaking, withing LFIs it is possible to express the notions of inconsistency and consistency inside the object language. The sentential unary connective $\circ$ of *formal consistency* is the more frequently used, where the sentence $\circ\alpha$ is intended to formally express the meaning that 'α is consistent'. As a consequence, contradiction does not generate triviality in general, unless the sentence involved is consistent. In formal terms, for any logic $\mathbf{L}$ that is a LFI, denoted by a consequence operator $\vdash_{\mathbf{L}}$, the following does not hold:

Observation 7. *Explosion principle*

$$\alpha, \neg\alpha \vdash_{\mathbf{L}} \beta,$$

but a distinct form of it is always the case:

Observation 8. *Gentle explosion principle*

$$\alpha, \neg\alpha, \circ\alpha \vdash_{\mathbf{L}} \beta.$$

The distinctions given by the LFIs not only separates the notion of contradiction from deductive triviality (like every paraconsistent logic), but also contradiction from inconsistency as well non-contradiction from consistency (in the object language level). So there is a clear distinction between contradictions that can be accepted from those that cannot. The idea to be captured is that no matter the nature of the contradictions a reasoner is willing to accept, there still are contradictions that cannot be accepted at all.

Definition 9. *In order to avoid ambiguity, the following notation is useful (for any logic* $\mathbf{L}$ *represented by Cn):*

a. *A set A is contradictory if and only if there is a sentence β such that $\{\beta, \neg\beta\} \subseteq Cn(A)$.*

b. *A set A is contradictory with respect to α (or is α-contradictory) if and only if, for some β equivalent to α (that is, for α and β such that $Cn(\alpha) = Cn(\beta)$), $\{\beta, \neg\beta\} \subseteq Cn(A)$.*

c. *A set A is trivial if and only $\perp \in Cn(A)$.*

d. *A sentence α is contradictory if and only if the set $\{\alpha\}$ is contradictory.*

e. *A sentence α is trivial if and only if the set $\{\alpha\}$ is trivial.*

Definition 10. *The most basic* **LFI** *in the family considered is the propositional logic* **mbC**. *The language $\mathbb{L}$ of* **mbC** *is generated by the connectives $\wedge, \vee, \rightarrow, \neg, \circ$.*

Definition 11 (**mbC** (Carnielli and Marcos 2002))**.** *The logic* **mbC** *is defined over the language $\mathbb{L}$ by means of a Hilbert system as follows:*

Axioms:

(A1) $\alpha \rightarrow (\beta \rightarrow \alpha)$
(A2) $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow (\beta \rightarrow \delta)) \rightarrow (\alpha \rightarrow \delta))$
(A3) $\alpha \rightarrow (\beta \rightarrow (\alpha \wedge \beta))$
(A4) $(\alpha \wedge \beta) \rightarrow \alpha$
(A5) $(\alpha \wedge \beta) \rightarrow \beta$
(A6) $\alpha \rightarrow (\alpha \vee \beta)$
(A7) $\beta \rightarrow (\alpha \vee \beta)$
(A8) $(\alpha \rightarrow \delta) \rightarrow ((\beta \rightarrow \delta) \rightarrow ((\alpha \vee \beta) \rightarrow \delta))$
(A9) $\alpha \vee (\alpha \rightarrow \beta)$
(A10) $\alpha \vee \neg\alpha$
(bc1) $\circ\alpha \rightarrow (\alpha \rightarrow (\neg\alpha \rightarrow \beta))$

Inference Rule:

(Modus Ponens) $\alpha, \alpha \rightarrow \beta \vdash \beta$

It is worth of noticing that (A1)-(A9) plus *Modus Ponens* constitutes an axiomatization for the classical positive logic $\mathbf{CPL}^{+}$, so that **mbC** can be understood as a extension of it, adding few constraints on negation and formal consistency by axioms (A10) and (bc1). Further constraints can be given by the axioms of **mbC**'s extensions, for instance: (ciw) $\circ\alpha \vee (\alpha \wedge \neg\alpha)$, (ci) $\neg\circ\alpha \rightarrow (\alpha \wedge \neg\alpha)$, (cl) $\neg(\alpha \wedge \neg\alpha) \rightarrow \circ\alpha$, (cf) $\neg\neg\alpha \rightarrow \alpha$, (ce) $\alpha \rightarrow \neg\neg\alpha$ and (cc) $\circ\circ\alpha$. A detailed taxonomy on **LFI**s can be found on the references.

Regarding implication, recall that deduction holds for *any* propositional logic where (A1) and (A2) can be derived when MP is the unique inference rule.

Observation 12 (deduction meta-theorem (Carnielli, Coniglio, and Marcos 2007))**.** *The* **mbC** *calculus satisfies the following:*
$\Gamma, \alpha \vdash_{mbC} \beta$ *iff* $\Gamma \vdash_{mbC} \alpha \rightarrow \beta$

Regarding paraconsistent negation, note that $\mathbf{CPL}^{+}$plus $\alpha \vee \neg\alpha$ is too weak (as expected). This axiom reflects that the truth-value of α partially determines the truth-value of $\neg\alpha$: if α is false, then $\neg\alpha$ must be true; but if α is true, $\neg\alpha$ can be either true or false. The only axiom that deals with the formal consistency in **mbC** is $\circ\alpha \rightarrow (\alpha \rightarrow (\neg\alpha \rightarrow \beta))$: similarly, if both α and $\neg\alpha$ are true, $\circ\alpha$ must be false.

Definition 13 (Valuations for **mbC** (Carnielli and Coniglio 2016))**.** *A function $v : \mathbb{L} \rightarrow \{0, 1\}$ is a* valuation for **mbC** *if it satisfies the following clauses:*
($v\wedge$) $v(\alpha \wedge \beta) = 1 \Leftrightarrow v(\alpha) = 1$ *and* $v(\beta) = 1$.
(Conjunction)
($v\vee$) $v(\alpha \vee \beta) = 1 \Leftrightarrow v(\alpha) = 1$ *or* $v(\beta) = 1$
(Disjunction)
($v\rightarrow$) $v(\alpha \rightarrow \beta) = 1 \Leftrightarrow v(\alpha) = 0$ *or* $v(\beta) = 1$
(Implication)
($v\neg$) $v(\neg\alpha) = 0 \Rightarrow v(\alpha) = 1$
(Paraconsistent/Weak negation)
($v\circ$) $v(\circ\alpha) = 1 \Rightarrow v(\alpha) = 0$ *or* $v(\neg\alpha) = 0$
(Formal Consistency)

The semantical consequence relation associated to valuations for **mbC** is defined as expected: $X \models_{\mathbf{mbC}} \alpha$ iff, for every **mbC**-valuation v, if $v(\beta) = 1$ for every $\beta \in X$ then $v(\alpha) = 1$. The following result is well-known:

Observation 14 (Adequacy of **mbC** w.r.t. bivaluations (Carnielli and Coniglio 2016))**.** *For every set of formulas $X \cup \{\alpha\}$: $X \vdash_{\mathbf{mbC}} \alpha$ if and only if $X \models_{\mathbf{mbC}} \alpha$.*

Remark 15. *Despite of the fact that we are considering in this presentation the logic* **mbC** *and extensions, it is worth noticing that the constructions here depends on more general restrictions, so that they can encompass a wider class of logics.*

Remark 16 (derived bottom particle and strong negation)**.** *The* falsum *(or bottom) is defined in* **mbC** *by means of the formula $\bot_\beta =_{def} \beta \wedge \neg\beta \wedge \circ\beta$, for any formula β. From this, the classical (or strong) negation is defined in* **mbC** *by $\sim_\beta\alpha =_{def} (\alpha \rightarrow \bot_\beta)$. Since $\bot_\beta$ and $\bot_{\beta'}$ are interderivable in* **mbC**, *for any β and β', then $\sim_\beta\alpha$ and $\sim_{\beta'}\alpha$ are also interderivable. Hence, the strong negation of α will be denoted simply by $\sim\alpha$. The same applies to $\bot$.*

The following propositions may prove useful for assessment of further results (they can be easily checked by valuations of Definition 13).

Proposition 17 (some properties of **mbC**)**.** *The following hold:*

i. $\bot \vdash \alpha$
ii. $\sim\alpha \vdash \neg\alpha$ *and so* $\vdash \sim\alpha \rightarrow \neg\alpha$
iii. $\circ\alpha \wedge \neg\alpha \vdash \sim\alpha$, *but* $\sim\alpha \nvdash \circ\alpha \wedge \neg\alpha$
iv. $\neg\neg\alpha \nvdash \alpha$

Remark 18. *As usual, $\alpha \leftrightarrow \beta$ is an abbreviation for $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.*

Proposition 19. *The following hold in* **mbC***:*

$\alpha \leftrightarrow \beta \nvdash \neg\alpha \leftrightarrow \neg\beta$

Since a classical negation $\sim$ can be defined in **mbC**, that logic can be understood as an expansion of the classical propositional logic **CPL** by adding a paraconsistent negation $\neg$ and a consistency operator $\circ$ satisfying certain axioms.

In formal terms, consider **CPL** defined over the language $\mathbb{L}_0$ generated by the connectives $\wedge, \vee, \rightarrow, \neg$ (observe that in $\mathbb{L}_0$ $\neg$ represents the classical negation instead of the paraconsistent negation of **mbC**). If $Y \subseteq \mathbb{L}_0$ then $\circ(Y) = \{\circ\alpha : \alpha \in Y\}$. Then, the following result can be obtained:

Observation 20 (Derivability Adjustment Theorem (Carnielli, Coniglio, and Marcos 2007))**.** *Let $X \cup \{\alpha\}$ be a set of formulas in $\mathbb{L}_0$. Then $X \vdash_{\mathbf{CPL}} \alpha$ if and only if $\circ(Y), X \vdash_{\mathbf{mbC}} \alpha$ for some $Y \subseteq \mathbb{L}_0$.*

Remark 21. *From now on, let us assume a* **LFI***, namely* **L**=$\langle \mathbb{L}, Cn_{\mathbf{L}} \rangle$, *such that* **L** *is an extension of* **mbC**. *Since the context is clear, we will omit the subscript, and simply denote the closure by Cn.*

The AGMp system

Let us assume a non-trivial state K such that $K = Cn(K)$.

Partial meet AGMp revisions In (Testa, Coniglio, and Ribeiro 2017) it is shown that a paraconsistent revision $K*\alpha$ can be defined by Levi identity as in classical AGM, that is, by a prior contraction by $\neg\alpha$ followed by a expansion by α (Definition 5). It is worth of noticing that one of the focus of that paper was showing the possibility of defining *external revision* for paraconsistent closed theories, in the sense of *reverse Levi identity* as defined by Hansson for Belief Bases. For our intends and purposes, this construction will not be taken into consideration – nevertheless, it should be noted that the results here advanced applies when taking into consideration the proper features of that operation.

In terms of postulates, the AGMp internal revision is characterized as the classical operation, but without the *extensionality* postulate, and changing *consistency* by *non-contradiction*. It should be noticed that in (Testa, Coniglio, and Ribeiro 2017) *vacuity* was replaced by *relevance*, since it was proven that both postulates are equivalent in standard, supraclassical and deductive logics). So the following holds:

Observation 22. *(Testa, Coniglio, and Ribeiro 2017) The operator $*$ is an operator of AGMp partial meet internal revision for a belief set K if and only if it satisfies closure, success, inclusion, vacuity and the following:*
(K$*$5') *if α is non-contradictory, then $K * \alpha$ is not contradictory* **(non-contradiction)**

The postulate of *non-contradiction* above is an adaptation of the classical postulate of *consistency*. That's exactly the necessity of still demanding a non-contradictory output for revision operation that will be further discussed. Furthermore, AGMp presupposes that K itself is also non-contradictory – in fact, in order to keep generality (in the sense of taking into account a contradictory belief set as an input), it could be said that (K$*$5') should specifically guarantees that $K * \alpha$ in not α-contradictory.

Extensionality lost The weakness of a paraconsistent negation has the advantages of allowing contradictions. Nevertheless, this same property come with the cost of loosing extensionality in general. By definition 5 and the negative results of propositions 19 and 17(iv), it is easy to check that, given the paraconsistent negation properties, partial meet paraconsistent revision is not extensional.

In order to restore a suitable form of extensionality, some assumptions on the underlying logic should be made, as proposed by (Testa, Coniglio, and Ribeiro 2017). We advance a refinement in the constructions in order to preserve that postulate in weaker LFIs (and other paraconsistent logics).

Non-contradiction *vs*.Triviality It is clear that AGMp assumes that the output of a revision should still be non-contradictory (with respect to the input). Despite that fact, a non-trivial revision was suggested in that paper, defined by the Levi identity applied to the strong negation.

Definition 23.

$$K *_\gamma \alpha = \big(\bigcap \gamma(K \perp \sim \alpha)\big) + \alpha$$

By proposition 17 iii. and definition of *remainder* it is easy to check that this construction assures that the output is not trivial, by retracting $\neg\alpha$ or $\circ\alpha$, as long α itself is non-trivial.

Some Refinements on the Paraconsistent Framework

In order to restore extensionality in the paraconsistent setting, as well to better capture the distinction between *non-contradictoriness* and *triviality*, a new definition of reminder set is advanced. It will be shown that, in classical Belief Revision, this construction is equivalent to the classical reminder set. Furthermore, in the paraconsistent setting, this construction defines a revision where the output is demanded to be non-trivial (denoted by $\breve{*}$), and suitable modifications on it defines revisions in which the output is non-contradictory with respect to the input (denoted by $\bar{*}$), and non-contradictory in general (denoted by $\bar{\bar{*}}$) – those revisions, when the underlying logic is CPL, are proven to be equivalent with the classical one. Those features, as we understand, captures the results of the *Derivability Adjustment Theorem*, advanced in the proposition 20.

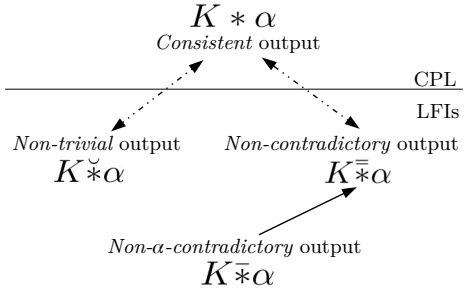


Figure 1: Relation between the revisions

Remainder sets: new constructions to revisions

Recall the definition of *remainder set*: for defining classical revision $K * \alpha$, we want to expand by α the intersection of some maximal subset of K that does not entail $\neg\alpha$ – in logical terms, a remainder for revision is designed to be a "$\neg\alpha$-saturated" subset of K.

Non-trivial remainder We define the *remainder* of K with respect to α as the set of all maximal subsets of K that, when expanded by α, are non-trivial (that is, do not entail $\perp$). This modification goes in the line of the one presented by (Delgrande 2008), advanced for Horn clause contraction function but here understood in the general context of an operation in logics without negation, as suggested by (Ribeiro 2012). We do, of course, have a negation – but given some weak properties of it (like the loss of extensionality as advanced before) the idea is to design the remainder not relaying on that.[1]

[1]There are some authors that discuss what are the necessary and sufficient conditions for a negation to be, effectively, a negation, taking the paraconsistent one as an example. This analysis

Formally the definition is the following:

Definition 24 (non-trivial remainder)**.** *Let K be a belief set, and let α be a formula. A set $X \in K \Downarrow \alpha$ if and only if:*

(i) $X \subseteq K$.
(ii) $X \cup \{\alpha\} \nvdash \bot$.
(iii) *If $X \subset X' \subseteq K$ then $X' \cup \{\alpha\} \vdash \bot$.*

$K \Downarrow \alpha$ is the non-trivial remainder of K with respect to α.

Remark 25. *It is clear that if $X \in K \Downarrow \alpha$, then $\{\neg\alpha, \circ\alpha\} \not\subseteq Cn(X)$, and that $X \cup \{\alpha\} \nvdash \beta \wedge \neg\beta \wedge \circ\beta$ for all β.*

Non-trivial revision A selection function for K is a γ defined as above. The *partial meet non-trivial revision* is, also, the intersection of the sets chosen by the selection function expanded by α.

Definition 26. *Let K be a belief set, and γ a selection function for K. The partial meet non-trivial revision on K that is generated by γ is the operation $\breve{*}_\gamma$ such that for all sentences α:*

$$K\breve{*}_\gamma\alpha = \bigcap \gamma(K \Downarrow \alpha) + \alpha.$$

An operation $\breve{*}$ is a partial meet non-trivial revision if and only if there is a selection function γ for K such that for all sentences $\alpha : K\breve{*}\alpha = K\breve{*}_\gamma\alpha$

It should be noticed that this operation does not explicitly use the construction of a contraction operator, as it is classically done by AGM – where revision is defined by Levi identity, alluding the contraction by the negated formula (cf. def. 5). Instead, the sentences to be retracted in order to accommodate the new belief-representing sentence are chosen directly by the revision's construction. The same happens with the further revisions to be presented.

Of course contraction could still be defined by a Harper-like identity, but that's not our focus on this paper.

Non-contradictory remainder with respect to the input A less permissive remainder can be defined – relative to contradictions. In a nutshell, it is designed to retract $\neg\beta$ from K, for all β equivalent to the new belief-representing sentence α – re-encoding the characteristics of the classical one, but now relative to a paraconsistent setting (endowed with a weak negation).

Definition 27 (non-contradictory remainder with respect to the input)**.** *Let K be a belief set, and let α be a formula. A set $X \in K{\downarrow_\alpha}\alpha$ if and only if:*

(i) $X \subseteq K$.
(ii) *For all $\beta \equiv \alpha$, $X \cup \{\alpha\} \nvdash \beta \wedge \neg\beta$.*
(iii) *If $X \subset X' \subseteq K$ then there exists a sentence $\beta \equiv \alpha$ such that $X' \cup \{\alpha\} \vdash \beta \wedge \neg\beta$*

$K{\downarrow_\alpha}\alpha$ is the non-α-contradictory remainder of K with respect to α.

Remark 28. *It is clear that if $X \in K{\downarrow_\alpha}\alpha$, then $\{\neg\alpha\} \not\subseteq Cn(X)$. More: $X \nvdash \neg\beta$ for all $\beta \equiv \alpha$.*

is of interest to Belief Revision, since much of the properties carried out by the systems tacitly assumes the properties of a classical negation.

Non-contradictory revision with respect to the input

Definition 29. *Let K be a belief set, and γ a selection function for K. The partial meet non-contradictory revision with respect to α on K that is generated by γ is the operation $\bar{*}_\gamma$ such that for all sentences α:*

$$K\bar{*}_\gamma\alpha = \bigcap \gamma(K{\downarrow_\alpha}\alpha) + \alpha.$$

An operation $\bar{*}$ is a partial meet non-contradictory revision with respect to α if and only if there is a selection function γ for K such that for all sentences $\alpha : K\bar{*}\alpha = K\bar{*}_\gamma\alpha$

Non-contradictory remainder

Definition 30 (non-contradictory remainder)**.** *Let K be a belief set, and let α be a formula. A set $X \in K \downarrow \alpha$ if and only if:*

(i) $X \subseteq K$.
(ii) *For all $\beta \in \mathbb{L}$, $X \cup \{\alpha\} \nvdash \beta \wedge \neg\beta$.*
(iii) *If $X \subset X' \subseteq K$ then there exists a sentence $\beta \in \mathbb{L}$ such that $X' \cup \{\alpha\} \vdash \beta \wedge \neg\beta$*

$K \downarrow \alpha$ is the non-contradictory remainder of K with respect to α.

Remark 31. *It is clear that if $X \in K{\downarrow}\alpha$, then $\{\neg\alpha\} \not\subseteq Cn(X)$. More: $X \nvdash \beta \wedge \neg\beta$ for all $\beta \in \mathbb{L}$.*

Non-contradictory revision

Definition 32. *Let K be a belief set, and γ a selection function for K. The partial meet non-contradictory revision on K that is generated by γ is the operation $\bar{\bar{*}}_\gamma$ such that for all sentences α:*

$$K\bar{\bar{*}}_\gamma\alpha = \bigcap \gamma(K \downarrow \alpha) + \alpha.$$

An operation $\bar{\bar{*}}$ is a partial meet non-contradictory revision if and only if there is a selection function γ for K such that for all sentences $\alpha : K\bar{\bar{*}}\,\alpha = K\bar{\bar{*}}_\gamma\alpha$

Relation between the remainders

Proposition 33. *The following identities hold:*

i. $K \Downarrow \alpha = K\bot(\alpha \to \bot) = K\bot \sim \alpha$
ii. $K\bot\alpha = K \Downarrow (\alpha \to \bot) = K \Downarrow\!\sim \alpha$

33.i. is an expected result, given definition 23. As it can be perceived, 33.ii. is an intermediate result for further defining contraction via non-trivial remainder.

Proposition 34. *In general, the remainder sets $K\bot\neg\alpha$, $K\Downarrow\alpha$, $K\downarrow\alpha$ and $K{\downarrow_\alpha}\alpha$ are different from each other.*

This is a predictable feature, since the concepts of triviality and contradiction are distinct in a paraconsistent setting and, moreover, contradictions are distinct to each other.

However, given the fact that in **CPL** all contradictions are alike, and equivalent to $\bot$, it is easy to check the following:

Proposition 35 (The classical collapse of consistency, non-triviality, non-contradictoriness and non-α-contradictoriness)**.** *When the underlying logic is* **CPL***:*

$$K\bot\neg\alpha = K \Downarrow \alpha = K \downarrow \alpha = K \downarrow_\alpha \alpha$$

From construction to postulates

In this section we will present each one of the paraconsistent revision functions through a set of postulates that determine the behavior of each one of these functions – establishing conditions or constrains that they must satisfy, as it is classically done. Through the postulates, the refinement made in the constructive level in order to capture the distinction between non-contradictoriness and triviality can be highlighted. In the paraconsistent setting, the consistency desideratum (classicaly captured by the *consistency* postulate) adduce three distinct new postulates: *non-triviality*, *non-α-contradiction* and *non-contradiction*, capturing respectively the intuition that the revision output should be non-trivial, non-contradictory relative to the new information to be incorporated or non-contradictory in general.

Another important feature of the new constructions advanced in this paper is that the revision of a belief set by logical equivalent sentences produces the same output in general – captured by the postulate of *extensionality*. Recall that this property was not valid in general in paraconsistent systems, as aforementioned.

Non-trivial partial meet revision

Proposition 36. *If $\breve{*}$ is an operator of non-trivial partial meet revision for a belief set K, then it satisfies the following postulates:*

(K$\breve{*}$1) $K\breve{*}\alpha = Cn(K\breve{*}\alpha)$. **(Closure)**

(K$\breve{*}$2) $\alpha \in K\breve{*}\alpha$. **(Success)**

(K$\breve{*}$3) $K\breve{*}\alpha \subseteq K + \alpha$. **(Inclusion)**

(K$\breve{*}$4) *If $K + \alpha$ is non-trivial, then $K\breve{*}\alpha = K + \alpha$.* **(Vacuity)**

(K$\breve{*}$5) *If α in non-trivial, then $K\breve{*}\alpha$ is non-trivial.* **(Non-triviality)**

(K$\breve{*}$6) *If $Cn(\alpha) = Cn(\beta)$, then $K\breve{*}\alpha = K\breve{*}\beta$.* **(Extensionality)**

Example 37. *Let $K = Cn(\{\neg\alpha, \gamma, \neg\gamma\})$. It is clear that K is non-trivial, since $\circ\gamma \notin K$.*
It can be easily checked that $\neg\alpha \in K\breve{*}\alpha$, since $\circ\alpha \notin K$. Furthermore, this operation does not retract γ nor $\neg\gamma$ from K.

Remark 38. *By the very definition of vacuity, it is clear that in weaker paraconsistent logics where there is no primitive or defined formal consistency operator (or, equivalently, where there is no strong negation nor bottom particle), a non-trivial revision is a plain expansion.*

Non-contradictory partial meet revision with respect to the input

Proposition 39. *If $\bar{*}$ is an operator of non-contradictory (with respect to the input) partial meet revision for a belief set K, then it satisfies the following postulates:*

(K$\bar{*}$1) $K\bar{*}\alpha = Cn(K\bar{*}\alpha)$ **(Closure)**

(K$\bar{*}$2) $\alpha \in K\bar{*}\alpha$ **(Success)**

(K$\bar{*}$3) $K\bar{*}\alpha \subseteq K + \alpha$ **(Inclusion)**

(K$\bar{*}$4) *If $K+\alpha$ is non-α-contradictory, then $K\bar{*}\alpha = K+\alpha$* **(Vacuity)**

(K$\bar{*}$5) *If α is non-contradictory, $K\bar{*}\alpha$ is non-α-contradictory* **(Non-α-contradiction)**

(K$\bar{*}$6) *If $Cn(\alpha) = Cn(\beta)$, then $K\bar{*}\alpha = K\bar{*}\beta$* **(Extensionality)**

Example 40. *Let K be the same of Example 37. It can be easily checked that $\neg\alpha \notin K\bar{*}\alpha$, but γ and $\neg\gamma$ are still in K.*

Non-contradictory partial meet revision

Proposition 41. *If $\bar{\bar{*}}$ is an operator of non-contradictory partial meet revision for a belief set K, then it satisfies the following postulates:*

(K$\bar{\bar{*}}$1) $K\bar{\bar{*}}\alpha = Cn(K\bar{\bar{*}}\alpha)$ **(Closure)**

(K$\bar{\bar{*}}$2) $\alpha \in K\bar{\bar{*}}\alpha$ **(Success)**

(K$\bar{\bar{*}}$3) $K\bar{\bar{*}}\alpha \subseteq K + \alpha$ **(Inclusion)**

(K$\bar{\bar{*}}$4) *If $K + \alpha$ is non-contradictory, then $K\bar{\bar{*}}\alpha = K + \alpha$* **(Vacuity)**

(K$\bar{\bar{*}}$5) *If α is non-contradictory, $K\bar{\bar{*}}\alpha$ is non-contradictory* **(Non-contradiction)**

(K$\bar{\bar{*}}$6) *If $Cn(\alpha) = Cn(\beta)$, then $K\bar{\bar{*}}\alpha = K\bar{\bar{*}}\beta$* **(Extensionality)**

Example 42. *Let K be the same of Example 37. It can be easily checked that $\neg\alpha \notin K\bar{\bar{*}}\alpha$. Furthermore, this operation also retracts γ or $\neg\gamma$ from K.*

Final Remarks and future works

In a classical setting, ensuring that the negation of the formula to be incorporated is not in the output is necessary and sufficient condition to keep the output non-trivial and forcibly non-contradictory. In paraconsistent reasoning, however, this condition is not necessary in order to ensure non-triviality (since contradictions do not entail triviality in general) nor sufficient in order to ensure non-contradictoriness (since negation is non-extensional in the sense that logically equivalent formulas do not have equivalent negated formulas in general). That asymmetry gives rise to at least three distinct paraconsistent revisions, entailed by more fine-tuned remainders.

Regarding the questions posed at the introduction of this paper, it is worth of noticing that assuming that paraconsistent revision is equivalent with a plain expansion presupposes that (i) consistency is necessarily equivalent to non-triviality in a paraconsistent setting and, furthermore, (ii) that all paraconsistent logics do not endow a bottom particle (primitive or defined). Both assumptions, as we've shown, are not true.

Recovering the extensionality in general is the first step for defining transitively relational partial meet paraconsistent revisions: by considering transitively relational selection functions γ in the remainder sets and, accordingly, by taking into account the supplementary postulates as originally advanced by classical AGM (providing the respective representation theorems).

Acknowledgments.

These are preliminaries results of the project on Paraconsistent Belief Revision, sponsored by FAPESP to Testa, project number 2017/10836-0. Fermé is partially supported by FCT MCTES and NOVA LINCS UID/CEC/04516/2013 and FCT MCTES IC&DT Project AAC 02/SAICT/2017. Garapa and Reis are supported by FCT through projects UID/MAT/04674/2013 (CIMA) and PTDC/CCI-COM/30990/2017. For citations, please check the authors' recent publications on the subject for up-to-date results.

References

Alchourrón, C. E.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic* 50(2):510–530. doi: 10.2307/2274239.

Carnielli, W., and Coniglio, M. E. 2016. *Paraconsistent Logic: Consistency, Contradiction and Negation*, volume 40 of *Logic, Epistemology, and the Unity of Science*. Springer. doi: 10.1007/978-3-319-33205-5.

Carnielli, W., and Marcos, J. a. 2002. A taxonomy of C-systems. In Carnielli, W. A.; Coniglio, M. E.; and D'Ottaviano, I. M. L., eds., *Paraconsistency: The Logical Way to the Inconsistent. Proceedings of the 2nd World Congress on Paraconsistency (WCP 2000)*, volume 228 of *Lecture Notes in Pure and Applied Mathematics*, 1–93. New York: Marcel Dekker.

Carnielli, W.; Coniglio, M. E.; and Marcos, J. a. 2007. Logics of Formal Inconsistency. In Gabbay, D. M., and Guenthner, F., eds., *Handbook of Philosophical Logic (2nd. edition)*, volume 14, 1–93. Springer. doi: 10.1007/978-1-4020-6324-4_1.

Chopra, S., and Parikh, R. 1999. An inconsistency tolerant model for belief representation and belief revision. In Dean, T., ed., *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99)*, volume 1, 192–197. Stockholm: Morgan Kaufmann.

Delgrande, J. P. 2008. Horn clause belief change: Contraction functions. *Proceedings of KR-08* 156–165.

Fermé, E., and Hansson, S. O. 2018. *Belief Change: Introduction and Overview*. SpringerBriefs in Intelligent Systems, Springer.

Girard, P., and Tanaka, K. 2016. Paraconsistent dynamics. *Synthese* 193:1–14. doi: 10.1007/s11229-015-0740-2.

Mares, E. D. 2002. A paraconsistent theory of belief revision. *Erkenntnis* 56(2):229–246.

Priest, G. 2001. Paraconsistent belief revision. *Theoria* 67(3):214–228. doi: 10.1111/j.1755-2567.2001.tb00204.x.

Restall, G., and Slaney, J. 1995. Realistic belief revision. In De Glas, M., and Pawlak, Z., eds., *Proceedings of the Second World Conference in the Fundamentals of Artificial Intelligence*, 367–378. Paris: Angkor.

Ribeiro, M. M. 2012. *Belief Revision in Non-Classical Logics*. SpringerBriefs in Computer Science. Springer.

Tamminga, A. M. 2001. *Belief Dynamics: (Epistemo)logical Investigations*. Ph.D. Dissertation, ILLC, University of Amsterdam, The Netherlands.

Testa, R. R.; Coniglio, M. E.; and Ribeiro, M. M. 2017. Agm-like paraconsistent belief change. *Log J IGPL* 25(4).

A logic of default justifications

Stipe Pandžić
Department of Theoretical Philosophy, Faculty of Philosophy &
Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, Faculty of Science and Engineering
University of Groningen, The Netherlands
s.pandzic@rug.nl

Abstract

We define a logic of default justifications that relies on operational semantics. One of the key features that is absent in standard justification logics is the possibility to weigh different epistemic reasons or pieces of evidence that might conflict with one another. To amend this inadequacy, we develop a semantics for "defeaters": conflicting reasons forming a basis to doubt the original conclusion or to believe an opposite statement. Our logic is able to address interactions of normal defaults without relying on priorities among default rules and introduces the possibility of extension revision for normal default theories.

Introduction

Justification logics provide a formal framework to deal with epistemic reasons. The first justification logic was developed as a logic of arithmetic proofs (LP) by Artemov (2001).[1] Possible world semantics for this logic was first proposed by Fitting (2005a; 2005b) in order to align justification logics within the family of epistemic modal logics. A distinctive feature of justification logic is replacing belief and knowledge modal operators that precede propositions ($\Box P$) by proof terms or, in a generalized epistemic context, justification terms and thereby forming justification assertions $t : P$ that read as "t is a reason that justifies P".

Although justification logic introduced the notions of justification and reason into epistemic logic, it does not formally study the ways of *defeat* among reasons. The importance of defeaters is highlighted by paradigmatic examples from classical literature on defeasible reasoning. The variants of the following example are discussed by Chisholm (1966) and Pollock (1987). Suppose you are standing in a room where you see red objects in front of you. This can lead you to infer that a red-looking table in front of you is in fact red. However, the reason that you have for your conclusion is defeasible. For a typical defeat scenario, suppose you learn that the room you are standing in is illuminated with red light. This gives you a reason to doubt your initial reason to conclude that the table is red, though it would not give you a reason to believe that it is not red. However, if you were to learn, instead, that the table has been painted in white, then you would also have a reason to believe a denial of the claim that the table is red.

The example specifies two different ways in which reasons defeat other reasons: the former is known as *undercut* and the latter as *rebuttal*.[2] Learning additional information

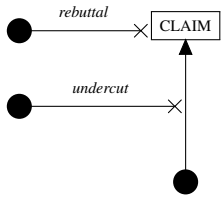


Figure 1: The types of defeat

about the light conditions incurs suspending the applicability of your initial reason to believe that the table is red. In contrast, learning that there is a separate reason to consider that the table is not red will not directly compromise your initial reason itself. The differences between undercutting and rebutting reasons are illustrated in Figure 1.

Only a restricted group of epistemic reasons may be treated as completely immune to defeaters: mathematical proofs. However, they form only a small part of possible reasons to accept a statement and, being a highly-idealized group of reasons, they have rarely been referred to as reasons. Fitting's possible world semantics for justification logics was meant to model not only mathematical and logical truths, but also facts of the world or "inputs from outside the structure" (Fitting 2009, p. 111). Yet the original intent of the first justification logic LP to deal with mathematical proofs, together with the fact that mathematics is cumulative, reflected in its epistemic generalizations. Accordingly, reasons that justify facts of the world were left encapsulated within a framework for non-defeasible mathematical proofs.

Non-mathematical reasons and justifications are commonly held to depend on each other in acquiring their status of "good" reasons and justifications. Still, the questions related to non-ideal reasons have only recently been raised in

[1] The idea of explicit proof terms as a way to find the semantics for provability calculus **S4** dates to 1938 and K. Gödel's lecture published in 1995 (Gödel 1995).

[2] The terminology is originally Pollock's (1987).

the justification logic literature.[3] In the present paper we develop a non-monotonic justification logic with justification terms such that (1) their defeasibility can be tracked from the term structure and (2) other justifications can defeat them by means of an undercut or a rebuttal. Our logic combines techniques from both default logic and justification logic to formalize conflicts of reasons produced in less-than-ideal ways.

Justification logic

The introduction of justifications into modal semantics opened up a possibility to study formal systems for non-defeasible epistemic reasons based on justification logic. These systems include an explicit counterpart to the modal *Truth axiom*: $\Box F \rightarrow F$.[4] Varieties of these systems have been extensively studied and described in e.g. (Kuznets 2000) and (Fitting 2008). Syntactic objects that represent mathematical proofs in LP are more broadly interpreted as epistemic or doxastic reasons by Fitting (2005a; 2005b) and Artemov and Nogina (2005). In order to introduce our system of default reasons, we build upon the existing systems for non-defeasible reasons. In this respect, one can see our strategy as being analogous to the standard default logic approach (Antoniou 1997; Reiter 1980) where agents reason from known or certain information. This section gives preliminaries on one of the logics of non-defeasible reasons.

Since we assume that an agent starts to reason from indefeasible information, we want our underlying logic to represent "factive" or "truth inducing" reasons. However, additional constraints on the system are not necessarily needed to introduce the system of default reasons. Therefore, we do not assume standard axioms and operations that ensure positive or negative introspection. Accordingly, an adequate logical account of factive justifications is the logic $\mathbf{JT}$, a justification logic with the axiom schemes that are explicit analogues of the axiom schemes for modal logic $\mathbf{T}$.[5] After we define the underlying logic, we develop our novel non-monotonic approach to justifications.

Syntax

Syntactically, knowledge operators take the form of justification terms preceding formulas: $t : F$. Given that "t" is a justification term and that "F" is a formula, we write "$t : F$", where t is informally interpreted as a reason or justification for F. We define the set Tm that consists of exactly all justification terms, constructed from variables $x_1, \ldots, x_n, \ldots$ and constants $c_1, \ldots, c_n, \ldots$ by means of operations $\cdot$ and $+$. The grammar of justification terms is given as follows:

$$t ::= x \mid c \mid (t_1 \cdot t_2) \mid (t_1 + t_2)$$

where x is a variable denoting an unspecified justification and c is a proof constant. Proof constant c is atomic within the system. For a justification term t, a set of subterms $Sub(t)$ is defined by induction on the construction of t. Formulas of $\mathbf{JT}$ are defined by the following grammar:

$$F ::= \top \mid P \mid (F_1 \rightarrow F_2) \mid (F_1 \vee F_2) \mid (F_1 \wedge F_2) \mid \neg F \mid t : F$$

where $P \in \mathcal{P}$ and $\mathcal{P}$ is an enumerable set of atomic propositional formulas and $t \in Tm$. The set Fm consists of exactly all formulas.

Axioms and rules of JT

We can now define the logic of non-defeasible reasons $\mathbf{JT}$. The logic $\mathbf{JT}$ is the weakest logic with "truth inducing" justifications containing axiom schemes for two basic operations $\cdot$ and $+$.[6] These are the axioms and rules of $\mathbf{JT}$:

A0 *All the instances of propositional logic tautologies from* Fm

A1 $t : (F \rightarrow G) \rightarrow (u : F \rightarrow (t \cdot u) : G)$ (Application)

A2 $t : F \rightarrow (t + u) : F$; $\quad u : F \rightarrow (t + u) : F$ (Sum)

A3 $t : F \rightarrow F$ (Factivity)

R0 *From* F *and* $F \rightarrow G$ *infer* G (Modus ponens)

R1 *If* F *is an axiom instance of A0-A3 and* $c_n, c_{n-1} \ldots, c_1$ *proof constants, then infer* $c_n : c_{n-1} \ldots c_1 : F$ (Iterated axiom necessitation)

Proof constants are justifications of basic logic truths. In justification logics, basic truths are taken to be justified (at any depth) by virtue of their status within a system and their justifications are not further analyzed. A set of instances of such canonical formulas in justification logic is called *Constant Specification* ($\mathcal{CS}$) set.

Definition 1 (Constant specification)**.** *The Constant Specification set is the set of instances of rule R1.*

$$\mathcal{CS} = \{c_n : c_{n-1} \ldots c_1 : A \mid A \text{ is an axiom instance of A0-A3, } c_n, c_{n-1}, \ldots, c_1 \text{ are proof constants and } n \in \mathbb{N}\}$$

The use of constants in R1 above is unrestricted. In such format, the rule generates a set of formulas where each axiom is justified by any constant at any depth. The set of formulas obtained in this way is called *Total Constant Specification* ($\mathcal{TCS}$). A more appropriate name for the logic above would therefore be $\mathbf{JT_{TCS}}$. It is possible to put restrictions on the use of constants in rule R1 in order to consider a limited class of $\mathcal{CS}$-sets. We restrict the constant specification

[3] The first proposed formalism that includes the idea of evidence elimination specific to a multi-agent setting is by Renne (2012). Baltag, Renne and Smets (2012; 2014) bring together ideas from belief revision and dynamic epistemic logic and offer an account of good and conclusive evidence. Several approaches ((Milnikel 2014), (Kokkinis et al. 2015), (Kokkinis, Ognjanović, and Studer 2016) and (Ognjanović, Savić, and Studer 2017)) start from the idea of merging probabilistic degrees of belief with justification logic, while (Fan and Liau 2015) and (Su, Fan, and Liau 2017) develop a possibilistic justification logic.

[4] In fact, in (Fitting 2008, p. 156) we find three different truth axiom schemes.

[5] Justification logic $\mathbf{JT}$ was first introduced by Brezhnev (2001). Justification logics with equivalent axiom schemes to the logic we define in this section are also defined and investigated in (Kuznets 2000) and (Fitting 2008). In future work, adding the axioms of positive and negative introspection could be considered.

[6] As Fitting (2005b; 2008) shows, we can also technically consider dropping the operator $+$ from our language. In this way we obtain the logic that he calls LP$^-$(T) (Fitting 2008, p. 162).

set $\mathcal{CS}$ following a simple intuition that each axiom instance has its own proof constant.[7]

Restriction 2. *$\mathcal{CS}$ is*

- *Axiomatically appropriate: for each axiom instance A, there is a constant c such that $c : A \in \mathcal{CS}$ and for each formula $c_n : c_{n-1} \dots c_1 : A \in \mathcal{CS}$, such that $n \geq 1$, $c_{n+1} : c_n : c_{n-1} \dots c_1 : A \in \mathcal{CS}$ for some c_{n+1};*
- *Injective: Each proof constant c justifies at most one formula.*

The logic $\mathbf{JT_{CS}}$ is defined by replacing the iterated axom necessitation rule of $\mathbf{JT_{TCS}}$ with the following rule dependent on Restriction 2:

R1* *If F is an axiom instance of A0-A3 and $c_n, c_{n-1} \dots, c_1$ proof constants such that $c_n : c_{n-1} : \dots c_1 : F \in \mathcal{CS}$, then infer $c_n : c_{n-1} : \dots c_1 : F$*

We say that the formula F is $\mathbf{JT_{CS}}$-provable ($\mathbf{JT_{CS}} \vdash F$) if F can be derived using the axioms A0-A3 and rules R0 and R1*.

Semantics

The semantics for $\mathbf{JT_{CS}}$ is an adapted version of the semantics for the logic of proofs ($\mathbf{LP}$) given by Mkrtychev (1997).[8]

Definition 3 ($\mathbf{JT_{CS}}$ model)**.** *We define a function* reason assignment *based on $\mathcal{CS}$ $*(\cdot) : Tm \to 2^{Fm}$, a function mapping each term to a set of formulas from Fm. It satisfies the following conditions:*

1. *If $F \to G \in *(t)$ and $F \in *(u)$, then $G \in *(t \cdot u)$*
2. *$*(t) \cup *(u) \subseteq *(t + u)$*
3. *If $c : F \in \mathcal{CS}$, then $F \in *(c)$*

A truth assignment $v : \mathcal{P} \to \{True, False\}$ is a function assigning truth values to propositional formulas in $\mathcal{P}$. We define the interpretation $\mathcal{I}$ as a pair $(v, *)$. For an interpretation $\mathcal{I}$, $\models$ is a truth relation on the set of formulas of $\mathbf{JT_{CS}}$.

For any formula $F \in Fm$, $\mathcal{I} \models F$ iff

- For any $P \in \mathcal{P}$, $\mathcal{I} \models P$ iff $v(P) = True$
- $\mathcal{I} \models \neg F$ iff $\mathcal{I} \not\models F$
- $\mathcal{I} \models F \to G$ iff $\mathcal{I} \not\models F$ or $\mathcal{I} \models G$
- $\mathcal{I} \models F \vee G$ iff $\mathcal{I} \models F$ or $\mathcal{I} \models G$
- $\mathcal{I} \models F \wedge G$ iff $\mathcal{I} \models F$ and $\mathcal{I} \models G$
- $\mathcal{I} \models t : F$ iff $F \in *(t)$

The interpretation $\mathcal{I}$ is *reflexive*, which means that the truth relation for $\mathcal{I}$ fulfills the following condition:

- For any term t and any formula F, if $F \in *(t)$, then $\mathcal{I} \models F$.

Definition 4 ($\mathbf{JT_{CS}}$ consequence relation)**.** *$\Sigma \models F$ iff for all reflexive interpretations $\mathcal{I}$, if $\mathcal{I} \models B$ for all $B \in \Sigma$, then $\mathcal{I} \models F$.*

Due to Restriction 2, the consequence relation for $\mathbf{JT_{CS}}$ is weaker than the $\mathbf{JT_{TCS}}$ consequence relation.

Definition 5 ($\mathbf{JT_{CS}}$ closure)**.** *JT_{CS} closure is given by $Th^{JT_{CS}}(\Gamma) = \{F | \Gamma \models F\}$, for a set of formulas $\Gamma \subseteq Fm$ and the JT_{CS} consequence relation $\models$ defined above.*

For any closure $Th^{JT_{CS}}(\Gamma)$, it follows that $\mathcal{CS} \subseteq Th^{JT_{CS}}(\Gamma)$.

We can prove that the compactness theorem holds for the $\mathbf{JT_{CS}}$ semantics.[9] Compactness turns out to be a useful result in defining the operational semantics of default reason terms. We first say that a set of formulas Γ is $\mathbf{JT_{CS}}$ *satisfiable* if there is an interpretation $\mathcal{I}$ that meets $\mathcal{CS}$ (via the third condition of Def. 3) for which all the members of Γ are true. A set Γ is $\mathbf{JT_{CS}}$-finitely satisfiable if every finite subset Γ' of Γ is $\mathbf{JT_{CS}}$ satisfiable.

Theorem 6 (Compactness)**.** *A set of formulas is $\mathbf{JT_{CS}}$ satisfiable iff it is $\mathbf{JT_{CS}}$-finitely satisfiable.*

Proof. See the Appendix. □

A logic of default justifications

In this section, we develop a system based on $\mathbf{JT_{CS}}$ in which agents form default justifications reasoning from an incomplete knowledge base. Justification logic $\mathbf{JT_{CS}}$ is capable of representing the construction of a new piece of evidence out of existing ones by application ("$\cdot$") or sum ("+") operation. However, to extend an incomplete $\mathbf{JT_{CS}}$ theory, we need to import reasons that are defeasible. We come up with both a way in which such reasons are imported and a way in which they might get defeated by introducing concepts familiar from defeasible reasoning literature into justification logic.

We start from the above-defined language of the logic $\mathbf{JT_{CS}}$ and develop a new variant of justification logic $\mathbf{JT_{CS}}$ that enables us to formalize the import of reasons outside the structure as well as to formalize *defeaters* or reasons that question the plausibility of other reasons.

Our logical framework of defeasible reasons represents both factive reasons produced via the axioms and rules of $\mathbf{JT_{CS}}$ and plausible reasons based on default assumptions

[7] For example, one such constant specification is defined by Artemov (2018, p. 31): "$c_n : A \in \mathcal{CS}$ iff A is an axiom and n is the Gödel number of A". The choice of $\mathcal{CS}$ is not trivial. If we define an empty $\mathcal{CS}$, that is, $\mathbf{JT}_\emptyset$, we eliminate logical awareness for agents, while defining an infinite $\mathcal{CS}$ imposes logical omniscience. To ensure that standard properties as *Internalization* (Artemov 2001) hold, $\mathcal{CS}$ has to be axiomatically appropriate. Moreover, different restrictions could affect complexity results, as discussed in e.g. (Milnikel 2007).

[8] The condition for justifications of the type '$!t$' are not needed in the $\mathbf{JT_{CS}}$ semantics. Mkrtychev's model can be thought of as a single world justification model. Since the notion of defeasibility introduced in the next section turns on the incompleteness of available reasons, our system eliminates worries about the trivialization of justification assertions that otherwise arise from considering justifications as modalities in a single-world model.

[9] A compactness proof for $\mathbf{LP}$ satisfiability in possible world semantics is given in (Fitting 2005b).

that "usually" or "typically" hold for a restricted context.[10] We follow the standard way (Reiter 1980) of formalizing default reasoning through default theories to extend the logic of factive reasons with defeasible reasons. Building on the syntax of $\mathbf{JT_{CS}}$, we introduce the definition of the *default theory*:

Definition 7 (Default Theory)**.** *A default theory T is defined as a pair (W, D), where the set W is a finite set of $\mathbf{JT_{CS}}$ formulas and D is a countable set of default rules.*

Each default rule is of the following form:

$$\delta = \frac{t : F :: (u \cdot t) : G}{(u \cdot t) : G}.$$

The informal reading of the default δ is: "If t is a reason for F, and it is consistent to assume that $(u \cdot t)$ is a reason for G, then $(u \cdot t)$ is a defeasible reason to believe G". The formula $t : F$ is called the *prerequisite* and $(u \cdot t) : G$ is both the *consistency requirement*[11] and the *consequent* of the default rule δ. We refer to each of the respective formulas as $pre(\delta)$, $req(\delta)$ and $cons(\delta)$. For the set of all consequents from the entire set of defaults D, we use $cons(D)$. The default rule δ introduces a unique reason term u, which means that, for a default theory T, the following holds:

1. For any formula $t : F \in Th^{JT_{CS}}(W)$, $u \neq t$ and
2. For any other default rule $\delta' \in D$ such that $\delta' = \frac{t' : F' :: (u' \cdot t') : G'}{(u' \cdot t') : G'}$, if $F \neq F'$ or $G \neq G'$, then $u \neq u'$.[12]

The reason term u witnesses the defeasiblity of the *prima facie* reason $(u \cdot t)$ for G. Whether a reason actually becomes defeated or not depends on other default-reason formulas from $cons(D)$. Other defaults might question both the plausibility of the default reason u and the plausibility of the proposition G.

A formal way of looking at a default reason of this kind is that $(u \cdot t)$ codifies the default step we apply on the basis of the known reason t. A distinctive feature of such rules is generating justification terms as if it were the case that $cons(\delta)$ was inferred by using an instance of the application axiom: $u : (F \to G) \to (t : F \to (u \cdot t) : G)$. The difference is that an agent cannot ascertain that an available reason justifies applying the conditional $F \to G$ without restrictions. Still, sometimes a conclusion must be drawn without being able to remove all of the uncertainty as to whether the relevant conditional actually applies or not. In such cases, an agent turns to a plausible assumption of a justified "defeasible" conditional $F \to G$ that holds only in the absence of any information to the contrary. While the internal structure of the default reason $(u \cdot t)$ indicates that it is formed on the basis of the formula $u : (F \to G)$, the defeasibility of $(u \cdot t)$ lies in the fact that the formula $u : (F \to G)$ is not a part of the knowledge base.

One can think of our use of the operation "$\cdot$" in default rules as the same operation that is used in the axiom A1, only being applied on an incomplete $\mathbf{JT_{CS}}$ theory. Similarly, we can follow Reiter (1980, p. 82) and Antoniou (1997, p. 21) in thinking of a standard default rule such as $\frac{A:B}{B}$ as merely saying that an implication $A \wedge \neg C \wedge \neg D \cdots \to B$ holds, provided that we can establish that a number of exceptions $C, D, \ldots$ does not hold. However, if the rule application context is defined sufficiently narrowly, the rule is classically represented as an implication $A \to B$. Generalizing on such interpretation of defeasibility, our defaults with justification assertions can be represented as instantiations of the axiom A1 applied in a sufficiently narrow application context.

Analogous to standard default theories, we take the set of facts W to be underspecified with respect to a number of facts that would otherwise be specified for a complete $\mathbf{JT_{CS}}$ interpretation. Besides simple facts, our underlying logic contains justification assertions. To deal with justification assertions, a complete $\mathbf{JT_{CS}}$ interpretation would also further specify whether a reason is acceptable as a justification for some formula. Therefore, except the usual incomplete specification of known propositions, default justification theories are also incomplete with respect to the actual specification of the reason assignment function. For our default theory, this means that, except the valuation v, default rules need to approximate an actual reason-assignment function $*(\cdot)$.

In "guessing" what a true model is, every default rule introduces a reason term whose structure codifies an application operation step from an unknown justified conditional. For example, in rule δ above, we rely on the justified conditional $u : (F \to G)$. Even though this justified conditional is not a part of the rule δ itself, it is the underlying assumption on the basis of which we are able to extend an incomplete knowlede base. Each underlying assumption of this kind can be made explicit by means of a function *default conditional assignment*: $\#(\cdot) : D \to Fm$. The function maps each default rule to a specific justified conditional as follows:

$$\#(\delta_i) = u_n : (F \to G),$$

where $\delta_i \in D$ and $\delta_i = \frac{t_k : F :: (u_n \cdot t_k) : G}{(u_n \cdot t_k) : G}$, for some reason terms t_k and u_n and some formulas F and G.

A set of all such underlying assumptions of default rules is called *Default Specification* ($\mathcal{DS}$) set.

Definition 8 (Default specification)**.** *For a default theory $T = (W, D)$, justified defeasible conditionals are given by the Default Specification set:*

$$\mathcal{DS} = \#[D] = \{u_n : (F \to G) \mid \#(\delta_i) = u_n : (F \to G) \text{ and } \delta_i \in D\}.$$

The use of underlying assumptions from $\mathcal{DS}$ is responsible for the non-monotonic character of default reasons and contrasts our default rules with the standard application operation represented by the axiom A1. The extended meaning of

[10] For a logical account of typicality based on ranked models and preferential reasoning, see Propositional Typicality Logic (PTL) developed by Booth, Meyer, and Varzinczak (2012). In PTL, a typicality operator is added to propositional logic and interpreted in terms of ranked models to formally capture the most typical situations in which a given formula holds.

[11] In order to avoid any misunderstanding, we avoid the name *justification* for the formula $req(\delta)$ since justification logic terms are commonly known as justifications.

[12] Similarly, Artemov (2018, p. 30) introduces "single-conclusion" (or "pointed") justifications to enable handling "justifications as objects rather than as justification assertions".

the application operation via default rules will be referred to as **default application**. Extending the interpretation of the application operation "$\cdot$" can be formally captured by the following definition:

Definition 9 (Default Application)**.** *For a default rule $\delta \in D$, if $u : (F \to G) = \#(\delta)$ and if $t : F = pre(\delta)$, then $(u \cdot t) : G = cons(\delta)$.*

Let us again consider the red-looking-table example from the Introduction to see how *prima facie* reasons and their defeaters are imported through default rules.

Example 10. *Let R be the proposition "the table is red-looking" and let T be the proposition "the table is red". Take t_a and u_a to be some specific individual justifications. The reasoning whereby one accepts the default reason $(u_a \cdot t_a)$ might be described by the following default rule:*

$$\delta_a = \frac{t_a : R :: (u_a \cdot t_a) : T}{(u_a \cdot t_a) : T}.$$

We can informally read the default as follows: "If you have a reason to believe that a table is red looking and it is consistent for you to assume that this gives you a reason supporting the claim that the table is red, then you have a defeasible reason to conclude that the table is red". Suppose you then get to a belief that "the room you are standing in is illuminated with red light", a proposition denoted by L. For some specific justifications t_b and u_b, the following rule gives you an undercutting reason for $(u_a \cdot t_a)$:

$$\delta_b = \frac{t_b : L :: (u_b \cdot t_b) : \neg[u_a : (R \to T)]}{(u_b \cdot t_b) : \neg[u_a : (R \to T)]},$$

where the rule is read as "If you have a reason to believe that the lighting is red and it is consistent for you to assume that this gives you a reason to deny your reason to conclude that the red-looking table is red, then you have a defeasible reason that denies your reason to conclude that the red-looking table is red". The formula $cons(\delta_b)$ denies the basis for the inference that led you to conclude $cons(\delta_a)$, although note that it is not directly inconsistent with it. In the next subsection we define what undercutting defeaters are semantically.

Suppose that instead of learning about the light conditions in the room as in δ_b, you learn that the table has been painted white. This would also prompt a rebutting defeater - a separate reason to believe the contradicting proposition $\neg T$. Let W denote the proposition "the table is painted white" and let t_c and u_c be some specific justifications. We have the following rule:

$$\delta_c = \frac{t_c : W :: (u_c \cdot t_c) : \neg T}{(u_c \cdot t_c) : \neg T}.$$

The rule reads as "If you have a reason to believe that the table has been painted white and it is consistent for you to assume that this gives you a reason supporting the claim that the table is not red, then you have a defeasible reason to conclude that the table is not red". Note that the formula $cons(\delta_c)$ does not directly mention any of the subterms of $(u_a \cdot t_a)$. The defeat among the reasons $(u_a \cdot t_a)$ and $(u_c \cdot t_c)$ comes from the fact that they cannot together consistently extend an incomplete $\mathbf{JT_{CS}}$ theory.

The entire example can be described by the following default theory $T_0 = (W_0, D_0)$, where $W_0 = \{t_a : R, t_b : L, t_c : W\}$ and $D_0 = \{\delta_a, \delta_b, \delta_c\}$.

Each defeater above is itself defeasible and considered to be a *prima facie* reason. The way in which *prima facie* reasons interact is further specified through their role in the operational semantics.

Operational semantics of default justifications

Between the two types of defeaters, the semantics of rebutting justifications is more straightforward since it rests on the known mechanism of multiple extensions used in standard default theories. What requires additional explanation is the semantics of undercutting defeaters. Notice that each formula $\#(\delta)$ has the format of a justified material conditional. This formula is not a part of a default inference δ itself, but the default application described by δ depends on assuming a reason for that conditional and the justification assertion $cons(\delta)$ encodes this assumption in the internal structure of the resulting reason term. This brings to attention the following possibility: a knowledge base may at the same time contain justified formulas of the type $t : F$, $(u \cdot t) : G$ and $v : \neg[u : (F \to G)]$, without the knowledge base being inconsistent. Although the application axiom A1 does not say that $t : F$ and $(u \cdot t) : G$ together entail the formula $u : (F \to G)$, the occurrence of the formulas $t : F$, $(u \cdot t) : G$ and $v : \neg[u : (F \to G)]$ together is not significant in standard justification logic. It only becomes significant with default application.[13]

The extension of the application operation to its defeasible variant opens new possibilities for a semantics of justifications. In particular, it enables reasoning that is not regimented by the standard axioms A1 and A2 of basic justification logic (Artemov 2008, p. 482). For instance, if a set of $\mathbf{JT_{CS}}$ formulas contains both a *prima facie* reason t and its defeater u, then the set containing a conflict of justifications does not support concatenation of reasons by which $t : F \to (t+u) : F$ holds for any two terms t and u. In other words, the possibility of a conflict between reasons eliminates the monotonicity property of justifications assumed in the sum axioms (A2).

The logic of default justifications we develop here relies on the idea of operational semantics for standard default logics presented in (Antoniou 1997). Here is an informal description of the key operational semantics steps. First, default reasons are taken into consideration at face value. After the default reasons have been taken together, we check dependencies among them in order to find out what are the non-defeated reasons. Finally, a rational agent includes in its knowledge base only acceptable pieces of information that are based on those reasons that are ultimately non-defeated.

The basis of operational semantics for a default theory $T = (W, D)$ is the procedure of collecting new, reason-

[13] Notice that a ($\mathbf{JT_{CS}}$-closed) knowledge base that contains the formulas $t : F$ and $(u \cdot t) : G$, also contains the formula $((c \cdot t) \cdot (u \cdot t)) : (F \to G)$, assuming that the constant c justifies the axiom $F \to (G \to (F \to G))$. This is so regardless of whether $u : (F \to G)$ is also in the knowledge base or not.

based information from the available defaults. A *sequence* of default rules $\Pi = (\delta_0, \delta_1, \ldots)$ is a possible order in which a list of default rules without multiple occurrences from D is applied (Π is possibly empty). Applicability of defaults is determined in the following way: for a set of $\mathbf{JT_{CS}}$-closed formulas Γ we say that a default rule $\delta = \frac{t:F::(u\cdot t):G}{(u\cdot t):G}$ is applicable to Γ iff

- $t : F \in \Gamma$ and
- $\neg(u \cdot t) : G \notin \Gamma$.

Reasons are brought together in the set of $\mathbf{JT_{CS}}$ formulas that represents the current evidence base:

Definition 11. $In(\Pi) = Th^{JT_{CS}}(W \cup \{cons(\delta) \mid \delta \textit{ occurs in } \Pi\})$.

The set $In(\Pi)$ collects reason-based information that is yet to be determined as acceptable or unacceptable depending on the acceptability of reasons and counter-reasons for formulas.

We need to further specify sequences of defaults that are significant for a default theory T: default processes. For a sequence Π, the initial segment of the sequence is denoted as $\Pi[k]$, where k stands for the number of elements contained in that segment of the sequence and where k is a minimal number of defaults for the sequence Π. Any segment $\Pi[k]$ is also a sequence. Intuitively, the set of formulas $In(\Pi)$ represents an updated incomplete knowledge base W where the new information is not yet taken to be granted. Using the notions defined above, we can now get clear on what a default process is:

Definition 12 (Process)**.** *A sequence of default rules* Π *is a* process *of a default theory* $T = (W, D)$ *iff every* k *such that* $\delta_k \in \Pi$ *is* applicable *to the set* $In(\Pi[k])$*, where* $\Pi[k] = (\delta_0, \ldots \delta_{k-1})$.

We will use default specification sets that are relativized to default processes:

$$\mathcal{DS}^{\Pi} = \{u_n : (F \to G) \mid \#(\delta_i) = u_n : (F \to G) \text{ and } \delta_i \in \Pi\}.$$

The kind of process that we are focusing on here is called *closed* process and we say that a process Π is closed iff every $\delta \in D$ that is applicable to $In(\Pi)$ is already in Π. For default theories with a finite number of defaults, closure for any process Π is obviously guaranteed by the applicability conditions. However, if a set of defaults is infinite, then this is less-obvious.

Lemma 13 (Infinite Closed Process)**.** *For a theory* $T = (W, D)$ *and infinitely many* k*'s, an infinite process* Π *is closed iff for every default rule* δ_k applicable *to the set* $In(\Pi[k])$*,* $\delta_k \in \Pi$.

Proof. From the compactness of $\mathbf{JT_{CS}}$ semantics we have that if a set $In(\Pi[k]) \cup \{req(\delta)\}$ is satisfiable for all the finite k's, it is also satisfiable for infinitely many k's. Therefore the applicability conditions for a rule δ are equivalent to the finite case. □

Besides the standard process of collecting new information, we need to explain the way in which an agent decides on the acceptability of reasons. We have already introduced the extended meaning of the application operation for a default theory T. Now we show how default application is essential to the operational semantics of default reasons. Ideally, an agent has all the factive reasons valid under some interpretation $\mathcal{I}$. In contrast, in reasoning from an incomplete knowledge base W, a closure $Th^{JT_{CS}}(W)$ is typically underspecified as to whether a reason t is acceptable for a formula F. In such context, reasoning starts from defeasible justification assertions in $\mathcal{DS}$ as the only available resource to approximate a reason assignment function that actually holds.

Notice that $\mathcal{DS}$ can be an inconsistent set of $\mathbf{JT_{CS}}$ formulas and that an agent needs to find out which reasons prevail in a conflicting set of reasons. One way in which reasons may conflict with each other is captured by the definition of undercut:

Definition 14 (Undercut)**.** *A reason* u *undercuts reason* t *being a reason for a formula* F *in a set of* $\mathbf{JT_{CS}}$*-closed formulas* $\Gamma \subseteq In(\Pi[k])$ *iff* $\bigvee_{(v)\in Sub(t)} u : \neg[v : (G \to H)] \in \Gamma$ *and* $v : (G \to H) \in \mathcal{DS}^{\Pi}$.

For a set Γ such that $Th^{JT_{CS}}(\Gamma)$ contains some reason u that undercuts t we say that Γ undercuts t. We can think of Γ as a set of reasons against which we test the reason t being reason for the formula F. This is further elaborated in the semantics of acceptability of reasons. We now define conflict-free sets of formulas:[14]

Definition 15 (Conflict-free sets)**.** *A set of* $\mathbf{JT_{CS}}$*-closed formulas* Γ *is conflict-free iff* Γ *does not contain both a formula* $t : F$ *with an undercut reason* t *and its undercutter* $u : G$.

As stated before, the set W contains certain information and this means that any information from W is always acceptable regardless of what has been collected later on. Therefore, any set of formulas Γ that extends the initial information contains W. To decide whether a consequent of a default δ is acceptable, an agent looks at those sets of reasons that can be defended against all the available counter-reasons. According to that, an agent looks at finding a defensible set of justified formulas among all certain information taken together with the consequents of the applicable defaults rules. Therefore, for a default theory $T = (W, D)$, an agent always considers potential extension sets of $\mathbf{JT_{CS}}$ formulas that meet the following conditions:

1. $W \subseteq \Gamma$ and
2. $\Gamma \subseteq \{W \cup cons(\delta) \mid \delta \text{ occurs in } \Pi_i\}$,

where Π_i is a closed process of T. For any potentially acceptable set Γ we define the notion of acceptability of a justified formula $t : F$:

[14] In characterizing sets of $\mathbf{JT_{CS}}$ formulas we use the terminology of Dung's (1995) abstract argumentation frameworks whenever possible. Abstract argumentation frameworks treat conflicts between arguments and they naturally overlap with our idea of conflicting reasons in many ways.

Definition 16 (Acceptability)**.** *For a default theory $T = (W, D)$, a formula $t : F \in cons(\Pi)$ is acceptable w.r.t. a set of $\mathbf{JT_{CS}}$ formulas Γ iff for each undercutting reason u for t being a reason for F such that $u : G \in In(\Pi)$, $Th^{JT_{CS}}(\Gamma)$ undercuts u being a reason for G.*

Informally, an agent has yet to test any potential extension against all the other available reasons before it can be considered as an admissible extension of the knowledge base.

Definition 17 (Admissible Extension)**.** *A potential extension set of $\mathbf{JT_{CS}}$ formulas Γ is an admissible extension of a default theory $T = (W, D)$ iff $Th^{JT_{CS}}(\Gamma)$ is conflict-free and if each formula $t : F \in \Gamma$ is acceptable w.r.t. Γ.*

After considering all the available reasons, an agent accepts only those defeasible statements that can be defended against all the available reasons against these statements.

The two latter definitions introduce the idea of "external stability" of knowledge bases (Dung 1995, p. 323) into default logic by taking into account all the reasons that question the plausibility of other reasons. In addition to that, our operational semantics prompts an implicit revision procedure. Any new default rule that is applicable to the set of formulas $In(\Pi[k])$ potentially makes changes to what an agent considered to be acceptable relying on the set of formulas $In(\Pi[k-1])$. Before we show this on the formalized example from the beginning of this section, we introduce the idea of default extension for a default theory T. Extension is the fundamental concept in defining logical consequence in standard default theories. We think of preferred extensions as maximal plausible world views based on the acceptability of reasons:

Definition 18 (Preferred Extension)**.** *For a default theory $T = (W, D)$, an admissible extension set of $\mathbf{JT_{CS}}$ formulas Γ, $Th^{JT_{CS}}(\Gamma)$ is a preferred extension of a default theory T iff for any other admissible extension Γ', $\Gamma \not\subset \Gamma'$.*

In other words, preferred extensions are maximal admissible extensions with respect to set inclusion. The existence of preferred extensions is universally defined for default theories. To ensure that this result also holds for the case of an infinite number of default rules and infinite closed processes, we make use of Zorn's lemma and restate it as follows:

Lemma 19 (Zorn)**.** *For every partially ordered set A, if every chain of (totally ordered subset of) B has an upper bound, then A has a maximal element.*

Theorem 20 (Existence of Preferred Extension)**.** *Every default theory $T = (W, D)$ has at least one preferred extension.*

Proof. If W is inconsistent, then for any default δ, negation of the consistency requirement $req(\delta)$ is contained in $Th^{JT_{CS}}(W)$ and the only closed process Π is the empty sequence. Therefore, the only potential and admissible extension is W itself and T has a unique preferred extension $Th^{JT_{CS}}(W)$ containing all the formulas of $\mathbf{JT_{CS}}$.

Assume that W is consistent. In general, if there is a finite number of default rules in D, any closed process Π of T is also finite. Admissible extensions obtained from closed processes form a complete partial order with respect to $\subseteq$. Since there are only finitely many admissible sets, any admissible set Γ has a maximum Γ' within a totally ordered subset of a set of all admissible sets. Therefore, $\Gamma \subseteq \Gamma'$ and $Th^{JT_{CS}}(\Gamma')$ is a preferred extension of T.

For the case where D is infinite and closed processes $\Pi_1, \Pi_2, \ldots$ are infinite, there is again a complete partial order formed from a set of all admissible sets. The argument for finite processes does not account for the case where Γ', the union of admissible sets $\Gamma_1, \Gamma_2, \ldots$, could be contained in some Γ'' for an ever increasing sequence $\Gamma_1, \Gamma_2, \ldots$. We first state that Γ', the union of an ever increasing sequence of admissible sets $\Gamma_1, \Gamma_2, \ldots$, is also an admissible set. To ensure this, we turn to its subsets. That is, if Γ' was not admissible, then some of its subsets Γ_n for $n \geq 1$ would not be conflict-free or would contain a formula that is not acceptable, but this contradicts the assumption that Γ_n is admissible. Now, for the set of all admissible sets ordered by $\subseteq$, any chain (totally ordered subset) has an upper bound, that is, the union of its members $\Gamma' = \bigcup_{n=1}^{\infty} \Gamma_n$. According to Lemma 19, there exists a maximal element and, therefore a preferred extension of T.

□

The semantics of defeasible reasons enables us to define additional types of extensions that are not necessarily based on the admissibility of reasons. One of them is stable extension familiar from formal argumentation theory:

Definition 21 (Stable Extension)**.** *For a default theory $T = (W, D)$ and its closed processes Π and Π', a stable extension is a $\mathbf{JT_{CS}}$ closure of a potential extension $\Gamma \subset In(\Pi)$ such that* **(1)** *$Th^{JT_{CS}}(\Gamma)$ undercuts all the formulas $t : F \in In(\Pi)$ outside $Th^{JT_{CS}}(\Gamma)$ and* **(2)** *for any formula $u : G \in \Gamma'$ such that $\Gamma' \subset In(\Pi')$ and $u : G \notin In(\Pi)$, it holds that $\Gamma \cup \{u : G\}$ is $\mathbf{JT_{CS}}$ inconsistent.*

The intuition behind the definition is that every reason left outside the accepted set of reasons is attacked. For our logic, this means that for every justification assertion outside of an extension, the extension undercuts one of its subterms and/or it contains a justification assertion inconsistent with it. We can check that in the red-looking-table example, stable and prefer extension coincide. Formally, theory T_0 has a unique stable and preferred extension $Th^{JT_{CS}}(W_0 \cup \{cons(\delta_b), cons(\delta_c)\})$. Moreover, note that the process (δ_a, δ_b) includes a revision of its respective admissible extension.

Stable extensions are not universally defined for any default theory T. Consider the following theory $T_1 = (W_1, D_1)$, where $W_1 = \{t : F\}$ and D_1 contains the default rules

$$\delta_1 = \frac{t : F :: (u \cdot t) : G}{(u \cdot t) : G} \text{ and}$$

$$\delta_2 = \frac{(u \cdot t) : G :: (v \cdot (u \cdot t)) : \neg[u : (F \rightarrow G)]}{(v \cdot (u \cdot t)) : \neg[u : (F \rightarrow G)]}.$$

While T_1 has a preferred extension $Th^{JT_{CS}}(W)$, it has no stable extension. This result conforms to similar results about preferred and stable semantics in abstract argumentation frameworks. In fact, T_1 is a justification logic formal-

ization of the concept of self-defeat, which is notorious in argumentation framworks.

In addition, we can easily add other significant notions of extensions, analogous to those in (Dung 1995). In particular, we can define variants of Dung's (1995, p. 329) *complete* and *grounded* extension. Different extensions definitions will enable us to give different corresponding characterizations of logical consequence. This will lead to proofs of additional theorems and fully establish the role of justification logic within the study of non-monotonic reasoning.

Related and future work

The above suggested connections between default justification logic and abstract argumentation frameworks are currently being investigated. Standard justification logics are known for their connection to modal logics. Artemov (2001) provided a proof of the *Realization Theorem* that connects the logic of arithmetic proofs LP with the modal logic S4. The result has been followed up by similar theorems for many other modal logics with known "explicit" justification counterparts.[15] As it stands now, default justification logic can be considered to provide explicit justification logic counterparts to (a subclass of) abstract argumentation frameworks. A proof of this conjecture is a part of the future work.

Further developments are possible starting from the basic logic of default justifications. On the technical side of our logic, we used only the expressiveness of normal default rules and we still need to investigate how to add non-normal default rules. In the general context of default logics, our logic introduces some new technical properties for normal default theories that are still to be thoroughly described. Among them are revision of extensions and interaction of different defaults that does not rely on their preference orderings, as commonly done in default logic (Delgrande and Schaub 2000). An extensive account of default reasons that makes use of preference orderings on defaults is developed by Horty (2012). Horty's logic is based on a propositional language and develops from a different notion of reasons, which makes it incomparable to our logic where reasons are explicitly featured in the language itself.

Our work provides a complementary addition to the study of less-than-ideal reasons in justification logic. Among related approaches, the logic of conditional probabilities developed by Ognjanović, Savić, and Studer (2017) introduces a way to model non-monotonic reasoning with justification assertions. Their proposal is based on defining operators for approximate probabilities of a justified formula given some condition formula. Using conditional probabilities, the logic models certain aspects of defeasibile inferences with justification terms. Yet the system can neither encode the defeasibility of justification terms in their internal structure nor model defeat among reasons, to mention only some differences from our initial desiderata.

Baltag, Renne, and Smets (2012) define a justification logic in which an agent may hold a justified belief that can be compromised in the face of newly received information. The logic builds on the ideas from belief revision and dynamic epistemic logic to model examples where epistemic actions cause changes to an agent's evidence. Concerning the possibility of modelling defeaters, the logic offers two dynamic operations that change the availability of evidence in a model, namely "updates" and "upgrades" (Baltag, Renne, and Smets 2012, p. 183). Evidence obtained by updates counts as "hard" or infallible, while upgrades bring about "soft" or fallible evidence. With the use of these actions, epistemic models can represent justified beliefs being defeated, for example, by means of an epistemic action of update with hard evidence. In this way, however, the mechanism by which reasons may conflict with one another is simply being "outsourced" to an extra-logical notion of fallibility and, therefore, the logic does not directly address the ways of defeat that we formalize in this paper.

Several interesting paths could be followed in connecting the logic of default justifications with formal argumentation frameworks. Among frameworks with abstract arguments, the AFRA framework (Baroni et al. 2011) with recursive attacks offers a possibility of representing attacks to attacks. This conceptual advance can be useful in connecting default reasons to abstract arguments. Our logic could be seen as closely related to the frameworks with structured arguments, which is why connections with systems such as ASPIC+ (Prakken 2010), DeLP (García and Simari 2004), SG (Hecham, Bisquert, and Croitoru 2018) and the logic-based argumentation framework by Besnard and Hunter (2001) are still to be explored. Since each of these frameworks elaborates on the notion of defeat, a thorough comparison to our logic would shed light on their formal connections. A different logic-based perspective on argumentation frameworks is given by Caminada and Gabbay (2009) and Grossi (2010). Both papers start from the idea of studying attack graphs and formalizing notions of extensions from abstract argumentation theory using modal logic, with the former approach being proof-theoretical and the latter model-theoretical. A further interesting research venue in the field of argumentation theory is the one about the logical interpretation of *prima facie* justified assumptions in (Verheij 2003). The DefLog system which is developed there is closely related to ours in motivation, but it develops from a perspective of a sentence-based theory of defeasible reasoning instead of a rule-based or argument-based approach.

Ever since the concept of justification entered into epistemic logics, there has been a tendency to model mainstream epistemology examples, proposed by e.g. Russell, Dretske and Gettier, with the use of justification logic (Artemov 2008; 2018). With the introduction of default justifications, however, we can expect a more full-blooded integration of the formal theory of justification with the study of knowledge in philosophy, since paradigmatic examples include both incomplete specification of reasons and defeated reasons. Potential benefits of a non-monotonic system of justifications in this context were anticipated by Artemov in (2008, p. 482) where he states that "to develop a theory of non-monotonic justifications which prompt belief revision" stands as an "intriguing challenge". One of many interesting topics from epistemology that could be investigated with default-justifications theory is how does accrual of justifica-

[15] See (Fitting 2016) for a good overview of realization theorems.

tion affect the degree of justification.[16]

Appendix

Proof of Theorem 6. The claim from left to right is obvious. For the other direction, take $\mathcal{CS}$ to be some specific axiomatically appropriate and injective constant specification. We first show that if a set Γ is $\mathbf{JT_{CS}}$-finitely satisfiable, then for all formulas $F \in Fm$, it holds that $\Gamma \cup \{F\}$ or $\Gamma \cup \{\neg F\}$ is $\mathbf{JT_{CS}}$-finitely satisfiable. Suppose that Γ is $\mathbf{JT_{CS}}$-finitely satisfiable and that $\Gamma \cup \{F\}$ and $\Gamma \cup \{\neg F\}$ are both not $\mathbf{JT_{CS}}$-finitely satisfiable. Then there would be finite subsets Γ' and Γ'' of Γ such that $\Gamma' \cup \{F\}$ and $\Gamma'' \cup \{\neg F\}$ are not $\mathbf{JT_{CS}}$ satisfiable. Since for no interpretation $\mathcal{I}$ it holds that $\mathcal{I} \models \{F, \neg F\}$, $\Gamma' \cup \{F, \neg F\}$ is never $\mathbf{JT_{CS}}$ satisfiable. But since for any possible interpretation $\mathcal{I}$ one of the formulas F or $\neg F$ holds, this means that $\mathcal{I} \models \Gamma' \subseteq \mathcal{I} \models \neg F$. In a similar way we get that $\mathcal{I} \models \Gamma'' \subseteq \mathcal{I} \models F$. Therefore, we have that $\mathcal{I} \models \Gamma' \cap \mathcal{I} \models \Gamma'' = \emptyset$ and, thus, $\Gamma' \cup \Gamma''$ is not $\mathbf{JT_{CS}}$-satisfiable. But $\Gamma' \cup \Gamma''$ is a finite subset of Γ and this contradicts the assumption that Γ is $\mathbf{JT_{CS}}$-finitely satisfiable.

The next step is proving a $\mathbf{JT_{CS}}$ variant of the Lindenbaum lemma. Using the above-proven statement that for any $\mathbf{JT_{CS}}$-finitely satisfiable set of formulas Γ and any formula F, $\Gamma \cup \{F\}$ or $\Gamma \cup \{\neg F\}$ is $\mathbf{JT_{CS}}$-finitely satisfiable together with the fact that $\Gamma \cup \{F, \neg F\}$ is never $\mathbf{JT_{CS}}$-finitely satisfiable, we can construct maximally $\mathbf{JT_{CS}}$-finitely satisfiable sets. Let $F_1, F_2, F_3, \ldots$ be an enumeration of $F \in Fm$. For a $\mathbf{JT}_{CS}$-finitely satisfiable set Γ and for all $i \in \mathbb{N}$ define an increasing sequence of sets of formulas as follows:

$\Gamma_0 = \Gamma$
$\Gamma_{i+1} = \Gamma_i \cup \{F_i\}$ if $\Gamma_i \cup \{F_1\}$ is $\mathbf{JT}_{CS}$-finitely satisfiable, otherwise $\Gamma_{i+1} = \Gamma_i \cup \{\neg F_i\}$
$\Gamma' = \bigcup_{i=0}^{\infty} \Gamma_i$

We can prove that Γ' is $\mathbf{JT}_{CS}$-finitely satisfiable by induction. The base case $\Gamma_0 = \Gamma$ holds by assumption. Then we claim that for all $i \in \mathbb{N}$, Γ_i is $\mathbf{JT_{CS}}$-finitely satisfiable. For some $n \in \mathbb{N}$, take Γ_n to be $\mathbf{JT_{CS}}$-finitely satisfiable. Then either $\Gamma \cup \{F_n\}$ or $\Gamma \cup \{\neg F_n\}$ is $\mathbf{JT_{CS}}$-finitely satisfiable and, therefore, Γ_{n+1} is also $\mathbf{JT_{CS}}$-finitely satisfiable.

From the construction of the increasing sequence, we have that for any finite set $\Gamma_k \subseteq \Gamma'$ there is a $\mathbf{JT_{CS}}$-finitely satisfiable finite set $\Gamma_{k+1} \subseteq \Gamma'$ such that $\Gamma_k \subseteq \Gamma_{k+1}$ and, therefore, Γ_k is $\mathbf{JT_{CS}}$-satisfiable. Since any finite subset of Γ' is $\mathbf{JT_{CS}}$ satisfiable, Γ' is $\mathbf{JT_{CS}}$-finitely satisfiable. The set Γ' is maximal according to the enumeration of the set of formulas Fm and contains exactly one of F_i or $\neg F_i$ for all $i \in \mathbb{N}$.

Now we define a valuation v such that $v(P) = True$ iff $P \in \Gamma'$ and the reason assignment $*(t) = \{F \mid t : F \in \Gamma'\}$. We only need to check the conditions on the reason assignment function. First, we show that $*(\cdot)$ satisfies the application condition. Since the formula $t : (F \to G) \to (u : F \to (t \cdot u) : G)$ is $\mathbf{JT_{CS}}$ valid, it is contained in Γ'. If $F \to G \in *(t)$ and $F \in *(u)$, then $\{t : (F \to G), u : F\} \in \Gamma'$. Since Γ is closed under *Modus ponens*, we have that $(t \cdot u) : G \in \Gamma'$ and, therefore, $G \in *(t \cdot u)$. Similarly, since the formulas $t : F \to (t + u) : F$ and $u : F \to (t + u) : F$ are both in Γ' we can easily check that the sum condition holds for $*(\cdot)$.

Finally, we have defined an interpretation $\mathcal{I} = (*, v)$ that meets $\mathcal{CS}$ and we need to prove that truth in this interpretation is equivalent to inclusion in Γ':

$\mathcal{I} \models F$ iff $F \in \Gamma'$

The proof is by induction on the structure of F. For the base case, suppose F is an atomic formula P: $\mathcal{I} \models P$ iff $v(P) = True$ iff $P \in \Gamma'$.

For the inductive step, suppose that if the result holds for F and G, then it also holds for $\neg F$, $F \wedge G$, $F \vee G$, $F \to G$ and $t : F$. For the negation case: $\mathcal{I} \models \neg F$ iff $\mathcal{I} \not\models F$. By the inductive hypothesis, $\mathcal{I} \not\models F$ iff $F \notin \Gamma'$. By the maximality of Γ', we have that $F \notin \Gamma'$ iff $\neg F \in \Gamma'$.

For the conjunction case: $\mathcal{I} \models F \wedge G$ iff $\mathcal{I} \models F$ and $\mathcal{I} \models G$. By the inductive hypothesis, $\mathcal{I} \models F$ and $\mathcal{I} \models G$ iff $F \in \Gamma'$ and $G \in \Gamma'$ iff $F \wedge G \in \Gamma'$. Since other connectives are definable in terms of $\neg$ and $\wedge$, we skip the remaining cases.

Finally for the justified formula case: $\mathcal{I} \models t : F$ iff $F \in *(t)$. By the definition of $*(\cdot)$, it holds that $F \in *(t)$ iff $t : F \in \Gamma'$.

Therefore, for any $\mathbf{JT_{CS}}$-finitely satisfiable set Γ there is an interpretation $\mathcal{I}$ based on a maximal $\mathbf{JT_{CS}}$-finitely satisfiable extension Γ' of Γ such that $\mathcal{I} \models \Gamma$. □

[16] The question is prominent in Pollock's work (Pollock 2001).

References

Antoniou, G. 1997. *Nonmonotonic Reasoning*. Cambridge, MA: MIT Press.

Artemov, S. N., and Nogina, E. 2005. Introducing justification into epistemic logic. *Journal of Logic and Computation* 15(6):1059–1073.

Artemov, S. N. 2001. Explicit provability and constructive semantics. *Bulletin of Symbolic logic* 1–36.

Artemov, S. N. 2008. The logic of justification. *The Review of Symbolic Logic* 1(4):477–513.

Artemov, S. N. 2018. Justification awareness models. In Artemov, S. N., and Nerode, A., eds., *International Symposium on Logical Foundations of Computer Science*, volume 10703 of LNCS, 22–36. Springer.

Baltag, A.; Renne, B.; and Smets, S. 2012. The logic of justified belief change, soft evidence and defeasible knowledge. In Ong, L., and de Queiroz, R., eds., *International Workshop on Logic, Language, Information, and Computation*, 168–190. Springer.

Baltag, A.; Renne, B.; and Smets, S. 2014. The logic of justified belief, explicit knowledge, and conclusive evidence. *Annals of Pure and Applied Logic* 165(1):49–81.

Baroni, P.; Cerutti, F.; Giacomin, M.; and Guida, G. 2011. Afra: Argumentation framework with recursive attacks. *International Journal of Approximate Reasoning* 52(1):19–37.

Besnard, P., and Hunter, A. 2001. A logic-based theory of deductive arguments. *Artificial Intelligence* 128(1-2):203–235.

Booth, R.; Meyer, T.; and Varzinczak, I. 2012. PTL: A propositional typicality logic. In del Cerro, L. F.; Herzig, A.; and Mengin, J., eds., *Logics in Artificial Intelligence: Proceedings of the 13th European conference on Logics in Artificial Intelligence*, volume 7519 of LNCS, 107–119. Springer-Verlag.

Brezhnev, V. 2001. On the logic of proofs. In Striegnitz, K., ed., *Proceedings of the Sixth ESSLLI Student Session, Helsinki*, 35–46.

Caminada, M. W., and Gabbay, D. M. 2009. A logical account of formal argumentation. *Studia Logica* 93(2-3):109.

Chisholm, R. M. 1966. *Theory of Knowledge*. Englewood Cliffs, NJ: Prentice-Hall.

Delgrande, J. P., and Schaub, T. 2000. Expressing preferences in default logic. *Artificial Intelligence* 123(1-2):41–87.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence* 77(2):321–357.

Fan, T.-F., and Liau, C.-J. 2015. A logic for reasoning about justified uncertain beliefs. In Yang, Q., and Wooldridge, M., eds., *Proceedings of the IJCAI 2015*, 2948–2954. AAAI Press.

Fitting, M. 2005a. A logic of explicit knowledge. In Běhounek, L., and Bílková, M., eds., *Logica Yearbook 2004*. Prague: Filosofia. 11–22.

Fitting, M. 2005b. The logic of proofs, semantically. *Annals of Pure and Applied Logic* 132(1):1–25.

Fitting, M. 2008. Justification logics, logics of knowledge, and conservativity. *Annals of Mathematics and Artificial Intelligence* 53(1-4):153–167.

Fitting, M. 2009. Reasoning with justifications. In *Towards Mathematical Philosophy*. Springer. 107–123.

Fitting, M. 2016. Modal logics, justification logics, and realization. *Annals of Pure and Applied Logic* 167(8):615–648.

García, A. J., and Simari, G. R. 2004. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming* 4(1+ 2):95.

Gödel, K. 1995. Vortrag bei Zilsel/Lecture at Zilsels (1938a). In *Kurt Gödel: Collected Works: Volume III: Unpublished Essays and Lectures*, volume 3. Oxford University Press. 87–114.

Grossi, D. 2010. Argumentation in the view of modal logic. In McBurney, P.; Rahwan, I.; and Parsons, S., eds., *7th International Workshop on Argumentation in Multi-Agent Systems, ArgMAS 2010*, volume 6614 of LNCS, 190–208. Springer.

Hecham, A.; Bisquert, P.; and Croitoru, M. 2018. On a flexible representation for defeasible reasoning variants. In Dastani, M.; Sukthankar, G.; André, E.; and Koenig, S., eds., *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018*, 1123–1131. International Foundation for Autonomous Agents and Multiagent Systems.

Horty, J. F. 2012. *Reasons as Defaults*. Oxford University Press.

Kokkinis, I.; Maksimović, P.; Ognjanović, Z.; and Studer, T. 2015. First steps towards probabilistic justification logic. *Logic Journal of the IGPL* 23(4):662–687.

Kokkinis, I.; Ognjanović, Z.; and Studer, T. 2016. Probabilistic justification logic. In Artemov, S. N., and Nerode, A., eds., *International Symposium on Logical Foundations of Computer Science*, volume 9537 of LNCS, 174–186. Springer.

Kuznets, R. 2000. On the complexity of explicit modal logics. In Clote, P. G., and Schwichtenberg, H., eds., *Computer Science Logic: 14th International Workshop, CSL 2000*, volume 1862 of LNCS, 371–383. Springer-Verlag.

Milnikel, R. S. 2007. Derivability in certain subsystems of the logic of proofs is Π_2^p-complete. *Annals of Pure and Applied Logic* 145(3):223–239.

Milnikel, R. S. 2014. The logic of uncertain justifications. *Annals of Pure and Applied Logic* 165(1):305–315.

Mkrtychev, A. 1997. Models for the logic of proofs. In Adian, S., and Nerode, A., eds., *Logical Foundations of Computer Science, 4th International Symposium, LFCS '97*, volume 1234 of LNCS, 266–275. Springer-Verlag.

Ognjanović, Z.; Savić, N.; and Studer, T. 2017. Justification logic with approximate conditional probabilities. In Baltag, A.; Seligman, J.; and Yamada, T., eds., *Logic, Rationality and Interaction, 6th International Workshop, LORI 2017*, volume 10455 of LNCS, 681–686. Springer.

Pollock, J. L. 1987. Defeasible reasoning. *Cognitive Science* 11(4):481–518.

Pollock, J. L. 2001. Defeasible reasoning with variable degrees of justification. *Artificial intelligence* 133(1-2):233–282.

Prakken, H. 2010. An abstract framework for argumentation with structured arguments. *Argument and Computation* 1(2):93–124.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13(1-2):81–132.

Renne, B. 2012. Multi-agent justification logic: Communication and evidence elimination. *Synthese* 185(1):43–82.

Su, C.-P.; Fan, T.-F.; and Liau, C.-J. 2017. Possibilistic justification logic: Reasoning about justified uncertain beliefs. *ACM Transactions on Computational Logic (TOCL)* 18(2):15.

Verheij, B. 2003. DefLog: On the logical interpretation of prima facie justified assumptions. *Journal of Logic and Computation* 13(3):319–346.

Zorn, M. 1935. A remark on method in transfinite algebra. *Bulletin of the American Mathematical Society* 41(10):667–670.

Support Trees For Answer Set Programs

Richard Watson
Department of Computer Science
Texas Tech University
Engineering Center, Room 316
902 Boston Ave.
Lubbock, TX. 79409
richard.watson@ttu.edu

Abstract

In 1994, Ben-Eliyahu and Dechter introduced the idea of proofs of Answer Sets of Extended Disjunctive Logic Programs. Unfortunately, as they stated, the method only works for programs that are head-cycle free. In this work, we present Support Trees for Answer Set programs. We formally define such trees. We then prove a one-to-one correspondence between certain paths of the Support Tree of a program and the program's Answer Sets. These paths are similar to the proofs in the aforementioned work; however, they allow for programs with head-cycles. In addition to the theoretical results, we compare this work to other related works and discuss several potential applications of this work.

1 Introduction

Since the introduction of the Stable Model Semantics for logic programs (Gelfond and Lifschitz 1988) and the subsequent extension to the Answer Set Semantics for extended disjunctive logic programs (Gelfond and Lifschitz 1991), there have been great advances in the field, from the constant evolution towards better, more advanced solvers to an increase in the number and scope of applications.

One significant early work was the paper, *Propositional Semantics for Disjunctive Logic Programs* (Ben-Eliyahu and Dechter 1994). The authors of that paper studied properties of head-cycle-free extended disjunctive logic programs (HEDLPs). Such programs are those in which the dependency graph for the program does not contain directed cycles through two or more literals in the head of the same rule. The oft most used result of that paper is that, in HEDLPs, the disjunctive rules can be replaced by non-disjunctive rules to obtain an equivalent non-disjunctive program. Another interesting idea presented in the paper, however, is that of a *proof* of a literal within an answer set. Such proofs can be seen as an explanation of at least one reason why a certain literal is in the answer set. Furthermore, all literals in an answer set must have a proof. While interesting, their method is only applicable to HEDLPs.

In this work, we present *support trees* for answer set programs. As the name suggests, such trees show the support for the rules along each path of the tree. While our research began as an attempt to provide proofs for literals in programs with head-cycles, the resultant support trees also provide a tree-based characterization of the semantics for answer set programs. In addition, there are applications of the theory to areas such as answer set solver development and query answering routines.

We begin with an overview of answer sets in Section 2. In Section 3, we give the formal definitions leading up to and including that of support trees. Examples of key concepts are also given. The definitions and proofs needed to show the correlation between paths of a support tree for a program and the answer sets of the program are given in section 4. Section 5 provides a discussion of ways of generalizing the construction of support trees in order to provide more flexibility in their construction. In Section 6, related works, including that of (Ben-Eliyahu and Dechter 1994), are discussed. Areas of application of support trees and future work are discussed in Section 7. Finally, there are some brief acknowledgements in section 8.

2 Answer Sets

Here we present a brief overview of the syntax and semantics of answer set programming as well as some other standard definitions. The syntax and semantics will be given through a series of definitions. Due to space, no examples will be given in this section. For a more complete overview of answer sets see (Gelfond and Kahl 2014).

Definition 1 (Term)**.** A *term* is defined as follows:

1. Any object constant or variable is a term.
2. If f is a function symbol of arity n and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

Definition 2 (Atom)**.** An *atom* (also known as an atomic statement) has the form

$$p(t_1, \ldots, t_n)$$

where p is a predicate symbol of arity n and $t_1, \ldots, t_n$ are terms.

Definition 3 (Literal and Extended Literal)**.** A *literal* is an atom, A, or its negation, $\neg A$. In addition, $\top$ and $\bot$ will denote distinguished literals with valuation *true* and *false* respectively. If l is a literal, then *not* l is an extended literal. For any literal l, $\overline{l} = \neg l$ if l is an atom and $\overline{l} = A$ if $l = \neg A$.

Definition 4 (Consistent)**.** A set of literals is *consistent* if it does not contain either $\bot$ or both A and $\neg A$ for some atom A.

Definition 5 (Rule, Program)**.** A *rule* of answer set programming is a statement of the form

$$l_1 \textit{ or } \ldots \textit{ or } l_k \leftarrow l_{k+1}, \ldots, l_m, \textit{not } l_{m+1}, \ldots, \textit{not } l_n.$$

where each l_i is a literal. An *answer set program* is a collection of rules.

The intuitive reading of the rule is "if an agent believes literals l_{k+1} through l_m and has no reason to believe literals l_{m+1} through l_n then the agent must believe one of l_1 through l_k. A term, rule, or program that does not contain variables is said to be *ground*. A rule that contains variables is considered to be a shorthand for the set of rules obtained by replacing the variables in the rule by each possible combination of ground terms. For the rest of the paper, unless otherwise indicated, all literals, rules, and programs will be assumed to be ground.

Definition 6 ($head(r), body^+(r), body^-(r)$)**.** If r is an ASP rule:

- $head(r)$ is the set of literals $\{l_1, \ldots, l_k\}$.
- $body^+(r)$ is the set of literals $\{l_{k+1}, \ldots, l_m\}$.
- $body^-(r)$ is the set of literals $\{l_{m+1}, \ldots, l_n\}$.

Definition 7 (Satisfies)**.** Given a set of literals, A, and a rule, r, *A satisfies*:

- $head(r)$ if $head(r) \cap A \neq \emptyset$.
- $body^+(r)$ if $body^+(r) \subseteq A$.
- $body^-(r)$ if $body^-(r) \cap A = \emptyset$.
- r if whenever A satisfies $body^+(r)$ and $body^-(r)$ it also satisfies $head(r)$.

The semantics of an answer set program are based on a rationality principle that "the agent should not believe anything they are not forced to believe". Furthermore, the agent's beliefs must be consistent and must satisfy all the rules of the program. The formal semantics will be defined in two steps.

Definition 8 (Answer Sets, Part 1)**.** Given an answer set program Π which does not contain *not* , a consistent set of literals A is an answer set of Π if A satisfies all of the rules of Π and no subset of A satisfies the rules of Π.

Definition 9 (Reduct)**.** Given an answer set program Π and a set of literals A, the *reduct of* Π *with respect to* A (denoted by Π^A) is the program obtained from Π by:

1. removing all rules containing extended literal *not* l for any literal $l \in A$; and
2. removing all extended literals *not* l where $l \notin A$ from the remaining rules.

Definition 10 (Answer Sets, Part 2)**.** Given an answer set program Π, a consistent set of literal A is an answer set of Π if A is an answer set of Π^A.

3 Support Trees

We begin our discussion of support trees with a few basic definitions. While some readers may be familiar with several of the definitions, some other definitions will differ from those commonly seen.

Definition 11 (Supported, Unblocked, Blocked)**.** Let r be a ground rule and $I = \langle IN, OUT \rangle$ be an ordered pair of sets of literals, IN and OUT. We say

- r is supported w.r.t. I if $body^+(r) \subseteq IN$ and $body^-(r) \subseteq OUT$.
- r is unblocked w.r.t. I if $body^+(r) \subseteq IN$ and $body^-(r) \cap IN = \emptyset$.
- r is blocked w.r.t. I if $body^+(r) \cap OUT \neq \emptyset$ or $body^-(r) \cap IN \neq \emptyset$.

As will be seen, support trees may have paths that are transfinite in length (as will be shown in Example 6). In such trees there are nodes other than the root that have no immediate predecessor (no parent). Due to this, the standard definition of tree cannot be used. Instead, we will use the following definitions concerning trees from (Nerode and Shore 1997). Note that, if we were to restrict ourselves to programs in which all answer sets are finite in size, the standard definition of trees could be used.

Definition 12 (Tree)**.** A *tree* is a set whose elements, called *nodes*, are partially ordered by $<_T$, and which has a unique least element called the *root*. The predecessors of every node are well ordered by $<_T$.

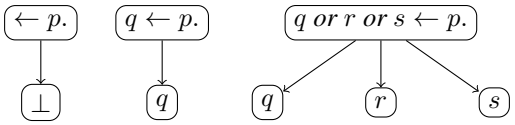
Given two nodes, x and y, x is *a child* of y (and y is *the parent* of x) if and only if $y <_T x$ and there does not exist a node z such that $y <_T z$ and $z <_T x$.

A *path* on a tree T is a maximal linearly ordered subset of T. The *path to node* n on a tree T is the maximal linearly ordered subset of T whose greatest element is n. When building a tree, a path is extended by adding each new node to the tree such that the new node is greater than (w.r.t $<_T$) any node in the path and incomparable to any node not in the path.

In this paper, each node of a tree will be labelled by either an extended literal or by a rule of an answer set program. Support trees will be formed by extending paths of the tree by appending simpler trees. These simpler trees will correspond to either supported rules or to rules that are not supported but are unblocked. Such trees will be referred to as *basic rule trees* and *chosen rule trees* respectively. Definitions and examples of each follow.

Definition 13 (Basic Rule Trees)**.** Let r be an ASP rule whose head is l_1 *or* $\ldots$ *or* l_n (where $n \geq 0$). Let τ be a tree whose root is labelled by r. If $n > 0$, the root has n children, which are leaves, where the ith child is labelled by l_i. If $n = 0$ then the root has one child, labelled by $\bot$.

Example 1 (Basic rule trees)**.** Given the following three rules:
$\leftarrow p.$
$q \leftarrow p.$
q *or* r *or* $s \leftarrow p.$
the rule trees for the rules are:



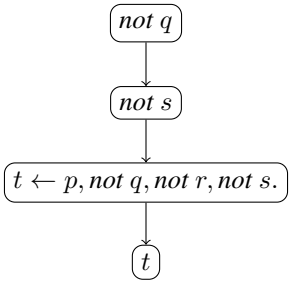
Definition 14 (Chosen Rule Tree)**.** Let r be an ASP rule and L be a set of literals. If $\{l_1, \ldots l_n\}$ is the set of all literals from $body^-(r) \backslash L$, then the *chosen rule tree for* r *with respect to* L is a tree with the following properties:

- The root of the tree is labelled by *not* l_1.
- For each $i < n$ the node labelled *not* l_i has exactly one child node, labelled *not* l_{i+1}.
- The node labelled *not* l_n has one child, the root of the basic rule tree with root r.

Example 2 (Chosen rule tree)**.** Given the rule

$$t \leftarrow p, \mathit{not}\ q, \mathit{not}\ r, \mathit{not}\ s.$$

and the set of literals $L = \{r\}$, the chosen rule tree for this rule with respect to L is:



The following notation will be used to refer to the literals that appear positively or negatively on a path.

Definition 15 ($lit(P)$, $lit^-(P)$)**.** Let τ be a tree. If P is a path on τ then:
$lit(P) = \{l \mid$ there is a node labelled by literal l on path $P\}$
$lit^-(P) = \{l \mid$ there is a node labelled by either $\overline{l}$ or *not* l on path $P\}$.

Definition 16 (Contradictory Path in a Tree)**.** Let τ be a tree and P be a path in τ. Path P is *contradictory* if $lit(P) \cap lit^-(P) \neq \emptyset$.

When building a tree, it will be necessary to know when a path of a tree is compatible with an answer set of a program. The definition below gives the precise meaning.

Definition 17 (Compatible with)**.** Let τ be a tree. If P is a path on τ and A is a set of literals then P is *compatible with* A if and only if $lit(P) \subseteq A$ and $lit^-(P) \cap A = \emptyset$.

In creating support trees for programs, to provide a normal form, we wish for the finished support tree of a given program to be unique. To help accomplish this we provide the following two definitions which provide orderings between either pairs of supported rules or between pairs of rules which are unblocked, but not supported, respectively. More general forms of support trees will be discussed in Section 5.

Definition 18 ($<_{Sup(\Pi,I)}$)**.** Let $\Pi = r_1, r_2, \ldots$ be a countable sequence of ground ASP rules and $I = \langle IN, OUT\rangle$ be an ordered pair of sets of literals, IN and OUT. Given two distinct rules, r_i and r_j, from Π which are supported w.r.t. I, then $r_i <_{Sup(\Pi,I)} r_j$ iff either:

1. $|head(r_i)| < |head(r_j)|$ or
2. $|head(r_i)| = |head(r_j)|$ and $i < j$.

Note that $<_{Sup(\Pi,I)}$ totally orders the rules of Π that are supported w.r.t. I.

Definition 19 ($<_{Unb(\Pi,I)}$)**.** Let $\Pi = r_1, r_2, \ldots$ be a countable sequence of ground ASP rules and $I = \langle IN, OUT\rangle$ be an ordered pair of sets of literals, IN and OUT. Given two distinct rules, r_i and r_j, from Π which are unblocked but not supported w.r.t. I, then $r_i <_{Unb(\Pi,I)} r_j$ iff either:

1. $body^-(r_i) \backslash OUT \subset body^-(r_j) \backslash OUT$;
2. $body^-(r_i) \backslash OUT = body^-(r_j) \backslash OUT$ and $|head(r_i)| < |head(r_j)|$; or
3. $body^-(r_i) \backslash OUT = body^-(r_j) \backslash OUT$ and $|head(r_i)| = |head(r_j)|$ and $i < j$.

Note that $<_{Unb(\Pi,I)}$ partially orders the rules of Π that are unblocked but not supported w.r.t. I. As it is only a partial order, as will be seen below, such unblocked rules will be applied in parallel, using *minimal choice sets*, as defined next.

Definition 20 (Minimal Choice Set)**.** Let $\Pi = r_1, r_2, \ldots$ be a countable set of ground ASP rules and $I = \langle IN, OUT\rangle$ be an ordered pair of sets of literals, IN and OUT.

The *minimal choice set* of Π w.r.t. I is the minimal set of rules of Π (w.r.t.$<_{Unb(\Pi,I)}$) such that for each rule r in the set:

- $head(r) \cap IN = \emptyset$; and
- r is unblocked but not supported w.r.t I.

Finally, the definition of a support tree is given. The definition, and the proofs in following section, use ordinal numbers. Those unfamiliar with ordinals can see (Nerode and Shore 1997), If all answer sets of a program are finite, natural numbers can be used and item 3 in the following definition is not needed.

Definition 21 (Support Tree)**.** Let Π be a countable set of ground ASP rules. *Support trees for* Π are defined as follows:

1. The tree, $\tau(0, \Pi)$, consisting of a single node, labelled by $\top$ is a *support tree* for Π.
2. For any ordinal α, the tree $\tau(\alpha + 1, \Pi)$, formed from $\tau(\alpha, \Pi)$ as follows, is a *support tree* for Π:
   for each noncontradictory path $P \in \tau(\alpha, \Pi)$, let $I = \langle lit(P), lit^-(P)\rangle$.
   (a) If $\exists_{r \in \Pi} head(r) \cap lit(P) = \emptyset$ and r supported by I, then let r' be the least (w.r.t. $<_{Sup(\Pi,I)}$) such rule and append the basic rule tree with root r' to path P.
   (b) If 2a does not apply, U is the minimal choice set of Π w.r.t I, and U is not empty, then for each $r \in U$, append the chosen rule tree for r w.r.t. $lit^-(P)$ to P (creating a new path for each r).
3. For any limit ordinal α, let $\tau(\alpha, \Pi) = \bigcup_{i<\alpha} \tau(i, \Pi)$ is a *support tree* for Π.

4 Properties of Support Trees

Now that the definition of support trees has been presented, this section will discuss properties of such trees, leading to the relation between certain paths of such trees and answer sets of the related program. This will also lead to the definition of a proof of a literal within an answer set.

For use in later proofs, the following definition is given.

Definition 22 (Ldepth)**.** Given a support tree, τ, and a path, $P \in \tau$, the *literal depth of* P, denoted $Ldepth(P)$, is the minimum ordinal, α, such that P is a path in $\tau(\alpha, \Pi)$. Furthermore, $Ldepth(\tau) = max_{P \in \tau} Ldepth(P)$. If n is a node on P labelled by literal l, then the Ldepth of that occurrence of l (denoted $Ldepth(l, P)$) is the Ldepth of the subpath to n on P.

Note that the $Ldepth$ is only defined for literals, not extended literals.

Observation 1. Notice that, for any countable set of rules Π and ordinal α, (2a) and (2b) from the definition of a support tree increase the $Ldepth$ of the tree by 1 (if either is applicable), hence $Ldepth(\tau(\alpha, \Pi)) \leq \alpha$.

Next it is shown that there exists a fixed-point at which the support tree of a program cannot be further extended.

Definition 23 (Finished)**.** A path in a support tree is *finished* if it cannot be extended by application of (2a) or (2b) from the definition of a support tree. A support tree is *finished* if all of its paths are finished.

Theorem 1. Let Π be a countable set of ground ASP rules. There exists a least ordinal, α, such that the support tree $\tau = \bigcup_{i \leq \alpha} \tau(i, \Pi)$ is finished.
Proof:
It can easily be seen from the definition of support tree that, for any ordinal n, $\tau(n, \Pi)$ is unique. Furthermore, it can be seen that no path is extended using a rule whose head contains a literal occurring previously on the path. Hence, no path contains more than one node labelled by the same literal. From Observation 1, if the tree is not finished, each extension increases the $Ldepth$ by 1. As there are a countable number of literals, there cannot be any paths with an uncountable $Ldepth$. Thus, if Ω is the first uncountable ordinal, then $\tau(\Omega, \Pi) = \tau(\Omega + 1, \Pi)$. Since the ordinals are well ordered, there is a smallest ordinal, α, such that $\tau(\alpha, \Pi) = \tau(\alpha + 1, \Pi)$. By definition, $\tau(\alpha, \Pi)$, the least fixed point, is finished. □

Now that the existence of a unique finished support tree has been shown, properties of paths of finished trees will be presented. It will be shown that, a set of literals in an answer set of a program if and only if there is a consistent finished path in the finished support tree that corresponds to the answer set. In one direction the proof is fairly straightforward.

Theorem 2. If Π is an ASP program, A is an answer set of Π, τ is a support tree for Π, P is a noncontradictory path in τ which is compatible with A, and $lit(P) = A$ then P is finished.
Proof:
If P could be extended by application of (2a) or (2b) from the definition of a support tree then there is some rule r whose head is not satisfied by A but is supported or unblocked with respect to $I = \langle lit(P), lit^-(P) \rangle$. It can easily be seen that, by definition of supported and unblocked, A satisfies the body of r^A. Hence A does not satisfy all the rules of rules of Π^A and therefore could not be an answer set. □

Before proving the opposite direction, several lemmas that will help in the proof will be introduced. The first two show the relationship between the rules on a path of a tree which is compatible with an answer set A and the rules in the reduct of the program with respect to A.

Lemma 3. If Π is an ASP program, A is a set of literals, $\tau(\alpha, \Pi)$ is a support tree for Π, and P is a path in $\tau(\alpha, \Pi)$ which is compatible with A, then for each rule, $r \in \Pi$, appearing on P the body of rule $r^A \in \Pi^A$ is satisfied by the set of literals appearing on the path before the rule.
Proof:
This follows directly from the definitions of support tree, minimal choice set, supported, unblocked, and satisfies. □

Lemma 4. If Π is an ASP program, τ is a support tree for Π, P is a noncontradictory finished path in τ, and $A = lit(P)$, then A satisfies all of the rules of Π^A.
Proof:
This follows directly from lemma 3 and the definition of finished. □

Next, it is shown that any unfinished path compatible with an answer set can be extended such that there is a superpath that is compatible with the answer set.

Lemma 5. If Π is an ASP program, A is an answer set of Π, $\tau(\alpha, \Pi)$ is a support tree for Π, P is a path in $\tau(\alpha, \Pi)$ which is compatible with A, and $lit(P) \subset A$, then $\tau(\alpha, \Pi)$ can be extended by application of either 2a or 2b in accordance to the definition of a support tree to create a new tree $\tau(\alpha + 1, \Pi)$ such that there is a path $P' \in \tau(\alpha + 1, \Pi)$ from root to leaf such that P is a subpath of P' and P' is compatible with A.
Proof:
Consider the following three cases:

1. Path P cannot be extended:
   In this case $lit(P)$ satisfies the rules of Π^A, but, as $lit(P) \subset A$, A would not be an answer set of Π.
2. Path P can be extended by application of 2a, but no superpath of P is compatible with A:
   Suppose rule r is the rule that would be used to extend P. Therefore, the body of r^A must be satisfied by $lit(P)$ (and hence by the literals in A). If no superpath of P is compatible with A it must be the case that $head(r) \cap A = \emptyset$, but this would mean A does not satisfy the rules of Π^A and hence A could not be an answer set.
3. Path P can be extended by application of 2b, but no superpath of P is compatible with A:
   If $I = \langle lit(P), lit^-(P) \rangle$ and U is the minimal choice set of Π w.r.t I, then there are two possibilities for this case to be true: either no rule in U has a body satisfied by A; or no rule in U whose body is satisfied by A has literal in its head which is in A. The first sub-case is similar to case 1: this would mean $lit(P)$ satisfies the rules of Π^A and

hence A could not be an answer set. The second sub-case is similar to case 2: A would not satisfy the rules of Π^A and hence A could not be an answer set. □

As will be seen, some consistent paths will correspond to answer sets. The sets of literals on other paths, however, may not meet the minimality condition in the definition of answer sets. This is due to the fact that programs may contain head-cycles. We can eliminate such non-minimal paths by requiring that paths have an additional property beyond just being consistent. This property, that of a *valid* path, will be defined next.

Definition 24 (Valid, Invalid)**.** Let Π be an ASP program, τ be a finished support tree for Π, and P be a path in τ. Path P is said to be *valid* if:

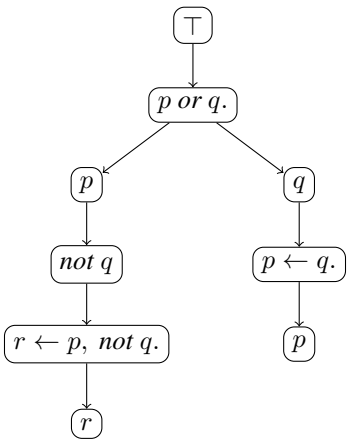
1. P is not contradictory; and
2. for any path, P', to a node labelled by a disjunctive rule, r, in τ, if P' is compatible with $lit(P)$, literal $l \in head(r)$, and $l \in lit(P)$, then there is a consistent finished path $P'' \in \tau$ which extends P' by the node labelled by l and $lit(P'') = lit(P)$.

A path is *invalid* if it is not valid.

Example 3 (Valid and Invalid Paths)**.** Given the program:

$$\Pi = \begin{cases} p \text{ or } q. \\ r \leftarrow p, \text{ not } q. \\ p \leftarrow q. \end{cases}$$

the finished support tree for the Π is:



The left path is valid as it is consistent and the only path through a disjunctive rule that is compatible with the literals on the path is the path itself. The right path, however, is invalid as the path to the node labelled by p in the left path is compatible with the literals occurring on the right path, but there is no finished path extending it which contains q.

The main theorems of the paper, showing the correspondence between valid paths of the finished support tree and answer sets of the program, are now presented.

Theorem 6. If Π is an ASP program, A is an answer set of Π, and τ is the finished support tree for Π then there exists a valid finished path P in τ such that $A = lit(P)$.
Proof:

First, we show that a finished path exists. By definition of support tree, for any path, P', compatible with A (including the path containing only the root), either $lit(P') \subset A$ and by Lemma 5 there is a path that extends P' and is compatible with A or $lit(P') = A$ and, by Theorem 2, is finished. By Theorem 1 all paths are eventually finished. This proves that at least one finished path, P, exists in τ such that $A = lit(P)$. To show that the path must also be valid consider any such path P. Since $A = lit(P)$ and A is an answer set of Π, P must be consistent. Next, by the same argument used to prove path P exists, for any path P' which meets the conditions in step 2 of the definition of valid, P' can be extended to a finished path P'' such that $lit(P'') = A$ and hence P is valid.

Theorem 7. Let Π be an ASP program. If τ is the finished support tree for Π, P is a valid path in τ, and $A = lit(P)$ then A is an answer set of Π.
Proof:
By Lemma 4, A satisfies the rules of Π^A, therefore, by the definition of answer set, either A is an answer set of Π or there exists an $A' \subset A$ which is closed under the rules of Π^A. Assume such an A' exists. Since $A' \subset A$, let α be the greatest ordinal such that there exists a $P_\alpha \in \tau(\alpha, \Pi)$ where P_α is a subpath of P and $lit(P_\alpha) \subseteq A'$. Consider $\tau(\alpha + 1, \Pi)$. By the definition of support tree, there must therefore exists some rule r which occurs on path P and extends P_α such that some literal, $l_P \in head(r)$, is the next literal on P. Notice that since r was either supported or unblocked, then the rule r^A is supported and therefore at least one literal from the head of r must also be in A'. By our choice of α, r must be disjunctive such that $l_P \notin A'$ but there exist some $l_{P'} \in head(r)$ where $l_{P'} \in A'$. Let P' be the path in $\tau(\alpha + 1, \Pi)$ that extends P_α using r with $l_{P'}$ as the next literal on the path. As $A' \subset A$, $l_{P'} \in A$ and therefore P' is also compatible with A. Notice as well that A contains at least two literals, l_p and $l_{p'}$, from $head(r)$. Since P is valid there must therefore exist a path P'_1 (or paths P'_i) that extend P' such that $lit(P'_i) = A$. But, since $A' \subset A$ there must be some greatest ordinal $\beta > \alpha + 1$ such that there is a path $P_\beta \in \tau(\beta, \Pi)$ where P_β extends P', P_β is a subpath of some P'_i, $lit(P_\beta) \subset A$, and $lit(P_\beta) \subseteq A'$. As will be shown, this is impossible. Let r_β be the rule used to extend P_β on P'_i. As above, r^A_β is supported by the literals on P_β hence at least one literal from $head(r_\beta)$ must be in A'. It is obvious that r_β cannot be a constraint. If $head(r_\beta)$ is a singleton then both A and A' contain the literal in r's head, but then β is not the greatest ordinal satisfying the conditions above. If $head(r_\beta)$ is disjunctive, A' must contain some literal from $head(r_\beta)$. As $A' \subset A$ it must be a literal that is also in A. Because P is valid, even if two or more literals from $head(r_\beta)$ are in A, any path $P_{\beta+1} \in \tau(\beta + 1, \Pi)$ which extends P_β by r_β ending in a rule from A must be a subpath of some P'_i, so again, β is not the greatest ordinal satisfying the above conditions. Therefore, there is no such β and, in turn, no such A'. Hence A is an answer set of Π. □

Theorem 8. Let Π be an ASP program. A set of literals A is an answer set of Π iff the finished support tree τ for Π contains a valid path P in τ such that $A = lit(P)$.

Proof:
Follows directly from Theorems 6 and 7. □

Returning to example 3, there was only one valid path in the finished support tree for the given program Π. It can be verified that the set of literals occurring on the path, $A = \{q, r, s\}$, is the only answer set of Π.

Proof of a literal in an answer set

One of the original goals of this work, following that of (Ben-Eliyahu and Dechter 1994), was to provide a proof as to why a literal was a member of an answer set. One could view a valid path of the finished support tree to be a proof of all literals in the answer set. This, however, would not indicate which rules were used to prove a specific literal. In order to provide more specific explanations, the following definitions are used.

Definition 25 (Necessary Proof Node)**.** Given a program, Π, the finished support tree, τ, for Π, an answer set, A, of Π, a literal l such that $l \in A$, and a path P such that $lit(P) = A$, then a node, n, of P is a *necessary proof node* in P w.r.t. l iff either:

- n is labelled by l; or
- n is labelled by a rule and is the parent of a necessary proof node in P w.r.t. l; or
- n is labelled by an extended literal l' and l' occurs in the body of a rule r such that there is a node labelled by r which is a necessary proof node in P w.r.t. l.

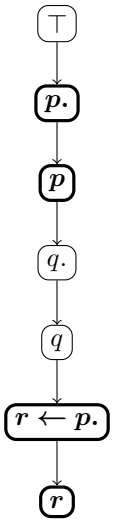
Definition 26 (Proof of a Literal in an Answer Set)**.** Given a program, Π, an answer set, A, of Π, and a literal $l \in A$, let τ be the finished support tree for Π. An ordered sequence, S, of nodes is a *proof of l in the answer set A of* Π if for some valid path $P \in \tau$ such that $lit(P) = A$, S is the subpath of P containing all, and only, necessary proof nodes in P w.r.t. l. To relate the proof to the tree, we may also refer to it as *the proof of l in the answer set A of* Π *on path P*.

The following example shows that, given an answer set of a program, there will be at least one proof of every literal in the answer set, however, there may exist proofs which do not occur in the tree.

Example 4. *(Proof)* Given the following program:

$$\Pi = \begin{cases} p. \\ q. \\ r \leftarrow p. \\ r \leftarrow q. \end{cases}$$

the only answer set is $A = \{p, q, r\}$. One could argue that there are two proofs of r in A: one since p is true and p entails r, and one because q is true and q also entails r. However, if the rules are ordered as listed, the finished support tree, τ, of Π is as shown below, with the bold nodes showing the only proof of r in A:



Often times a single proof is enough, so the fact that some proofs may not occur in the support tree is not necessarily problematic. Other proofs could be made possible by generalizing the construction of support trees, as discussed in section 5.

When a proof uses a disjunctive rule and two or more literals from the head of the rule are true in the answer set, the form of proof of a literal given above may not be convincing. In this case one may opt for the following version of a proof of a literal.

Definition 27 (Tree Proof of a Literal in an Answer Set)**.** Given a program, Π, an answer set, A, of Π, and a literal $l \in A$, let τ be the finished support tree for Π and $S = \{S' \mid S' \text{ is a proof of } l \text{ in } A\}$. A set $P \subseteq S$ is a *tree proof of l in A* if P is a nonempty, minimal (with respect to set theoretic inclusion) subset of S such that, for any $s_i, s_j \in S$ that diverge at some disjunctive rule, if $s_i \in P$ then $s_j \in P$.

Notice that if the chosen path P does not contain any disjunctive rules with two or more literals from A in the head, then the tree proof is simply the proof given by the previous definition.

5 Generalized Support Trees

The definitions given in the sections above guarantee exactly one support tree exists for any given program. While this makes it easier to prove properties of the trees, it constrains the choice of rules used when building the tree. By loosening the restrictions on the rules used to extend support trees uniqueness of the tree is lost, but freedom in building such trees is gained.

There are two fairly obvious places in which more freedom of choice may be allowed. The first is in clause 2a of the definition of support trees. In this step, among the rules that are supported and whose heads are not satisfied, the least w.r.t $<_{Sup(\Pi,I)}$ is chosen. While not proven here, the choice of any rule from among those that are supported would result in finished trees with paths equivalent to those in the definition of support tree in section 3 and hence the results should still hold.

The second place where restrictions can be loosened involves clause 2b of the definition of support trees. One could relax the definition of $<_{Unb(\Pi,I)}$ used in the construction of the minimal choice set. Recall that the definition of $<_{Unb(\Pi,I)}$ had three cases. The second and third case determine preference between rules which have the same set of unsupported rules in the negative part of the body. Again, allowing a choice among such rules would still produce correct results.

Note however that, if one chooses a subset of unblocked rules, the first case from the definition of $<_{Unb(\Pi,I)}$ cannot be dropped. This can be seen in the following example.

Example 5 (Important property of $r_i <_{Unb(\Pi,I)} r_j$)**.** Suppose the following program is given:

$$\Pi = \begin{cases} p \leftarrow \mathit{not}\ r,\ \mathit{not}\ s. \\ q \leftarrow \mathit{not}\ p,\ \mathit{not}\ r,\ \mathit{not}\ s. \end{cases}$$

When building the support tree OUT is initially empty and neither rule is supported. By definition, as both rules are unblocked but the first rule is less than the second rule w.r.t $<_{Unb(\Pi,I)}$, a path from the root is extended using the first rule. At this point $I = \langle\{p\},\{r,s\}\rangle$ and hence the second rule is blocked. The support tree is finished and has one valid finished path corresponding to the one answer set of this program, $\{p\}$. Suppose however, the definition of $<_{Unb(\Pi,I)}$ was changed to allow one to choose which rule to use if $body^-(r_i)\backslash OUT \subset body^-(r_j)\backslash OUT$. If the second rule was chosen rather than the first, the resultant tree would start with an initial path containing the chosen rule tree for the second rule. After this step $I = \langle\{q\},\{p,r,s\}\rangle$ and hence the first rule is supported. However, extending the tree using this rule results in an inconsistent path containing both p and *not* p. The tree is finished but has no valid paths.

Another form of generalized support tree would be one in which all possible rules were fired at every step. For supported rules, rather than choosing one, all such rules could be fired in parallel, creating a separate path for each possible choice. For unblocked rules, one would drop the use of the minimal choice set and simply append the chosen rule trees for each unblocked rule, in parallel, to the path. This would result in a finished support tree that was a super-tree of the one in our original definition and whose valid paths were equivalent to existing paths. one benefit of such a tree is that, for any answer set of a program, it would contain a path for every possible ordering of the rules that would lead to that answer set. However, using such a definition would often result in a much broader tree (possibly infinitely broader) with many equivalent paths. Such trees, which will be referred to as *maximal choice support trees*, can be of theoretical interest. Their practical use is, for obvious reasons, more limited.

6 Related Work

As mentioned in the introduction, the paper that motivated this work was (Ben-Eliyahu and Dechter 1994). In that paper, its authors presented a definition of a proof of a literal with respect to a context and a program. In their terminology, a context was simply a set of ground literals. Given a context, S, and a program, Π, a proof of a literal l was a sequence of rules, $r_1,\ldots,r_n$ such that:

- only one literal from the head of each r_i is a member of S (they denote this literal as $h_S(r_i)$),
- the body of each r_i is satisfied by S,
- r_1 has an empty ***positive*** body,
- for each $i > 1$, for every literal, l_j occurring in the positive body of r_i there exists a $k < i$ such that $l_j = h_S(r_k)$, and
- $l = h_S(r_n)$.

They go on to say that, if Π is an HEDLP, a consistent context S is an answer set of Π iff it satisfies all rules of Π and each literal in S has a proof with respect to S and Π. Note that the definition of proof here differs slightly from their original definition in that the word "***positive***" in our third bullet above was missing in their definition. This was clearly a minor oversight as, under their definition, the program with a single rule, "$p \leftarrow$ not q.", would not have a proof of p with respect to this program and the set $\{p\}$.

Our definition of the proof of a literal in an answer set is similar to theirs. However, it should be noted that our definition is based on our construction of the support tree while theirs defines what such a proof is but gives no constructive means of finding them. There is an obvious tradeoff in choosing one over the other, while ours are constructive, as will be shown, they will often have proofs that will not occur in our support trees (unless the tree is generalized). Another key difference is that we have no requirement that exactly one literal from the head of a rule in the proof be in the answer set. As a result, we include both rules and literals in our proof so that you know which literal was used. We also choose to keep default negated extended literals as well. These literals could have been dropped as they are clearly satisfied by the answer set, but we felt they added to the use of the proof as an explanation.

Unfortunately, if a program is a HEDLP, there is not a definite correspondence between proofs in either direction. Consider a program which contains two rules:

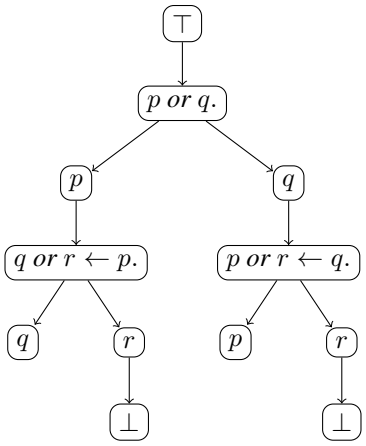
$$\Pi = \begin{cases} p. \\ p\ \mathit{or}\ q. \end{cases}$$

This program has one answer set, $A = \{p\}$. In Ben-Eliyahu and Dechter's, the sequence containing one rule, the second rule listed, would constitute a proof. Under our construction of support trees, only the first rule would be used in the tree and hence this proof would not occur.

In the other direction, consider the program:

$$\Pi = \begin{cases} p\ \mathit{or}\ q. \\ p\ \mathit{or}\ r \leftarrow q. \\ q\ \mathit{or}\ r \leftarrow p. \\ \leftarrow r. \end{cases}$$

This program also has only one answer set, $A = \{p,q\}$. If the order of the rules is the one given in the program, the finished support tree would be



Under our definition of proof, the literal p would have two possible proofs, each corresponding to one of the subpaths in the graph ending at a node labelled by p. If we consider the sequence of rules on each such subpath, neither is a proof under the definition in the paper by Ben-Eliyahu and Dechter as both literals from the head of the first rule occur in the answer set.

If we were to use maximal choice support trees however, it would hold that, for every proof of a literal l in A w.r.t. Ben-Eliyahu and Dechter's paper, there would exist a path P in the maximal choice support tree such that the proof of l in the answer set A of Π on path P would have the same sequence of rules. As the second program above showed, the converse would not be the case.

Another related work is presented in (Marek and Remmel 2012). First, it is important to note that their work allows disjunction, but no strong negation, and, as such, they use $\neg$ for default negation rather than *not*. In their paper, they define *selector functions*. Roughly speaking, the selector function indicates which non-empty set of atoms from the head of a rule should be chosen if the body of the rule is satisfied. Using this, they define *selector stable models*, which are stable models of the program that honor a chosen selector function. They further state that a set of atoms is a stable model iff it is a minimal selector stable model.

Marek and Remmel next provide the definition of a form of proof called a (D, f)*-proof scheme*. Given a disjunctive propositional logic program D and a selector function f, a (D, f)-proof scheme of length n is a sequence:

$$\langle\langle C_1, f(C_1)\rangle, \ldots, \langle C_n, f(C_n)\rangle, U\rangle$$

such for each C_i the positive part of the body is supported by the set of atoms $\bigcup_{j<i} f(C_j)$ and all atoms in the negative part of the body are in U. In comparing to our work, recall example 3. The program, Π, had one answer set $\{p, q, r\}$. Let f be the function such that $f(p \textit{ or } q.) = q$, $f(p \textit{ or } r \leftarrow q.) = r$, and, as all other rules from Π have singleton heads, f returns the head for each such rule. The following would then be the (Π, f)-proof scheme that proves s:

$$\langle\langle p \textit{ or } q., q\rangle, \langle p \textit{ or } r \leftarrow q., r\rangle, \langle s \leftarrow r,\ \textit{not } p., s\rangle, \{p\}\rangle$$

This is very close to the proof of s in the support tree shown in the example, with the only difference being the form, but not the content, of the proof. If one were to take a proof, with respect to our work presented in this paper, and choose a selector function which agreed with the sections made along the path used in our proof, then there exists an obvious equivalent proof in terms of Marek and Remmel.

Similar to the work of Ben-Eliyahu and Dechter, while such proofs exist, the definition is not constructive. They will, of course, have proofs we will not get in a support tree. Some of these proofs are somewhat strange. Consider the following program:

$$\Pi = \left\{ \begin{array}{l} p. \\ q. \\ p \textit{ or } q. \end{array} \right.$$

There is no requirement that the selector function return a single element from the head of a rule, so we may choose f such that, for each rule, the function selects all atoms in the head. As a result, the proof scheme

$$\langle\langle p \textit{ or } q., \{p, q\}\rangle, \{\ \}\rangle$$

can be seen as a proof of both p and q. This contradicts the standard wisdom that a disjunctive rule cannot be used to support more than one literal in its head. Unfortunately, this also leads to an error in their paper. Consider the program:

$$\Pi = \left\{ \begin{array}{l} p \textit{ or } q. \\ r \leftarrow p, \textit{not } q, \textit{not } r. \\ r \leftarrow q, \textit{not } p, \textit{not } r. \end{array} \right.$$

There are three possible choices for a selector function. In each, r must be selected for the second and third rules. For the first rule, however, we can select either p or q or both p and q. If we select either p or q the resulting program does not have a selector stable model, however if we choose both, the set $\{p, q\}$ is a selector stable model w.r.t. that selector function. As it is the only selector stable model, it is therefore a minimal selector stable model and by their theorem should be a stable model of the program. However, it can easily be seen that this program has no stable models. While we have not proven so, it seems that if the selector functions were restricted to selecting a single element from the head of each rule, their theorems with respect to stable models would be correct. We have not examined what effect this change would have on other results presented in their paper.

A third area of work that is less closely related is that of (Schulz and Toni 2014) on the use of argumentation to justify answer sets. In their work, assumption-based argumentation (ABA) is used to explain why a literal is, or is not, contained in a given answer set. This is, of course, somewhat different than our work in that we make no attempt at proving why something is not in an answer set. Their ABA-based answer set justifications are quite different from our proofs. Any connections between our proofs and their justifications are not obvious. The relationships between the two is left for future work.

A more recent related work is presented in (Oetsch, Pührer, and Tompits 2018). In that paper, the authors present a method for stepping through the computation of answer set. The motivation for their work was to provide for a debugging tool for answer set programming. As we only recently became aware of this work, the comparison presented

here is preliminary. A first major difference is in semantics. The semantics used in their work is that presented in (Oetsch, Pührer, and Tompits 2012). Their semantics allows for a richer class of programs which allow abstract constraints (which can be used to mimic choice rules and aggregates allowed by many current solvers). In this work, only the basic semantics are used and extensions to such larger classes are left for future work. It should be noted however, their semantics do not totally conform to those used by the most common solvers. A second difference is that their work seems to be restricted to the finite case. This makes sense as their motivation was to create a practical tool for debugging whereas our motivation was a more general theoretic result. It is likely their work could be expanded to the infinite case if desired. Finally, their computations are quite different than our trees and proofs. Further work will be needed to investigate the correlations between them.

Other related works are those of (Pontelli, Son, and El-Khatib 2008), (Liu et al. 2010), and (Brochenin, Lierler, and Maratea 2014). These works are less related, and comparison is omitted due to space considerations.

7 Areas of Application and Future Work

In this section we discuss possible applications of this research as well as future work.

Computing Answer Sets

The construction of a support tree can be viewed as a forward chaining algorithm for computing the answer sets of a program. When encountering a disjunctive rule, for each literal in the head, one can compute the sets of literals on valid paths which extend through that literal. If, for some such set, the set contained two or more literals from the head of the rule, one could then check to see if the same set of literals was on a valid path that extended through each of those other literals. If so, the path is valid, if not, it is not valid itself and can be dropped. By doing so, answer sets can be computed without the need of computing a reduct to perform a minimality check.

We are currently using this approach in our implementation of the latest version of SigmaSolver (Videtich 2014), a just-in-time grounding solver which allows for sorts and external functions.

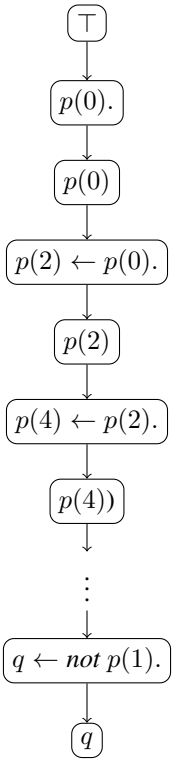
Answer Sets which are Infinite in Size

In answer set programming, if we allow an infinite set of ground terms, one can easily write non-ground programs which have infinite answer sets. In such cases, the support trees will have infinite, or even transfinite, paths. The support trees for such programs will contain one or more infinite sequences of rules and their consequences. In the tree, any such infinite sequence either ends the path or is followed by another section which starts at the next limit ordinal. Each such infinite section may have some initial distinct rules that can fire, but for any program with a finite non-ground representation there must be a loop through a pattern of ground instances within the rules in order to form the infinite sequence. The following is an example of this.

Example 6 (A transfinite tree)**.** Given the following program:

$$\Pi = \begin{cases} p(0). & \\ p(X+2) & \leftarrow \quad p(X). \\ q & \leftarrow \quad \textit{not}\ p(1). \end{cases}$$

where X ranges over natural numbers, then the finished support tree, τ for Π, is the tree:



There is only one path in this tree, which is transfinite in length (there is an infinite sequence of pairs of nodes for each subsequent $p(X)$ before the last two nodes and therefore there is no node in the tree which is a parent of the node labelled by $q \leftarrow \textit{not}\ p(1).$). The set of literals on the path is $A = \{p(0), p(2), p(4), \ldots, q\}$ which is the answer set of Π.

A study of such trees may then give insight into ways of providing finite representations of infinite answer sets. This is an area for further research in the future.

Query Answering

As one can use support trees to construct answer sets, one can also use the construction of support trees as a means to answer queries. As a simplifying example, assume the query is a single literal. Notice that, at any point in the construction of a support tree, if every consistent path contains the query, then the query must be true in any answer set of the program. This may allow one to show that a query is true even if the actual answer sets of the program are infinite and cannot be computed. Furthermore, if a valid finished path was found that did not contain the query, then the query is not supported.

Using the idea of a generalized support tree one could prioritize rules that would result in the query being made true

on a path or in the path being shown to be inconsistent. Once the query appeared on a path, that path could be abandoned in favor of expanding other paths to in turn show that they either contained the query or were inconsistent. The creation of such a query answering system could possibly be incorporated into the work on SigmaSolver, although it is not something we are working on at this time.

Extensions of Support Trees

In the work presented here, support trees were defined for very basic extended disjunctive logic programs. An obvious area for future work is to expand support trees to allow for choice rules, aggregates, and other language constructs used by current answer set solvers. While it was beyond the scope of this initial paper, such extensions will be needed for the theory to be fully applicable.

We also intend to expand the theory of support trees to cover the language of epistemic specifications (Gelfond 1994). Epistemic specifications are an extension of answer set programming that add two model operators, K and M. Due to unintended models in Gelfond's original paper, there have been several works, including (Truszczynski 2011), (Kahl 2014), and (Shen and Eiter 2016), which propose modified semantics for such programs. We intend to develop an extended version of support trees in order to capture one or more of the modified semantics. This can give more insight into the different semantics and, as with answer sets, provide an equivalent tree-based semantics. There has also been some work on solvers for the work from Kahl's doctoral dissertation (see (Kahl et al. 2015) and (Kahl, Leclerc, and Son 2016)). As with the work on SigmaSolver, such extended support trees may provide the basis for new solvers for epistemic specifications.

Explanations/Software Engineering Tools

As was shown in the section on related work, there have been several approaches that provide a definition of a proof, or other form of justification, for the occurrence of a given literal in an answer set. Such ideas are critical in forming software engineering tools. Support trees have a structure that lends itself to a simple graphical representation. The fact that they are constructive is also a benefit over some other proof methods mentioned. Development of such tools is a topic for future work.

8 Acknowledgments

The author would like to thank Michael Gelfond for numerous discussions and feedback on this work, Nelson Rushton for discussions concerning ordinals, especially the fixed-point proof, and Patrick Kahl for helping prompt this work. Their time and efforts helped make this work possible.

References

Ben-Eliyahu, R., and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence* 12(1):53–87.

Brochenin, R.; Lierler, Y.; and Maratea, M. 2014. Abstract disjunctive answer set solvers. In *Proceedings of the Twenty-first European Conference on Artificial Intelligence*, ECAI'14, 165–170. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Gelfond, M., and Kahl, Y. 2014. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. New York, NY, USA: Cambridge University Press.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In Kowalski, R., and Bowen, K., eds., *Proceeding of ICLP-88*, 1070–1080. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9(3-4):365–386.

Gelfond, M. 1994. Logic programming and reasoning with incomplete information. *Annals of Mathematics and Artificial Intelligence* 12(1):89–116.

Kahl, P.; Watson, R.; Balai, E.; Gelfond, M.; and Zhang, Y. 2015. The language of epistemic specifications (refined) including a prototype solver. *Journal of Logic and Computation* exv065.

Kahl, P. T.; Leclerc, A. P.; and Son, T. C. 2016. A parallel memory-efficient epistemic logic program solver: Harder, better, faster. *CoRR* abs/1608.06910.

Kahl, P. T. 2014. *Refining the Semantics for Epistemic Logic Programs*. Ph.D. Dissertation, Texas Tech University.

Liu, L.; Pontelli, E.; Son, T.; and Truszczynski, M. 2010. Logic programs with abstract constraint atoms: The role of computations. *Artificial Intelligence* 174:295–315.

Marek, V. W., and Remmel, J. B. 2012. Disjunctive programs with set constraints. In Erdem, E.; Lee, J.; Lierler, Y.; and Pearce, D., eds., *Correct Reasoning*. Berlin, Heidelberg: Springer-Verlag. 471–486.

Nerode, A., and Shore, R. 1997. *Logic for Applications*. Berlin, Heidelberg: Springer-Verlag, 2nd edition.

Oetsch, J.; Pührer, J.; and Tompits, H. 2012. An flp-style answer-set semantics for abstract-constraint programs with disjunctions. *Leibniz International Proceedings in Informatics, LIPIcs* 17.

Oetsch, J.; Pührer, J.; and Tompits, H. 2018. Stepwise debugging of answer-set programs. *Theory and Practice of Logic Programming* 18(1):30–80.

Pontelli, E.; Son, T. C.; and El-Khatib, O. 2008. Justifications for logic programs under answer set semantics. *CoRR* abs/0812.0790.

Schulz, C., and Toni, F. 2014. Justifying answer sets using argumentation. *CoRR* abs/1411.5635.

Shen, Y.-D., and Eiter, T. 2016. Evaluating epistemic negation in answer set programming. *Artificial Intelligence* 237:115 – 135.

Truszczynski, M. 2011. Revisiting epistemic specifications. *CoRR* abs/1108.3279.

Videtich, A. 2014. A limited grounding approach for solving answer set programs. Master's thesis, Texas Tech University.

Manipulation of Semantic Aggregation Procedures for Propositional Knowledge Bases and Argumentation Frameworks

Adrian Haret and **Johannes P. Wallner**
Institute of Logic and Computation, TU Wien, Austria

Abstract

We study the potential for manipulation of semantic aggregation procedures for argumentation frameworks (AFs) and propositional knowledge bases. The basis for our study is a framework that investigates questions of manipulability and strategy-proofness for logic-based merging, which provides the foundation to several operators for aggregating AFs. We enrich a key component of this framework—the satisfaction indices which formalize when manipulation by an agent pays off—with two main approaches to AF reasoning: skeptical and credulous acceptance of arguments. In propositional knowledge bases, skeptical consequences are propositions true in all models, while credulous consequences are propositions true in at least one model. We find that, even in restricted cases, most aggregation procedures are vulnerable to manipulation by an agent for both (i) propositional knowledge bases and (ii) AFs under the stable, preferred, and grounded semantics. However, aggregation is not manipulable (i.e., is strategy-proof) under one well-known operator when the bases, or AFs, are semantically complete (i.e., have exactly one semantical model). Finally, we provide complexity results for computing our novel indices.

1 Introduction

Complex decision making often faces the challenge of aggregating multiple, possibly conflicting viewpoints. While the literature on social choice (Lang and Xia 2016; Zwicker 2016) offers a wealth of aggregation procedures, these studies also highlight an important stumbling block to their successful implementation: participants in a deliberation process may have an incentive to misrepresent their positions, if doing so can bring an advantage. Effects of manipulation of the aggregation process are various: from the possibility that overall non-optimal results may be chosen (Faliszewski and Procaccia 2010; Conitzer and Walsh 2016), to calling into question the meaning of the whole process, as well as potential societal harm that can be inflicted by malicious agents (Brundage et al. 2018).

The field of formal argumentation in Artificial Intelligence (Bench-Capon and Dunne 2007), which lends itself naturally to (group) deliberation through collective argumentation, has, in recent years, witnessed a steady increase in research on aggregation procedures. Most of the study of aggregation in argumentation focuses on the core formalism of argumentation frameworks (AFs) (Dung 1995). AFs underlie many approaches to formal argumentation in AI, and can be represented as directed graphs whose vertices are arguments and the directed edges represent directed conflicts (*attacks*) among the arguments. Semantics of AFs, several of which have been proposed to different effect, specify criteria for selecting sets of arguments (called *extensions*) that can be reasonably deemed to be jointly acceptable.

When considering argumentation in a multi-agent setting, it is plausible that agents have different stances towards arguments and disagree about where to place conflicts. Following up on this idea, strategies for aggregating AFs can be distinguished among two main lines (see the recent survey by (Bodanza, Tohmé, and Auday 2017)): (i) approaches that aggregate the graph structures, and (ii) approaches that aggregate the semantics of the AFs (Delobelle et al. 2016), which can be seen as a procedure for aggregating different viewpoints that can arise in a debate. In this work we focus on the latter, semantic approach.

Example 1. City Hall must decide if it will develop a certain area of the city. Two projects are put forward: expensive high-rises and affordable housing. The constraint City Hall is operating under is that either development goes through with exactly *one* of the projects, or the whole thing is abandoned. Furthermore, City Hall wants to involve the community in the deliberative process, and four groups of interest are invited. The main arguments being deliberated upon are that the area must be developed (a), that development should consist of high-rises (b), or affordable housing (c). The stances of the four agents are as follows: (1) real-estate developers argue that developing an area and building affordable housing are at odds with each other, thereby favouring expensive high-rises (a and c mutually attack each other); (2) residents of the adjacent neighborhoods hold the opposite viewpoint, arguing that high-rises and development do not go together, as expensive skyscrapers raise rent levels and erode feeling of community (a and b mutually attack each other); (3) a group of social activists holds that affordable housing is a necessity, and that it should take precedence over ambitions of development and high-rises (c attacks other arguments); (4) a businessman who happens to own a large part of the land under consideration for development thinks there should certainly be investment in the area, but that investment should go neither into high-rises nor af-

fordable housing, but rather into something like office space, as that would bring more profit (a attacks other arguments).

The differing opinions give rise to different conflicts between the three main arguments, depicted in Figure 1, with AFs F_{1-4} corresponding to each of the interested parties. Under the standard stable semantics for AFs, we get the following stable extensions: viewpoints of F_1 are $\{a, b\}$ and $\{b, c\}$; F_2 accepts $\{a, c\}$ and $\{b, c\}$; F_3 considers $\{c\}$ acceptable; F_4 chooses $\{a\}$. We aggregate these extensions using the standard aggregation operator $\Delta_{\mu}^{d_H,\Sigma}$. Intuitively, this operator returns extensions that minimize the sum (Σ) of symmetric differences (d_H) to the given extensions, while being restricted to outcomes permitted by the integrity constraint μ, which in this case is equivalent to $(\neg a \wedge \neg b \wedge \neg c) \vee (a \wedge (b \leftrightarrow \neg c)$. Application of $\Delta_{\mu}^{d_H,\Sigma}$ yields the unique extension $\{a, c\}$. But if F_4 opts to, untruthfully, misrepresent their views in a way that implies that b shall also be accepted (e.g, by leaving out the attack from a to b), then the result changes to two extensions: $\{a, b\}$ and $\{a, c\}$.

As seen in the example, reporting untruthfully can change the aggregated outcome: in Example 1, the effect is to introduce doubt on whether c is unambiguously part of the result. This might seem, at first glance, innocuous: but if one focuses on the skeptical consequences (a main reasoning task in formal argumentation, looking at arguments present in all extensions), then the skeptical consequences in Example 1 change from $\{a, c\}$ to $\{a\}$: exactly what F_4, the businessman, is aiming for.

In this paper we look at semantic aggregation of AFs, to see how the result can be influenced by one agent misrepresenting their viewpoint, when satisfaction with the result is computed considering the skeptical (or credulous) consequences of the aggregated outcome, i.e., by considering arguments that are part of all (or some) extensions. Our approach to the aggregation of AFs and its manipulation follows the framework of propositional merging (Konieczny and Pérez 2011; Everaere, Konieczny, and Marquis 2007), where the objects of study are operators that aggregate the semantics of propositional knowledge bases (or formulas), rather than AFs. Hence, we are able to understand aggregation of AFs by studying *propositional* aggregation operators: this is fitting, as operators for the aggregation of AFs proposed by (Delobelle et al. 2016) are, at their core, distance-based operators from propositional merging (Konieczny and Pérez 2011). However, the framework for manipulability of (Everaere, Konieczny, and Marquis 2007) (henceforth, *EKM*) does not include the connection to argumentative reasoning, in particular with respect to skeptical and credulous consequences, and can thus not be directly applied to our setting. Our main contributions are as follows:

- We extend EKM's framework to include the skeptical and credulous consequences, in the form of novel satisfaction indices, which measure the distance between an agent's truthful knowledge and the outcome of aggregation.
- We give the full landscape of (non-)manipulability for aggregating in the propositional case, which extends the reach of our contributions to not only AFs, but any semantic aggregation procedure inspired by propositional merging based on AF-like skeptical and credulous consequences. Concretely, we show that all main aggregation operators are manipulable wrt our new indices, except when aggregating so-called complete bases (i.e., each formula has exactly one model) without integrity constraint and using aggregation operator $\Delta_{\top}^{d_H,\Sigma}$ (defined below).
- Inspired by strategy-proofness (non-manipulability) of $\Delta_{\top}^{d_H,\Sigma}$, we extend an earlier observation by (Delobelle et al. 2016) that this operator respects certain kinds of majorities in the input, while other operators fail to do so.
- We provide complexity results for computing our novel indices, which can be computed in polynomial time for complete bases without integrity constraint, but are intractable in other settings.
- We show that all main results of manipulability of aggregating propositional bases can be imported to semantic aggregation of AFs under the grounded, stable, and preferred AF semantics, including the strategy-proof case.

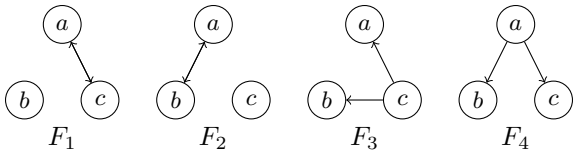


Figure 1: AFs for viewpoints in Example 1.

For some proofs of the results, we refer to the appendix.

2 Aggregation of Propositional Bases

We assume a finite set $\mathcal{P}$ of propositional atoms and $\mathcal{L}$ is the set of formulas generated from $\mathcal{P}$ with the usual connectives. A (knowledge) base K is a formula from $\mathcal{L}$ and a *profile* $P = (K_1, \ldots, K_n)$ is a finite collection of bases. The universe $\mathcal{U}$ is the set of possible interpretations for formulas in $\mathcal{L}$. The models of a propositional formula μ are the interpretations which satisfy it, and we write $[\mu]$ for the set of models of μ. We typically write interpretations as words where letters are the atoms assigned to true, e.g., $\{\{a, b\}, \{b, c\}\}$ is written as $\{ab, bc\}$. If $\varphi_1, \varphi_2 \in \mathcal{L}$, we say that $\varphi_1 \models \varphi_2$, if $[\varphi_1] \subseteq [\varphi_2]$, and that $\varphi_1 \equiv \varphi_2$, if $[\varphi_1] = [\varphi_2]$. A formula φ is consistent (satisfiable) if $[\varphi] \neq \emptyset$. The set of consistent knowledge bases is $\mathcal{L}_c$. By $v \Delta w$ we denote the symmetric difference of atoms assigned to true between two interpretations v and w, defined as $v \Delta w = (v \setminus w) \cup (w \setminus v)$.

In the framework of logic-based merging (Konieczny and Pérez 2011), aggregation of a profile $P = (K_1, \ldots, K_n)$ of consistent propositional bases is achieved through an operator $\Delta_{\mu}^{d,f}$ working on the semantic level, the main ingredients of which are as follows: a distance d between interpretations, an aggregation function f and an integrity constraint (a formula) μ. We recall choices for d, f, and μ in the following.

If v and w are two interpretations, the drastic distance d_D is 0 iff both interpretations are identical and 1 otherwise. The Hamming distance d_H denotes the size of the symmetric difference of v and w, i.e., $d_H(v, w) = |v \Delta w|$. If there is no danger of ambiguity, we drop subscripts H or D from

d_H and d_D. We can then lift the distance notion to a distance $d(v, [\varphi])$ between an interpretation v and a set of interpretations $[\varphi]$ for a formula φ, by taking $d(v, [\varphi])$ to be the minimum of the distances of v to each model of φ.

Definition 1. If v is an interpretation, $d \in \{d_H, d_D\}$ is a distance, and φ is a propositional formula, then $d(v, [\varphi]) = min_{w \models \varphi} d(v, w)$.

Using an aggregation function f, we further lift the distance notions to talk about distances between an interpretation v and a profile $P = (K_1, \ldots, K_n)$. If $X = (x_1, \ldots, x_n)$ is a tuple of non-negative integers, we use the following aggregation functions:

- $\Sigma(X) = \Sigma_{i=1}^{n} x_i$,
- $\texttt{max}(X) = max(\{x_i \mid 1 \leq i \leq n\})$, and
- $\texttt{GMax}(X)$ is X in descending order.

Definition 2. If v is an interpretation, $d \in \{d_H, d_D\}$ is a distance, $f \in \{\Sigma, \texttt{max}, \texttt{GMax}\}$ is an aggregation function and $P = (K_1, \ldots, K_n)$ is a profile, then $d_f(v, P) = f(d(v, K_1), \ldots, d(v, K_n))$.

That is, $d_f(v, P)$ is the result of aggregating, via f, the distances between v and each $K_i \in P$. For $f \in \{\Sigma, \texttt{max}\}$ $d_f(v, P)$ is an integer, and thus interpretations can be ordered with respect to their distance to P. For $f = \texttt{GMax}$, $d_{\texttt{GMax}}(v, P)$ is an n-tuple made up of the numbers $d(v, K_1), \ldots, d(v, K_n)$ ordered in descending order. To rank interpretations via GMax one then uses an order on such tuples, typically a lexicographic ordering: $(x_1, \ldots, x_n) <_{lex} (y_1, \ldots, y_n)$ if $x_i < y_i$ for the first i where x_i and y_i differ from each other. We often omit for a distance d the aggregation function f, i.e., write d instead of d_f, unless not clear from the context.

An aggregation operator $\Delta_\mu^{d,f}$ is then defined as follows.

Definition 3. Let $P = (K_1, \ldots, K_n)$ a profile, d a distance between interpretations, f an aggregation function, and μ a propositional formula (integrity constraint). The *propositional aggregation operator* $\Delta_\mu^{d,f}$ is defined as $[\Delta_\mu^{d,f}(P)] = min([\mu], \leq_P) = \{v \in [\mu] \mid \nexists w \in [\mu], w \leq_P v\}$, with $\leq_P$ defined by $v \leq_P w$ iff $d_f(v, P) \leq d_f(w, P)$ for $f \in \{\Sigma, \texttt{max}\}$ and $d_f(v, P) \leq_{lex} d_f(w, P)$ for $f = \texttt{GMax}$.

In this definition, the result of aggregation of all bases in a profile is a set of interpretations. This set consists of all models of the integrity constraint μ which have a minimum distance to the profile, with the distances specified via distance notion d and aggregation function f.[1]

Example 2. We can represent the truthful viewpoints of the four agents in Example 1 by four propositional knowledge bases K_{1-4} over the alphabet $\mathcal{P} = \{a, b, c\}$, where $[K_1] = \{ab, bc\}$, $[K_2] = \{ac, bc\}$, $[K_3] = \{c\}$ and $[K_4] = \{a\}$. To this we add knowledge base K_4', which represents agent 4's reported beliefs, where $[K_4'] = \{ab\}$. We write $P^t = (K_1, K_2, K_3, K_4)$ for the truthful profile (containing the true beliefs of agent 4), and P^m for the manipulated profile (containing K_{1-3} and K_4' instead of K_4). The constraint is given by the propositional formula $\mu = (a \wedge (b \leftrightarrow \neg c)) \vee (\neg a \wedge \neg b \wedge \neg c)$. Table 1 illustrates the results of aggregating the profiles P^t and P^m under constraint μ with the operator $\Delta_\mu^{d_H,\Sigma}$.

More concretely, the first column contains all interpretations over $\mathcal{P}$, the second to sixth columns show the distances (in this case, Hamming distances) between each interpretation and the models of the various bases. The final two columns show, for each interpretation, the aggregated distance under Σ. The result can be read off Table 1 by picking, among the models of μ (highlighted in gray), the ones with minimum aggregated distance (written in bold font). For instance, $d_H(ab, [K_2]) = d_H(ab, [ac, bc]) = min(d_H(ab, ac), d_H(ab, bc)) = min(2, 2) = 2$. We get that $[\Delta_\mu^{d_H,\Sigma}(P)] = \{ac\}$ and $[\Delta_\mu^{d_H,\Sigma}(P')] = \{ab, ac\}$, i.e., by reporting a different K_4', agent 4 has introduced a seed of doubt as to whether c is unambiguously part of the result.

Note that $\Delta_\mu^{d_D,\Sigma}$ and $\Delta_\mu^{d_D,\texttt{GMax}}$ are equivalent, for any profile P and constraint formula μ ($[\Delta_\mu^{d_D,\Sigma}(P)] = [\Delta_\mu^{d_D,\texttt{GMax}}(P)]$). Thus, we will usually only specify results for the former. The operator $\Delta_\mu^{d_D,\texttt{max}}$ delivers $[\bigwedge P \wedge \mu]$, if consistent, and $[\mu]$ otherwise.

3 The EKM Framework for Manipulability

We recall Everaere, Konieczny, and Marquis's (2007) framework for manipulability when aggregating propositional bases. In this framework, a satisfaction index i measures how "satisfied" an agent is with the outcome of aggregation. Formally, such an index is defined as follows.

Definition 4. A *satisfaction index* $i : \mathcal{L}_c \times \mathcal{L}_c \to \mathbb{N}^+$ is a function that maps a pair of satisfiable formulas to a non-negative integer.

For a given profile $P = (K_1, \ldots, K_n)$, a satisfaction index i formally defines each agent's satisfaction with the outcome of aggregation, written $i(K_j, \Delta_\mu^{d,f}(P))$, where K_j is an agent's reported base. In our setting, the lower $i(K_j, \Delta_\mu^{d,f}(P))$ is, the better the outcome is for K_j.[2] Manipulation occurs when an agent is able to achieve a better (lower) satisfaction index by reporting a different base K_j'. In other words, manipulation occurs when $i(K_j, \Delta_\mu^{d,f}(P^m)) < i(K_j, \Delta_\mu^{d,f}(P^t))$, where $P^t = (P_1, \ldots, P_n)$ is the truthful profile and $P^m = \{K_1, \ldots, K_{j-1}, K_j', K_{j+1}, \ldots, K_n\}$ is a (manipulated) profile that is the same as P^t, except that K_j is replaced by K_j'.

Definition 5. If i is a satisfaction index, a profile P^t is *manipulable by* $K \in P^t$ *for* i *and an aggregation operator* $\Delta_\mu^{d,f}$ if there is a base K' s.t. $i(K, \Delta_\mu^{d,f}(P^m)) < i(K, \Delta_\mu^{d,f}(P^t))$, with $P^m = (P^t \setminus \{K\}) \cup \{K'\}$.

[1] Recall, we assume that each base in the profile is consistent.

[2] In the EKM framework (Everaere, Konieczny, and Marquis 2007) an agent is more satisfied when the index increases; in our case, it is more consistent with the spirit of the indices we consider to reverse this metric.

	$[K_1]$ $\{ab, bc\}$	$[K_2]$ $\{ac, bc\}$	$[K_3]$ $\{c\}$	$[K_4]$ $\{a\}$	$[K'_4]$ $\{ab\}$	$\Delta_\mu^{d_H,\Sigma}(P^t)$	$\Delta_\mu^{d_H,\Sigma}(P^m)$
$\emptyset$	2	2	1	1	2	6	7
a	1	1	2	0	1	4	5
b	1	1	2	2	1	6	5
c	1	1	0	2	3	4	5
ab	0	2	3	1	0	6	**5**
ac	2	0	1	1	2	**4**	**5**
bc	0	0	1	3	2	4	3
abc	1	1	2	2	1	6	5

Table 1: Example of aggregation and manipulation. Gray cells denote the permitted models when integrity constraint $\mu = (a \wedge (b \leftrightarrow \neg c)) \vee (\neg a \wedge \neg b \wedge \neg c)$. Bold numbers indicate models with minimum distance.

We say that an operator $\Delta_\mu^{d,f}$ is *manipulable for index* i if there exists a profile that is manipulable for i and $\Delta_\mu^{d,f}$. An operator $\Delta_\mu^{d,f}$ is *strategy-proof* if there is no possibility that an agent can report a different base than her true one and achieve a better index. Formally, this is defined as follows.

Definition 6. Let i be a satisfaction index. An aggregation operator $\Delta_\mu^{d,f}$ is *strategy-proof for* i iff there is no integrity constraint μ and profile P s.t. P is manipulable for i.

4 Satisfaction Indices for Acceptance

In this section we present our new satisfaction indices, inspired by an argument's acceptance with respect to an AF. Particularly, semantics of AFs are criteria for distinguishing between sets of arguments that are acceptable (*extensions*) and sets that are not, i.e., each AF semantics (of which several exist) defines sets of extensions, on the basis of which specific arguments are then selected as accepted. We discuss AF background formally in Section 8. Here we focus on the last step, i.e., notions of credulous/skeptical acceptance, adapted, in our case, to models of propositional formulas.

Definition 7. If M is a non-empty set of interpretations over $\mathcal{P}$, we define the following sets:

- $\texttt{Cred}(M) = \bigcup_{v \in M} v$,
- $\texttt{Skept}(M) = \bigcap_{v \in M} v$,
- $\texttt{No}(M) = \mathcal{P} \setminus \texttt{Cred}(M)$.

In words, for a formula φ and its set of models $[\varphi]$, an atom is in $\texttt{Cred}([\varphi])$ (*credulously accepted*) if it is true in at least one model; an atom is in $\texttt{Skept}([\varphi])$ (*skeptically accepted*) if it is true in all models; and an atom is in $\texttt{No}([\varphi])$ if it is true in no model. Based on these notions of acceptance of atoms, we define the following satisfaction indices.

Definition 8. If K and K_Δ are two propositional formulas, we define the following indices:

$$i_{\texttt{Skept}}(K, K_\Delta) = |\texttt{Skept}([K]) \Delta \texttt{Skept}([K_\Delta])|$$
$$i_{\texttt{Cred}}(K, K_\Delta) = |\texttt{Cred}([K]) \Delta \texttt{Cred}([K_\Delta])|, \text{ and}$$
$$i_{\texttt{No}}(K, K_\Delta) = |\texttt{No}([K]) \Delta \texttt{No}([K_\Delta])|.$$

The indices represent the difference of accepted atoms (arguments) with respect to credulous (skeptical, no) acceptance for an agent's true base K and an aggregated outcome K_Δ. That is, an agent is fully satisfied, with regards to an index $i \in \{i_{\texttt{Skept}}, i_{\texttt{Cred}}, i_{\texttt{No}}\}$ when the corresponding (not) accepted atoms are the same for her own base and the outcome, in which case the index is equal to zero. Each difference in the accepted atoms contributes one to the dissatisfaction of an agent. For instance, if an atom a is true in all models of an agent's base, but not true in all models of the aggregated outcome, then this contributes one to the skeptical index $i_{\texttt{Skept}}$; similarly, if a is not true in all models of an agent's base, but true in all models of the aggregated outcome, this, likewise, contributes one to the skeptical index.

Example 3. Consider the two sets $[\varphi] = \{ab, a\}$ and $[\psi] = \{ab, b\}$. We have $\texttt{Cred}([\varphi]) = \{a, b\} = \texttt{Cred}([\psi])$, $\texttt{Skept}([\varphi]) = \{a\}$, and $\texttt{Skept}([\psi]) = \{b\}$. Then the indices are $i_{\texttt{Cred}}(\varphi, \psi) = 0$ and $i_{\texttt{Skept}}(\varphi, \psi) = 2$.

It can also be the case that the credulous index is higher than the skeptical index, witnessed by $[\varphi'] = \{a, b\}$ and $[\psi'] = \{b, c\}$. We have $\texttt{Cred}([\varphi']) = \{a, b\}$, $\texttt{Cred}([\psi']) = \{b, c\}$, $\texttt{Skept}([\varphi']) = \emptyset = \texttt{Skept}([\psi'])$. The indices are $i_{\texttt{Cred}}(\varphi, \psi) = 2$ and $i_{\texttt{Skept}}(\varphi, \psi) = 0$.

As seen in the preceding example, indices $i_{\texttt{Skept}}$ and $i_{\texttt{Cred}}$ do not directly relate to each other, in that each may be higher or lower than the other. On the other hand, the indices denoting credulous acceptance and no acceptance (i.e., not credulously accepted) coincide.

Observation 1. For any two formulas φ and ψ, it holds that $i_{\texttt{Cred}}(\varphi, \psi) = i_{\texttt{No}}(\varphi, \psi)$.

5 Manipulability and Strategy-proofness

In this section we look at manipulability of propositional merging operators under our indices. We begin with an illustrative case of manipulation by reference to the, by now familiar, running example.

Example 4. Recall the scenario in Example 1, formalized using four AFs, depicted in Figure 1, and the constraint μ. The extensions of these AFs were aggregated in Example 2 through the operator $\Delta_\mu^{d_H,\Sigma}$, with the result of aggregating the truthful profile P^t being the set of interpretations $\{ac\}$. Notice that these extensions can just as easily be thought of as the models of propositional formulas: for instance, agent 1 is represented by

the formula $K_1 = b \wedge (a \leftrightarrow \neg c)$, whose models are $[K_1] = \{ab, bc\}$. Agent 4's satisfaction with the result, if measured according to the skeptical index, is $|a \Delta ac| = 1$, since $\texttt{Skept}([K_4]) = a$ and $\texttt{Skept}([\Delta_\mu^{d_H,\Sigma}(P^t)]) = \texttt{Skept}(\{ac\}) = ac$. However, if agent 4 submits a base K_4' with $[K_4'] = \{ab\}$, the result of aggregating the manipulated profile P^m becomes $[\Delta_\mu^{d_H,\Sigma}(P^m)] = \{ab, ac\}$, thereby bringing agent 4's skeptical satisfaction index down to 0, since $\texttt{Skept}([\Delta_\mu^{d_H,\Sigma}(P^m)]) = \texttt{Skept}(\{ab, ac\}) = a$. Thus, agent 4 can manipulate the aggregation result by submitting different opinions than the ones it actually holds.

As Example 4 shows, manipulation is possible in the general case for aggregation operator $\Delta_\mu^{d_H,\Sigma}$. In the following, we provide a more detailed analysis of the operators and fragments of the propositional language for which manipulation can occur. For the sake of readability, we illustrate manipulative cases mainly for the aggregation operator $\Delta_\mu^{d_H,\Sigma}$ and the skeptical index and refer the reader to the appendix for examples regarding the other operators (and also for further proof details). For the example, we assume a profile $P^t = (K_1, \ldots, K_n)$, where K_n is taken to be the manipulating agent's true beliefs and K_n' its reported beliefs, when it is advantageous for the agent to report beliefs that are different from its true ones. Consequently, $P^m = (K_1, \ldots, K_n')$ is the manipulated profile. Note that in many of the subsequent results, we assume that $\mu \equiv \top$. We investigate the effect of a different μ at the end of this section.

General Case. We first look at the full propositional language, i.e., the given profile P may contain any Boolean formulas. As already indicated in Example 4, manipulation is possible in the general case when taking the Hamming distance and the sum aggregation function. In the following theorem we summarize our findings that all considered aggregation operators are manipulable in the general case. The proof of this theorem follows from formal statements below, which show manipulability even in more restricted cases.

Theorem 1. *Let $d \in \{d_H, d_D\}$ and $f \in \{\Sigma, \texttt{max}, \texttt{GMax}\}$. The operator $\Delta_\mu^{d,f}$ is manipulable for the skeptical and credulous indices.*

Complete Bases. A base (a Boolean formula) K is *complete* if K has exactly one model, i.e., $|[K]| = 1$. Accordingly, a profile is complete if each base in it is complete.

In the EKM framework (Everaere, Konieczny, and Marquis 2007) strategy proofness is obtained for certain operators and indices under certain restrictions, e.g., making the manipulating agent's knowledge base K_n complete. Interestingly, this result does not carry over for the indices we are working with.

Proposition 2. *Let $d \in \{d_H, d_D\}$ and $f \in \{\Sigma, \texttt{max}, \texttt{GMax}\}$. The operator $\Delta_\top^{d,f}$ is manipulable for the skeptical and credulous indices, even if the manipulating agent's truthful and changed bases are complete.*

Proof for $\Delta_\top^{d_H,\Sigma}$ and $i_{\texttt{Skept}}$. Consider $[K_1] = \{bc, ac\}$, $[K_2] = \{ab\}$, and $[K_2'] = \{a\}$, with the profiles being $P^t = (K_1, K_2)$ and $P^m = (K_1, K_2')$. We get $[\Delta_\top^{d_H,\Sigma}(P^t)] = \{a, b, ab, ac, bc, abc\}$ and $[\Delta_\top^{d_H,\Sigma}(P^m)] = \{a, ac\}$. The skeptical index of K_2 with respect to the manipulated profile from the truthful profile goes from 2 to 1. □

Proposition 2 tells us that, if we suspect a particular agent of being manipulative, then requiring them to remove any kind of uncertainty about their stance on the issues deliberated upon (i.e., requesting an explicit "yes" or "no" wrt every atom), does not prevent that agent from manipulation.

Interestingly, requiring the same for all agents, that is, requiring the whole profile to be complete, in fact implies strategy-proofness when aggregating under the Hamming distance and the sum aggregation function. Technically, the result relies on the fact that the operator $\Delta_\top^{d_H,\Sigma}$ respects majorities, in the sense that a manipulative agent cannot "override" majorities wrt skeptical or credulous consequences. Formally, we use the following definition and a lemma. The definition states that an agent supports skeptical (credulous) consequence of an atom if that atom follows skeptically (credulously) from that agent's base. The number of such supports is defined as follows.

Definition 9. Let P be a profile and $a \in \mathcal{P}$. We define $\texttt{Credsupp}_P(a) = |\{K \in P \mid a \in \texttt{Cred}([K])\}|$ and $\texttt{Skeptsupp}_P(a) = |\{K \in P \mid a \in \texttt{Skept}([K])\}|$.

Now, the following lemma gives a characterization of the aggregation via $\Delta_\top^{d_H,\Sigma}$ when the profile is complete.

Lemma 3. *Let $P = \{K_1, \ldots, K_n\}$ be a profile of complete bases, and $M = [\Delta_\top^{d_H,\Sigma}(P)]$. For any $v \in M$, it holds that $\texttt{Skeptsupp}_P(x) > \frac{n}{2}$ implies $x \in v$ and $\texttt{Skeptsupp}_P(x) < \frac{n}{2}$ implies $x \notin v$.*

Thus, if an interpretation v is part of the semantical output, for a complete profile and when using $\Delta_\top^{d_H,\Sigma}$, then v must assign all atoms to true that are true in a majority of bases, and to false whenever that atom is false in a majority of bases. The remaining atoms, without majorities, receive all values. With this we can show the following result.

Theorem 4. *The operator $\Delta_\top^{d_H,\Sigma}$ is strategy-proof for the skeptical and credulous indices if the profile is complete.*

Proof. (sketch) This follows from Lemma 3: whenever an agent changes the value of an atom, then majorities might change; yet, changing an atom's value to true cannot remove that atom as a skeptical consequence. Analogous reasoning shows the remaining cases. □

Unfortunately, strategy proofness does not hold when switching from the sum aggregation function to either max or GMax aggregation functions, as stated next.

Proposition 5. *The operators $\Delta_\top^{d_H,\texttt{GMax}}$ and $\Delta_\top^{d_H,\texttt{max}}$ are manipulable for the skeptical and credulous indices, even if the given profile is complete.*

Even/Odd Number of Bases. All examples of manipulation provided so far have used profiles with an even number of bases in them. Thus, one might wonder whether this makes a difference to the manipulability of the operators considered so far.

Proposition 6. *For $d \in \{d_H, d_D\}$ and $f \in \{\Sigma, \texttt{max}, \texttt{GMax}\}$, the operators $\Delta_{\top}^{d,f}$ are manipulable for $i \in \{i_{\texttt{Skept}}, i_{\texttt{Cred}}\}$, with both an odd and an even number of bases in the profile, even if the manipulating base is complete.*

Proof for $\Delta_{\top}^{d_H,\Sigma}$ and $i_{\texttt{Skept}}$. Prop. 2 proves the claim for an even number. Therefore, we show that profiles with an odd number of bases can be manipulated. For the skeptical index and the operators $\Delta_{\top}^{d_H,\Sigma}$ and $\Delta_{\top}^{d_D,\Sigma}$, take the profile $P^t = (K_1, K_2, K_3)$, where $[K_1] = \{ab, bc\}$, $[K_2] = \{ac, bc\}$ and $[K_3] = \{b\}$. Agent 3 manipulates with $[K_3'] = \{ab\}$. □

A particular consequence of Proposition 6 (derived from Proposition 2) is that the operators we are working with are manipulable even if the manipulating base is complete *and* the profile contains only two bases. This is another instance in which restrictions which make merging operators strategy-proof in (Everaere, Konieczny, and Marquis 2007) prove to be ineffectual here.

Influence of the Integrity Constraint The only strategy-proof case identified so far is obtained when merging complete bases under $\Delta_{\mu}^{d_H,\Sigma}$ with $\mu \equiv \top$. It is natural to ask, then: what if $\mu \not\equiv \top$? The answer turns out to be that strategy-proofness does not hold anymore when permitting arbitrary integrity constraints.

Proposition 7. *The operator $\Delta_{\mu}^{d_H,\Sigma}$ is manipulable for $i \in \{i_{\texttt{Skept}}, i_{\texttt{Cred}}\}$ for complete bases and arbitrary μ.*

Proof for $\Delta_{\mu}^{d_H,\Sigma}$. Let $[K_1] = \{abce\}$, and $[K_2] = \{e\}$. Define μ with $[\mu] = \{abe, cde\}$. We find that $[\Delta_{\mu}^{d_H,\Sigma}(K_1, K_2)] = \{abe\}$. When the second base is changed to $[K_2'] = \{ce\}$, we get $[\Delta_{\mu}^{d_H,\Sigma}(K_1, K_2')] = \{abe, cde\}$. Truthfully, the second agent's skeptical index is 2, while with the modified base the same agent achieves a skeptical index of 0. For the credulous index, take $[K_1] = \{d\}$, $[K_2] = \{abcd\}$, and $[K_2'] = \{abd\}$. The credulous index decreases by 2 when the second agent reports K_2. □

6 One Agent Influencing the Outcome

In the previous section we investigated (non-)manipulability, and the only strategy-proof case turned out to be with complete bases when aggregating under $\Delta_{\top}^{d_H,\Sigma}$. The main insight for why this holds is that a manipulative agent cannot overturn certain notions of majority. In this section we consider a related problem, namely whether an agent (manipulative or not) can influence the outcome of aggregation in terms of skeptical and credulous consequences.

We turn our attention again to the aggregation operator $\Delta_{\top}^{d_H,\Sigma}$. We show that, under this operator, certain forms of majority cannot be overturned, even for non-complete bases. In particular, in the setting of $n-1$ agents, if more than $\frac{n}{2}$ support skeptical acceptance or the credulous non-acceptance of a proposition a, then this is also the case when aggregating under the profile with an additional nth agent (who may argue wrt a arbitrarily). This means that the nth agent's contribution to the aggregation cannot change these kinds of majorities. Note that this is not in contrast to the manipulability results above: while the nth agent cannot remove the skeptical acceptance of a, in case of a majority, the nth agent can change skeptical consequences of propositions that do not have a majority in their favour.

We first show in the following lemma that whenever a majority of agents do not support (as in Def. 9) the credulous consequence of an atom, then this is reflected in the aggregated outcome, as well.

Lemma 8. *Let $P = (K_1, \ldots, K_{n-1})$ be a profile and $a \in \mathcal{P}$. If $\texttt{Credsupp}_P(a) < \frac{n}{2}$, then $a \notin \texttt{Cred}([\Delta_{\top}^{d_H,\Sigma}(P \cup \{K_n\})])$ for any base K_n.*

Regarding skeptical consequences, a similar result holds, which was presented in (Delobelle et al. 2016). Taken together, this implies the next proposition.

Proposition 9. *Let $P = (K_1, \ldots, K_{n-1})$ be a profile. Further, let $X = \{x \mid \texttt{Skeptsupp}_P(x) > \frac{n}{2}\}$ and $Y = \{x \mid \texttt{Credsupp}_P(x) < \frac{n}{2}\}$. For any base K_n and $M = \Delta_{\top}^{d_H,\Sigma}(P \cup \{K_n\})$, it holds that*

- $X \subseteq \texttt{Skept}(M)$, *and*
- $Y \subseteq (\mathcal{P} \setminus \texttt{Cred}(M))$.

Comparing Proposition 9 and Lemma 3, we see that the latter is concerned with a full characterization under complete bases, and the former with arbitrary bases; yet, instead of $\texttt{Skeptsupp}_P(x) < \frac{n}{2}$, this condition needs to be strengthened to $\texttt{Credsupp}_P(x) < \frac{n}{2}$.

A similar result does not hold for operator $\Delta_{\top}^{d_H,\texttt{max}}$, i.e., when using max instead of Σ. Thus, for max majorities may be overturned, as illustrated in the next example.

Example 5. Consider $[K_1] = \{abc\}$, $[K_2] = \{b\}$, and $[K_3] = \{c\}$. Using aggregation operator $\Delta_{\top}^{d_H,\texttt{max}}$, the result is $\{bc\}$. When the first agents reports $[K_1'] = \{ab\}$ instead of K_1, the result is $\{\emptyset, a, b, bc, abc\}$. Thus, by changing the reported base, the first agent can get a to be true in a model of the output, even if less than half of the agents have a assigned to true in some model of their base (in fact only the first agent accepts a credulously).

7 Computational Complexity

In this section we present complexity results for computing our new indices. We make use of the complexity classes P, NP, $coNP$, DP, FP^{NP}, and $FP^{NP[log\ n]}$, and assume familiarity with the first three classes. A decision problem is in DP if this problem is the intersection of a problem in NP and a problem in $coNP$. A function problem is in FP^{NP} if the solution can be computed by a poly-time algorithm with access to an NP oracle. If the number of calls is bounded logarithmically, then the problem is in $FP^{NP[log\ n]}$.

First, we turn our attention to complete bases, particularly, the case when each agent's model is given explicitly, e.g., as a list. We call such a profile an explicit complete profile.

Proposition 10. *Computing the skeptical and credulous indices for an explicit complete profile can be done in polynomial time for aggregation operator $\Delta_{\top}^{d,f}$ with $d = d_D$ and $f \in \{\Sigma, \texttt{max}, \texttt{GMax}\}$, and with $d = d_H$ and $f = \Sigma$.*

However, when the integrity constraint $\mu \not\equiv \top$, the situation significantly changes: it is already intractable to decide whether an agent is fully satisfied, as we show next.

Proposition 11. *Deciding whether the skeptical index is equal to zero for a given explicit complete profile and operator $\Delta_\mu^{d,f}$ is*

- *coNP-hard for $d = d_H$ and $f = \Sigma$, and*
- *DP-complete for $d = d_D$ and $f \in \{\Sigma, \texttt{max}, \texttt{GMax}\}$.*

Generally, one can derive the skeptical and credulous indices for an agent, and any associated profile, via an FP^{NP} algorithm, in case the skeptical (credulous) consequences of that agent are given explicitly as input (i.e., input includes a list of the agent's skeptical or credulous consequences). The following results use general algorithms for propositional merging from (Konieczny, Lang, and Marquis 2002).

Proposition 12. *Computing the skeptical (credulous) index for an agent in a given profile with explicit skeptical (credulous) consequences of that agent's base is*

- *in $FP^{NP[log\ n]}$ for aggregation operator $\Delta_\mu^{d_H,f}$ and $f \in \{\Sigma, \texttt{max}\}$, and*
- *in FP^{NP} for aggregation operator $\Delta_\mu^{d_H,\texttt{GMax}}$.*

8 Manipulation when Aggregating AFs

In this section we import the preceding results to the case of aggregating semantics of argumentation frameworks (Dung 1995) when using the operators defined by (Delobelle et al. 2016). We start with defining necessary preliminaries.

Definition 10. An *argumentation framework (AF)* is a pair $F = (A, R)$, where A is a finite set of arguments and $R \subseteq A \times A$ is the attack relation. The pair $(a, b) \in R$ means that a attacks b. An argument $a \in A$ is *defended* (in F) by a set $S \subseteq A$ if, for each $b \in A$ such that $(b, a) \in R$, there exists a $c \in S$ such that $(c, b) \in R$.

Semantics for AFs are defined through a function σ which assigns to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. We consider for σ the functions *adm*, *grd*, *prf*, and *stb*, standing for admissible, grounded, preferred, and stable respectively. We make use of the characteristic function of AFs, defined for an AF $F = (A, R)$, by $\mathcal{F}_F(S) = \{x \in A \mid x \text{ is defended by } S\}$. Moreover, for a set $S \subseteq A$, the *range of S* is $S_R^+ = S \cup \{x \mid (y, x) \in R, y \in S\}$.

Definition 11. Let $F = (A, R)$ be an AF. An $S \subseteq A$ is conflict-free (in F), if there are no $a, b \in S$, s.t. $(a, b) \in R$. We denote conflict-free sets by *cf*(F). For an $S \in$ *cf*(F), it holds that $S \in$ *stb*(F) iff $S_R^+ = A$; $S \in$ *adm*(F) iff $S \subseteq \mathcal{F}_F(S)$; $S \in$ *grd*(F) iff S is the l.f.p. of $\mathcal{F}_F$; and $S \in$ *prf*(F) iff $S \in$ *adm*(F) and there is no $T \in$ *adm*(F) with $S \subset T$.

Example 6. Let $F = (\{a, b, c, d\}, R)$ be an AF with $R = \{(a, b), (b, a), (a, c), (b, c), (c, d)\}$. We have *stb*$(F) =$ *prf*$(F) = \{ad, bd\}$ and *grd*$(F) = \{\emptyset\}$.

We note that, in this work, when referring to stable semantics, we assume that any AF F has at least one stable extension, i.e., *stb*$(F) \neq \emptyset$.

Aggregation of AFs, according to (Delobelle et al. 2016), takes as input a profile of AFs $P = (F_1, \ldots, F_n)$, where all the F_i's are over the same set of arguments A, and a semantics σ. Aggregation proceeds in two steps: (1) aggregate $(\sigma(F_1), \ldots, \sigma(F_n))$ (i.e., the semantics of the F_i's), which results in a set $S \subseteq 2^A$; (2) if required, an AF F', or a set of AFs $W = \{F'_1, \ldots, F'_m\}$ is constructed s.t. $\sigma(F') = S$, or $S = \bigcup_{F'_i \in W} \sigma(F'_i)$. Here we focus on the semantical side of aggregation, without concrete instantiations of AFs for the aggregated result. In case the AFs do not share the same set of arguments, we consider an expansion of each AF to the "whole" set of arguments as studied in (Coste-Marquis et al. 2007), i.e., we assume that each AF contains the same set of arguments.

With regards to concrete operators that aggregate AFs' semantics, in the work of (Delobelle et al. 2016), the same operators can be used as when merging semantics of propositional bases.

Definition 12. Given a profile of AFs $P = (F_1, \ldots, F_n)$ and a semantics σ, define the semantics of aggregation operator $\Delta_\mu^{d,f}$ by $[\Delta_\mu^{d,f}(P)] = [\Delta_\mu^{d,f}(P')]$ with $P' = (K_1, \ldots, K_n)$ and $[K_i] = \sigma(F_i)$ for all $1 \leq i \leq n$.

That is, the result of aggregating a profile of AFs, under operator $\Delta_\mu^{d,f}$ is the same as when aggregating propositional bases whose semantics are the same as for the AFs under the considered AF semantics, i.e., $[K_i] = \sigma(F_i)$ for all i.

Example 7. Consider the AFs from Example 2 shown in Figure 1, the same integrity constraint $\mu = (a \wedge (b \leftrightarrow \neg c)) \vee (\neg a \wedge \neg b \wedge \neg c)$, and operator $\Delta_\mu^{d_H,\Sigma}$. The semantical output, under stable and preferred semantics, is shown, in detail in Table 1. More concretely, we have $[\Delta_\mu^{d_H,\Sigma}(\{F_1, F_2, F_3, F_4\})] = \{ac\}$.

There is one barrier to porting our results from propositional bases to AFs, under a semantics σ: AF semantics are not as expressive as the (classical) propositional logic. In other words, for any finite set of interpretations M there exists a propositional formula φ s.t. $M = [\varphi]$. However, the same does not hold for AFs, i.e., there are sets of sets of arguments M s.t. there is no AF F with $M = \sigma(F)$ for a specific semantics σ (Dunne et al. 2015). Thus, if we give examples for manipulation in the form of a set of models M for a base K ($M = [K]$), the corresponding result can only be adapted to AFs if the set of models, interpreted as a set of sets arguments, is in the signature $\Sigma_\sigma = \{\sigma(F) \mid F \text{ is an AF}\}$ for semantics σ, i.e., it must hold that $M \in \Sigma_\sigma$. We have chosen our examples for manipulability in such a way that if a base K is complete, then $[K] \in \Sigma_\sigma$ for $\sigma \in \{$*stb*, *prf*, *grd*$\}$ and if K is not complete then $[K] \in \Sigma_{\sigma'}$ for $\sigma' \in \{$*stb*, *prf*$\}$. That is, in this case, there is an AF whose stable (preferred, grounded) semantics is equal to $[K]$ in case the base is complete, and equal to stable (preferred) if K is not complete. For complete bases, the only problematic case is when $\emptyset$ is among the set of models: then the only AF F s.t. $\emptyset \in$ *stb*(F) is the AF with no arguments. For non-complete bases there is, naturally, no AF whose grounded semantics matches (since there is always only one grounded extension); for stable and preferred semantics we have tailored the examples accordingly. This allows us to port all major results of aggregating propositional

bases directly to AFs, as we summarize in the following theorems. As for propositional bases, an AF F is complete, for semantics σ, if $|\sigma(F)| = 1$.

Theorem 13. *Let $d \in \{d_H, d_D\}$, $f \in \{\Sigma, \texttt{max}, \texttt{GMax}\}$, $f' \in \{\texttt{max}, \texttt{GMax}\}$, and $i \in \{i_{\texttt{Cred}}, i_{\texttt{Skept}}\}$. Aggregating a profile of AFs is manipulable for i under*

- *the aggregation operator $\Delta_{\top}^{d,f}$ with semantics $\sigma \in \{prf, stb\}$, even if the manipulating agent's AF and changed AF are complete for σ;*
- *the aggregation operator $\Delta_{\top}^{d_H,f'}$ with semantics $\sigma \in \{prf, stb, grd\}$, even if the profile is complete for σ; and*
- *the aggregation operator $\Delta_{\mu}^{d_H,\Sigma}$ with semantics $\sigma \in \{prf, stb, grd\}$, even if the profile is complete for σ and μ is permitted to be arbitrary.*

Theorem 14. *Aggregating a complete profile of AFs for $\sigma \in \{prf, stb, grd\}$ is strategy-proof under the aggregation operator $\Delta_{\top}^{d_H,\Sigma}$.*

9 Related work

Relation to the EKM framework. The EKM framework (Everaere, Konieczny, and Marquis 2007) studies three indices: strong/weak drastic, and probabilistic. For each of these three indices and our skeptical and credulous indices, one can find example profiles s.t. their indices are lower (higher) than our new indices. Further, (non-)manipulability in the EKM framework does not transfer directly to our work: strategy-proofness for our indices holds for complete bases and $\mu \equiv \top$, while in the EKM framework there are other restrictions which ensure strategy-proofness for their indices but not for our indices.

Social Choice. Since aggregation in the belief-merging framework typically delivers a set of interpretations, there is an obvious parallel to be drawn with multi-winner voting and committee elections (Meir et al. 2008; Amanatidis et al. 2015; Barrot, Lang, and Yokoo 2017; Faliszewski et al. 2017), according to which a merging operator is like a voting rule and interpretations are candidates. However, there are certain differences between the problem as it is studied in social choice and as we approach it: for one thing, multi-winner elections often assume that the output is a set of candidates of fixed size, whereas we make no such assumption; furthermore, the way in which we have defined the satisfaction indices and the presence of constraints does not make it immediately clear how results on manipulation of multi-winner voting rules would apply to the scenarios we study.

Our work intersects with social choice in the special case when the profile is complete and the number of bases is odd. Indeed, in this case the aggregation problem corresponds to a Judgment Aggregation problem, with the $\Delta_{\top}^{d_H,\Sigma}$ delivering the majority opinion on the atoms (considered as issues). Our strategy-proofness result for $\Delta_{\top}^{d_H,\Sigma}$ dovetails neatly with a similar result in Judgment Aggregation (Endriss 2016; Baumeister, Rothe, and Selker 2017), though our treatment is slightly more general, as it accommodates both an even and an odd number of bases. That $\Delta_{\top}^{d_H,\Sigma}$ delivers the majority opinion on atoms when the profile is complete corresponds to an observation, made before, that the majority opinion minimizes the sum of the Hamming distances to voters' approval ballots (Brams, Kilgour, and Sanver 2007).

Aggregation on Argumentation Frameworks. The study of aggregation issues with respect to AFs was launched in (Coste-Marquis et al. 2007). The current state of the art is surveyed in (Bodanza, Tohmé, and Auday 2017) and (Delobelle, Konieczny, and Vesic 2018) is a more recent contribution. It is apparent from the survey that the operators introduced in (Delobelle et al. 2016) and studied here are the only ones focusing on the semantic aggregation of sets of extensions. Several further properties have been studied, e.g., axioms and complexity (Dunne, Marquis, and Wooldridge 2012), graph aggregation in general (Endriss and Grandi 2017), and preservation of semantic properties (Chen and Endriss 2017). Manipulation, and strategy-proofness, was studied for a setting of judgment aggregation of AFs (Awad et al. 2017a; 2017b; 2017c; Caminada and Booth 2016; Booth, Awad, and Rahwan 2014) and for argumentation mechanism design (Rahwan, Larson, and Tohmé 2009). Operators studied for judgment aggregation are different: the input of each agent and the output is restricted to one labelling. Argumentation mechanism design differs from our setting in that agents put forward arguments into an AF, from which acceptability of arguments is checked. A very recent approach (Lisowski, Doutre, and Grandi 2018) studies manipulation issues when aggregating orders on values (i.e., aggregating preference orders) in a value-based AF (Bench-Capon 2003).

Rationalizability. In (Airiau et al. 2017) a restriction on profiles to aggregate is studied. In brief a profile is called rationalizable whenever there exists a particular AF F^* s.t. it is possible to "recover" each AF of the profile when associating preferences over arguments of F^* which discard attacks of strictly less preferred attackers wrt the attackee (with potentially further constraints). The examples we utilize, when ported to AFs, from Proposition 2 can be adapted to be rationalizable, implying that main results regarding manipulability and strategy-proofness carry over to aggregation of rationalizable profiles. We think it is interesting for future work to study under which further restrictions (non-)manipulability remains, such as those that arise, e.g., when using structured argumentation formalisms (e.g. (Modgil and Prakken 2013)) and different preferences for each agent.

10 Conclusions

In this work we have looked at the potential for manipulation of semantic aggregation procedures for AFs and propositional bases. Since the semantics of AFs and propositional formulas are similar, aggregation at this level can be handled within a single framework, namely that of logic-based merging. We modified the existing EKM framework (Everaere, Konieczny, and Marquis 2007) by looking at new satisfaction indices, defined using notions of acceptability imported from abstract argumentation. We found that, with these indices, there is a wide avenue for manipulation, even in cases

where the EKM framework found strategy-proofness. Importantly, we showed that one operator ($\Delta_{\top}^{d_H,\Sigma}$) resists manipulation in the case when all bases are complete, which is, e.g., the case when aggregating AFs under the grounded semantics. Further, we presented complexity results that indicate tractability for computing our new indices for complete bases, yet intractability on other settings.

Future work will aim to understand manipulability for more operators, finding further cases of strategy-proofness, and obtaining a general result on strategy-proofness for merging operators, in the spirit of the Gibbard-Satterthwaite theorem in Social Choice (Gibbard 1973; Satterthwaite 1975). An important aspect worth pursuing is an agent's best response, if it knows the viewpoints of other agents. Thus, future work in this direction would focus on understanding the manipulation task from an algorithmic perspective.

Appendix

Proof of Thm. 1 and Prop. 2. ($\Delta_{\top}^{d_H,\Sigma}$) For the credulous index, take $[K_1] = \{b, c\}$, $[K_2] = \{a\}$, and $[K_2'] = \{ab\}$. We get $[\Delta_{\top}^{d_H,\Sigma}(\{K_1, K_2\})] = \{\emptyset, a, b, c, ab, ac\}$ and $[\Delta_{\top}^{d_H,\Sigma}(\{K_1, K_2'\})] = \{b, ab\}$. The credulous index of K_2 wrt the manipulated profile from the truthful profile goes from 2 to 1. ($\Delta_{\mu}^{d_H,\texttt{max}}$, $\Delta_{\mu}^{d_H,\texttt{GMax}}$) For the skeptical (credulous) index, the same knowledge bases as for $\Delta_{\mu}^{d_H,\Sigma}$ with the skeptical (credulous) index work here. ($\Delta_{\mu}^{d_D,\Sigma}$, $\Delta_{\mu}^{d_D,\texttt{max}}$) For the skeptical index, take $[K_1] = \{ab, bc\}$, $[K_2] = \{abc\}$, and $[K_2'] = \{ab\}$. For the credulous index, take $[K_1] = \{ac, bc\}$, $[K_2] = \{c\}$, and $[K_2'] = \{ac\}$. □

Proof sketch of Lemma 3. Let $X = \{v \mid \texttt{Skeptsupp}_P(x) > \frac{n}{2} \Rightarrow x \in v, \texttt{Skeptsupp}_P(x) < \frac{n}{2} \Rightarrow x \notin v\}$. Suppose $M \not\subseteq X$, i.e., there is a $v \in M$ s.t. $v \notin X$. This means there is an x s.t. $x \notin v$ and $\texttt{Skeptsupp}_P(x) > \frac{n}{2}$, or $x \in v$ and $\texttt{Skeptsupp}_P(x) < \frac{n}{2}$. Suppose the first case (other case symmetric). Then $d(v \cup \{x\}, P) < d(v, P)$, since strictly more than half models of the complete bases assign x to true. Thus, $v \notin M$. Suppose $X \not\subseteq M$, i.e., there is a $v \in X$ s.t. $v \notin M$. Thus, there is a $w \in M$ s.t. $d(w, P) < d(v, P)$. By the previous reasoning, we know that $M \subseteq X$, and that $w \in X$. This means that for all x if strictly more than half models of the complete bases assign x to true (false), then $x \in w$ ($x \notin w$). Thus, $x \in v$ iff $x \in w$ for all x where there is a strict majority, and $x \notin v$ iff $x \notin w$ where there is a strict majority against. Therefore, there is a y s.t. $y \in v$ and $y \notin w$ or $y \notin v$ and $y \in w$ where $\texttt{Skeptsupp}_P(y) = \frac{n}{2}$. In this case, we have $d(v, P) = d(w, P)$. □

Proof of Proposition 5. For the skeptical index, take a profile $P^t = (K_1, \ldots, K_6)$, where $[K_1] = [K_6] = \{abcd\}$, $[K_2] = \{abd\}$, $[K_3] = \{bcd\}$, $[K_4] = \{acd\}$, $[K_5] = \{d\}$. Agent 6 manipulates with $[K_6'] = \{ab\}$. If $P^m = (K_1, \ldots, K_5, K_6')$, we get $[\Delta_{\top}^{d_H,\texttt{GMax}}(P^t)] = \{abd, acd, bcd\}$ and $[\Delta_{\top}^{d_H,\texttt{GMax}}(P^m)] = \{abd\}$. The skeptical index of K_6 goes from 3 to 1. The same example holds for $\Delta_{\mu}^{d_H,\texttt{max}}$ For the credulous index and $\Delta_{\mu}^{d_H,\texttt{GMax}}$, take a profile $P^t = (K_1, \ldots, K_6)$, where $[K_1] = [K_6] = \{d\}$, $[K_2] = \{ad\}$, $[K_3] = \{bd\}$, $[K_4] = \{cd\}$, $[K_5] = \{abcd\}$. Agent 6 manipulates with $[K_6'] = \{ad\}$. We get $[\Delta_{\top}^{d_H,\texttt{GMax}}(P^t)] = \{ad, bd, cd\}$ and $[\Delta_{\top}^{d_H,\texttt{GMax}}(P^m)] = \{ad\}$. The credulous index of K_6 goes from 3 to 1. For the credulous index and $\Delta_{\mu}^{d_H,\texttt{max}}$, take K_{1-6} as before and $[K_6'] = \{ab\}$, with the index going from 3 to 2. □

Proof of Proposition 6. For $\Delta_{\top}^{d_H,\texttt{GMax}}$ and $\Delta_{\top}^{d_H,\texttt{max}}$, take K_1 and K_2 as for Σ and $[K_3] = \{ab\}$. Agent 3 manipulates with $[K_3''] = \{a\}$. If $[K_3] = \{abc\}$, agent 3 manipulates $\Delta_{\top}^{d_D,\texttt{max}}$ with $[K_3'''] = \{bc\}$. For the credulous index and $\Delta_{\top}^{d_H,\Sigma}$ and $\Delta_{\top}^{d_D,\Sigma}$, take $[K_1] = \{a, b\}$, $[K_2] = \{a, c\}$ and $[K_3] = \{abc\}$: agent 3 manipulates with $[K_3'] = \{b\}$. If $[K_3] = \{b\}$, agent 3 manipulates $\Delta_{\top}^{d_H,\texttt{GMax}}$ and $\Delta_{\top}^{d_H,\texttt{max}}$ with $[K_3''] = \{bc\}$. For $\Delta_{\top}^{d_D,\texttt{max}}$, take $[K_1] = [K_2] = \{ab\}$ and $[K_3] = \{a\}$. Agent 3 manipulates with $[K_3'''] = \{ab\}$. □

Proof sketch of Lemma 8. Let $P' = (K_1, \ldots, K_{n-1}, K_n)$ for any K_n. Further, let v be an interpretation over $\mathcal{P}$ with $a \in v$ and $\nexists v'$ over $\mathcal{P}$ s.t. $a \in v'$ and $d(v, P') < d(v', P')$ (v has minimum sum of distances to all agents wrt interpretations that include a). It holds that for $v \setminus \{a\} = w$ we have $d(v, P') > d(w, P')$, which follows from the observation that strictly less than $\frac{n}{2}$ agents have a model with a true. □

Proof of Proposition 10. For $d = d_H$ and $f = \Sigma$, Lemma 3 yields the result. For $d = d_D$ and Σ, only models of the profile can be in the output (all others have higher distance). For $\texttt{max}$, the output is only weakly constrained: each interpretation is in the output, except when all bases agree on the model (then the output is that model). □

Proof sketch of Proposition 11. ($\Delta_{\mu}^{d_H,\Sigma}$) We show *coNP* hardness via a reduction from checking whether a formula φ, over X is tautological. Construct profile $P = (K_1, K_2, K_3)$ and integrity constraint μ. Further let D be a set of fresh atoms with $|D| > 3 \cdot |X|$. Let $K_1 = \bigwedge_{x \in \mathcal{P}} \neg x$, $K_2 = K_3 = \bigwedge_{d \in D} d \wedge \bigwedge_{x \in \mathcal{P} \setminus D} \neg x$, and $\mu = (\varphi \rightarrow \bigwedge_{d \in D} \neg d) \wedge (\neg\varphi \rightarrow \bigwedge_{d \in D} d)$. It holds that φ is tautological iff there are no skeptical consequences of $\Delta_{\mu}^{d_H,\Sigma}(P)$ (trivial if φ is tautological; otherwise non-models of φ with D true have lower distance than models of φ and D false). ($\Delta_{\mu}^{d_D}$) We reduce from checking whether $\varphi \models \top$ and ψ is sat. W.l.o.g. we assume that φ is satisfiable. Construct $P = (K_1)$ with $K_1 = a \wedge \neg b \wedge \neg c \wedge \neg d \wedge \bigwedge_{x \in \mathcal{P} \setminus \{a,b,c,d\}} \neg x$, and $\mu = (b \nleftrightarrow c) \wedge (\varphi \rightarrow a) \wedge (\neg\psi \rightarrow d)$. It holds that $i_{\texttt{Skept}}$ is zero iff $\varphi \models \top$ and ψ is satisfiable (since $|P| = 1$, f is redundant; $K_1 \wedge \mu \models \bot$, thus only models of μ are relevant). Finally, membership follows from checking $\mu \models a$ ($\mu \not\models a$), and whether $v \models \mu$ for base's models v. □

Acknowledgments

This work was supported by the Austrian Science Fund (FWF): P30168-N31 and Y698.

References

Airiau, S.; Bonzon, E.; Endriss, U.; Maudet, N.; and Rossit, J. 2017. Rationalisation of profiles of abstract argumentation frameworks: Characterisation and complexity. *J. Artif. Intell. Res.* 60:149–177.

Amanatidis, G.; Barrot, N.; Lang, J.; Markakis, E.; and Ries, B. 2015. Multiple referenda and multiwinner elections using hamming distances: Complexity and manipulability. In *Proc. AAMAS 2015*, 715–723.

Awad, E.; Bonnefon, J.; Caminada, M.; Malone, T. W.; and Rahwan, I. 2017a. Experimental assessment of aggregation principles in argumentation-enabled collective intelligence. *ACM Trans. Internet Techn.* 17(3):29:1–29:21.

Awad, E.; Booth, R.; Tohmé, F.; and Rahwan, I. 2017b. Judgement aggregation in multi-agent argumentation. *J. Log. Comput.* 27(1):227–259.

Awad, E.; Caminada, M. W. A.; Pigozzi, G.; Podlaszewski, M.; and Rahwan, I. 2017c. Pareto optimality and strategy-proofness in group argument evaluation. *J. Log. Comput.* 27(8):2581–2609.

Barrot, N.; Lang, J.; and Yokoo, M. 2017. Manipulation of hamming-based approval voting for multiple referenda and committee elections. In *Proc. AAMAS 2017*, 597–605.

Baumeister, D.; Rothe, J.; and Selker, A.-K. 2017. Strategic behavior in judgment aggregation. In Endriss, U., ed., *Trends in Computational Social Choice*. AI Access. 145–168.

Bench-Capon, T. J. M., and Dunne, P. E. 2007. Argumentation in artificial intelligence. *Artif. Intell.* 171(10-15):619–641.

Bench-Capon, T. J. M. 2003. Persuasion in practical argument using value-based argumentation frameworks. *J. Log. Comput.* 13(3):429–448.

Bodanza, G. A.; Tohmé, F.; and Auday, M. 2017. Collective argumentation: A survey of aggregation issues around argumentation frameworks. *Argument & Computation* 8(1):1–34.

Booth, R.; Awad, E.; and Rahwan, I. 2014. Interval methods for judgment aggregation in argumentation. In *Proc. KR 2014*, 594–597.

Brams, S. J.; Kilgour, D. M.; and Sanver, M. R. 2007. A minimax procedure for electing committees. *Public Choice* 132(3):401–420.

Brundage, M.; Avin, S.; Clark, J.; Toner, H.; Eckersley, P.; Garfinkel, B.; Dafoe, A.; Scharre, P.; Zeitzoff, T.; Filar, B.; Anderson, H. S.; Roff, H.; Allen, G. C.; Steinhardt, J.; Flynn, C.; hÉigeartaigh, S. Ó.; Beard, S.; Belfield, H.; Farquhar, S.; Lyle, C.; Crootof, R.; Evans, O.; Page, M.; Bryson, J.; Yampolskiy, R.; and Amodei, D. 2018. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *CoRR* abs/1802.07228.

Caminada, M., and Booth, R. 2016. A dialectical approach for argument-based judgment aggregation. In *Proc. COMMA 2016*, volume 287 of *FAIA*, 179–190.

Chen, W., and Endriss, U. 2017. Preservation of semantic properties during the aggregation of abstract argumentation frameworks. In *Proc. TARK 2017*, volume 251 of *EPTCS*, 118–133.

Conitzer, V., and Walsh, T. 2016. Barriers to manipulation in voting. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. 127–145.

Coste-Marquis, S.; Devred, C.; Konieczny, S.; Lagasquie-Schiex, M.; and Marquis, P. 2007. On the merging of Dung's argumentation systems. *Artif. Intell.* 171(10-15):730–753.

Delobelle, J.; Haret, A.; Konieczny, S.; Mailly, J.; Rossit, J.; and Woltran, S. 2016. Merging of abstract argumentation frameworks. In *Proc. KR 2016*, 33–42.

Delobelle, J.; Konieczny, S.; and Vesic, S. 2018. On the aggregation of argumentation frameworks: operators and postulates. *J. Log. Comput.* DOI: 10.1093/logcom/exy023.

Dung, P. M. 1995. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.* 77(2):321–357.

Dunne, P. E.; Dvořák, W.; Linsbichler, T.; and Woltran, S. 2015. Characteristics of multiple viewpoints in abstract argumentation. *Artif. Intell.* 228:153–178.

Dunne, P. E.; Marquis, P.; and Wooldridge, M. 2012. Argument aggregation: Basic axioms and complexity results. In *Proc. COMMA 2012*, volume 245 of *FAIA*, 129–140.

Endriss, U., and Grandi, U. 2017. Graph aggregation. *Artif. Intell.* 245:86–114.

Endriss, U. 2016. Judgment Aggregation. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. 399–426.

Everaere, P.; Konieczny, S.; and Marquis, P. 2007. The strategy-proofness landscape of merging. *J. Artif. Intell. Res.* 28:49–105.

Faliszewski, P., and Procaccia, A. D. 2010. AI's war on manipulation: Are we winning? *AI Magazine* 31(4):53–64.

Faliszewski, P.; Skowron, P.; Slinko, A.; and Talmon, N. 2017. Multiwinner voting: A new challenge for social choice theory. In Endriss, U., ed., *Trends in Computational Social Choice*. AI Access. 27–47.

Gibbard, A. 1973. Manipulation of voting schemes: a general result. *Econometrica: journal of the Econometric Society* 587–601.

Konieczny, S., and Pérez, R. P. 2011. Logic based merging. *J. Philosophical Logic* 40(2):239–270.

Konieczny, S.; Lang, J.; and Marquis, P. 2002. Distance based merging: A general framework and some complexity results. In *Proc. KR 2002*, 97–108.

Lang, J., and Xia, L. 2016. Voting in combinatorial domains. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. 197–222.

Lisowski, G.; Doutre, S.; and Grandi, U. 2018. Preventing manipulation in aggregating audiences in value-based argumentation frameworks. In *Proc. SAFA*, volume 2171 of *CEUR Workshop Proceedings*, 48–59.

Meir, R.; Procaccia, A. D.; Rosenschein, J. S.; and Zohar, A. 2008. Complexity of strategic behavior in multi-winner elections. *J. Artif. Intell. Res.* 33:149–178.

Modgil, S., and Prakken, H. 2013. A general account of argumentation with preferences. *Artif. Intell.* 195:361–397.

Rahwan, I.; Larson, K.; and Tohmé, F. A. 2009. A characterisation of strategy-proofness for grounded argumentation semantics. In *Proc. IJCAI 2009*, 251–256.

Satterthwaite, M. A. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of economic theory* 10(2):187–217.

Zwicker, W. S. 2016. Introduction to the theory of voting. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*. Cambridge University Press. 23–56.

Belief Revision Operators with Varying Attitudes Towards Initial Beliefs

Adrian Haret and **Stefan Woltran**
Institute of Logic and Computation, TU Wien, Austria

Abstract

Classical axiomatizations of belief revision include a postulate stating that if new information is consistent with initial beliefs, then revision amounts to simply adding the new information to the initial knowledge base. This postulate assumes a conservative attitude towards initial beliefs, in the sense that an agent faced with the task of revising them will seek to preserve initial beliefs as much as possible. In this work we look at operators that can assume different attitudes towards original beliefs, and make the case that these operators can be put to use when doing revision in fragments of propositional logic. We provide axiomatizations of these operators by varying the aforementioned postulate and obtain representation results which characterize the new types of operators in terms of preorders on possible worlds. We also present concrete examples for each new type of operator, using notions inspired from decision theory.

Introduction

Belief revision models rational changes of an agent's epistemic state, triggered by the availability of new, trusted information. In the standard logical approach, an agent's epistemic state is represented by propositional formulas, while the standards of rationality that a revision operator is expected to abide by are encoded as logical axioms (Alchourrón *et al.* 1985; Gärdenfors 1988; Katsuno and Mendelzon 1992b; Fermé and Hansson 2018). Remarkably, the classical set of revision postulates turn out to define a class of operators that can be looked at in two ways: on the one hand, as change, guided by logical postulates, of propositional theories in response to new data; and on the other hand, as choice functions over possible worlds exploiting plausibility rankings over such interpretations. This correspondence tells us that an agent faced with revision of its initial beliefs acts as if it chooses from a set of feasible possible worlds the ones that it considers most plausible.

A distinguishing feature of revision operators, as typically axiomatized, is that they can be assumed to adopt a particular attitude towards initial beliefs, enforced through what are called the *Inclusion* and *Vacuity* postulates in the AGM formulation (Fermé and Hansson 2018), or through a single postulate equivalent to their conjunction in the KM axiomatization (Katsuno and Mendelzon 1992b). This attitude spells out how the agent's prior information behaves with respect to new data: thus, in the KM axiomatization, the postulate in question states that if new information μ is consistent with existing beliefs κ, then the result of revision is simply $\kappa \wedge \mu$. In other words, the agent retains its initial beliefs and simply supplements them with the new item of information, if it can do so in a consistent way. This is in line with a view of revision where the information κ with which the agent starts off represents the possible worlds the agent finds most plausible, information not to be given up unless challenged by conflicting new data, and spells out a conservative attitude towards initial beliefs, guided by the desire to preserve them as much as possible.

In the current work we view such a conservative attitude as one among many that an agent can have towards its initial beliefs. By varying the postulate responsible for enforcing this attitude we are able to axiomatize revision operators that embody a wider range of attitudes towards prior information, and characterize these operators in terms of the types of preorders they induce on the set of possible worlds. To illustrate these principles we provide concrete operators, constructed using two ingredients: a notion of *distance* between possible worlds and a *comparison function* that ranks possible worlds depending on the initial beliefs. We also show, in each case, how these operators fit into the landscape of new postulates introduced. Without the theoretical apparatus of the new postulates, the concrete operators put forward would be merely classified as deviant, since they do not satisfy the traditional blend of *Inclusion* and *Vacuity*. But through the present analysis they can be viewed as encoding distinct and characterizable stances an agent can take towards its beliefs.

Moreover, we argue that our insights can be put to use in the current research stream of *fragment-based* revision (Creignou *et al.* 2014; Delgrande and Peppas 2015; Zhuang *et al.* 2017; Delgrande *et al.* 2018), which seeks to understand belief revision in more applied formalisms. The main motivation behind this line of research is that there is much to be gained computationally if we assume that an agent states its beliefs in a simpler language, e.g., a restricted fragment of propositional logic, and that by revising the agent is able to stay within this fragment. However, it turns out that such a mild assumption (i.e., that revision returns a knowledge base expressible in a certain fragment) clashes systematically with commonly accepted properties, including the combined version of the aforementioned *In-*

clusion and *Vacuity* postulates: their conjunction puts certain demands on the underlying language (e.g., that the conjunction of κ and μ is always expressible in it), which are not met in all scenarios that interest us. Thus, there is an unexpected payoff in looking, as we do here, at weaker versions of the standard postulates and their semantic characterizations.

Considering alternatives to the classical revision postulates has a distinguished history, going as far back as the original publications in the field (Gärdenfors 1988; Katsuno and Mendelzon 1992a; Hansson 1999; Herzig and Rifi 1999; Velázquez-Quesada 2017). However, we believe that a systematic analysis of the intuition underlying KM postulate R_2, as we perform here, sheds new light on familiar topics.

Preliminaries

We assume a finite set $\mathcal{P}$ of propositional atoms, from which the set Prop of propositional formulas is generated using the usual propositional connectives. A propositional knowledge base κ is a finite set of propositional formulas, which we typically identify with the conjunction of its formulas. Thus, we think of a knowledge base κ as a single formula, i.e., $\bigwedge_{\mu\in\kappa}\mu$. The set of all propositional knowledge bases is 2^{Prop}. The universe $\mathcal{U}$ is the set of all possible interpretations (also called *possible worlds*) for formulas in Prop. The models of a propositional formula μ are the interpretations which satisfy it, and we write $[\mu]$ (respectively, $[\kappa]$) for the set of models of μ (respectively, for $\bigcap_{\mu\in\kappa}[\mu]$). If there is no danger of ambiguity, we write models as words where the letters are the atoms assigned to true, e.g., $\{\{a,b\},\{b,c\}\}$ is written as $\{ab,bc\}$; hence, for instance, $[a\vee b]=\{a,b,ab\}$. If μ_1 and μ_2 are propositional formulas, we say that μ_1 entails μ_2, written $\mu_1\models\mu_2$, if $[\mu_1]\subseteq[\mu_2]$, and that they are equivalent, written $\mu_1\equiv\mu_2$, if $[\mu_1]=[\mu_2]$. A formula μ (a knowledge base κ) is consistent if $[\mu]\neq\emptyset$ ($[\kappa]\neq\emptyset$), and *complete* if it has exactly one model. The set of consistent knowledge bases is $2^{\mathsf{Prop}}_{\mathsf{cons}}$. If κ is a propositional knowledge base, then *the dual* $\overline{\kappa}$ *of* κ is obtained by replacing every literal l appearing in κ with its negation. If w is an interpretation, the dual interpretation $\overline{w}$ is $\mathcal{P}\setminus w$. If W is a set of interpretations, its dual $\overline{W}$ is defined as $\{\overline{w}\mid w\in W\}$.

Example 1. If $\mathcal{P}=\{a,b,c\}$ and $\kappa=\{a,a\to b\}$, then $\overline{\kappa}=\{\neg a,\neg a\to\neg b\}$. We have that $[\kappa]=\{ab,abc\}$, the dual of the interpretation ab is $\overline{ab}=c$ and $\overline{[\kappa]}=\{c,\emptyset\}$.

In Example 1 we obtain that $\overline{[\kappa]}=[\overline{\kappa}]$. Though we do not provide a formal proof, we mention here that this holds more generally, i.e., for any $\kappa\in 2^{\mathsf{Prop}}$, it holds that $\overline{[\kappa]}=[\overline{\kappa}]$.

If $\mathcal{M}$ is a set, then $Bin(\mathcal{M})$ is the set of binary relations on $\mathcal{M}$. We write $<$ for the strict part of $\leq$, i.e., $x<x'$ if $x\leq x'$ and $x'\not\leq x$; moreover, $x\approx x'$ if $x\leq x'$ and $x'\leq x$. The *$\leq$-minimal elements of $\mathcal{M}$ with respect to* $\leq$ are defined as $\min_{\leq}\mathcal{M}=\{x\in\mathcal{M}\mid\nexists x'\in\mathcal{M}\text{ such that }x'<x\}$. An *assignment from $\mathcal{M}_1$ to $\mathcal{M}_2$* is a function $\alpha\colon\mathcal{M}_1\to Bin(\mathcal{M}_2)$. We write $\leq_\kappa$ instead of $\alpha(\kappa)$ if there is no danger of ambiguity. If W is a set of interpretations, we denote by φ_W a propositional formula such that $[\varphi_W]=W$. If w_1 and w_2 are two interpretations, $\varphi_{1,2}$ is a propositional formula such that $[\varphi_{1,2}]=\{w_1,w_2\}$. A renaming ρ is a permutation on the set $\mathcal{P}$ which, applied to a formula μ, yields a formula $\rho(\mu)$ whose atoms are replaced according to ρ.

Example 2. If ρ is a renaming such that $\rho(a)=b$, $\rho(b)=c$ and $\rho(c)=a$, then $\rho(\neg a\wedge(b\to c))=\neg b\wedge(c\to a)$.

Revision: axioms and characterizations

A propositional revision operator is a function $\circ\colon 2^{\mathsf{Prop}}_{\mathsf{cons}}\times\mathsf{Prop}\to\mathsf{Prop}$. The intention is that $\kappa\circ\mu$ encodes changes brought to existing held beliefs κ such that new, trusted information μ is accepted. A sensible revision operator is expected to resolve any inconsistencies between κ and μ and to satisfy other rationality criteria, presented below.

Basic postulates. If $\kappa,\kappa_1,\kappa_2\in 2^{\mathsf{Prop}}_{\mathsf{cons}}$ and $\mu,\mu_1,\mu_2\in\mathsf{Prop}$, we first single out the following core set of axioms:

- (R_1) $\kappa\circ\mu\models\mu$.
- (R_2) If μ is consistent, then $\kappa\circ\mu$ is consistent.
- (R_3) If $\kappa_1\equiv\kappa_2$ and $\mu_1\equiv\mu_2$, then $\kappa_1\circ\mu_1\equiv\kappa_2\circ\mu_2$.
- (R_4) $(\kappa\circ\mu_1)\wedge\mu_2\models\kappa\circ(\mu_1\wedge\mu_2)$.
- (R_5) If $(\kappa\circ\mu_1)\wedge\mu_2$ is consistent, then $\kappa\circ(\mu_1\wedge\mu_2)\models(\kappa\circ\mu_1)\wedge\mu_2$.
- (R_N) $\rho(\kappa\circ\mu)\equiv\rho(\kappa)\circ\rho(\mu)$.

Postulates R_{1-5} form part of the standard set of KM postulates (Katsuno and Mendelzon 1992b), saying that a revision operator incorporates new information μ (R_1), returns a consistent output if μ is consistent (R_2), performs its task irrespective of how beliefs are written down (R_3) and satisfies some coherence constraints when the revision formula is varied (R_{4-5}).[1] We have found it suitable to add here a neutrality axiom R_N, requiring that a revision operator does not favor propositional atoms based solely on their names. This idea is expressed by requiring the revision output to be invariant under a renaming ρ of atoms, and, in conjunction with the irrelevance of syntax postulate R_3, it is perhaps natural to expect it from any revision operator. The inspiration for postulate R_N is the social choice literature (see (Rothe 2015), Chapter 4) and, though it has appeared in belief change before, under various guises (Herzig and Rifi 1999; Marquis and Schwind 2014; Haret *et al.* 2016), neutrality usually goes unstated in standard presentations of revision.

A revision operator is *basic* if it satisfies postulates R_{1-5} and *neutral* if it satisfies postulate R_N. We will typically assume that all revision operators we work with are basic.

Basic assignments. Reflection on postulates R_{1-5} reveals that an operator $\circ$ satisfying them chooses among models of μ and, in doing so, behaves as if it has preferences over units of information. Formally, this is cashed out by assigning to each consistent knowledge base κ in $2^{\mathsf{Prop}}_{\mathsf{cons}}$ a binary relation $\leq_\kappa$ on interpretations in $\mathcal{U}$: to revise κ by μ, then, becomes equivalent to choosing the best models of μ in $\leq_\kappa$. And, in the same way that revision operators are expected to satisfy a set of basic properties (postulates R_{1-5}), rankings on $\mathcal{U}$ must

[1] Note that R_2 as we show it does not coincide with KM postulate R_2. The KM postulate shows up in the latter part of this section.

satisfy a set of properties, to be introduced in the following, that are conducive to rational choice.

First of all, let us extend our notion of renamings to cover interpretations. Thus, if w is an interpretation and ρ is a renaming of atoms, then $\rho(w)$ is an interpretation obtained by replacing every atom p in w with $\rho(p)$. If $\mathcal{M}$ is a set of interpretations, then $\rho(\mathcal{M}) = \{\rho(w) \mid w \in \mathcal{M}\}$.

Example 3. Take $\mathcal{M} \subseteq \mathcal{U}$ such that $\mathcal{M} = \{a, ab, bc\}$ and a renaming ρ such that $\rho(a) = b$, $\rho(b) = c$ and $\rho(c) = a$. We get $\rho(\mathcal{M}) = \{\rho(a), \rho(ab), \rho(bc)\} = \{b, bc, ac\}$.

For an assignment from consistent knowledge bases in $2^{\mathsf{Prop}}_{\mathsf{cons}}$ to binary relations on $\mathcal{U}$, a revision operator $\circ$ and $\kappa, \kappa_1, \kappa_2 \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$, $w_1, w_2 \in \mathcal{U}$, we look at these properties:

(o_1) $\leq_\kappa$ is reflexive.

(o_2) $\leq_\kappa$ is transitive.

(o_3) If $\kappa_1 \equiv \kappa_2$, then $\leq_{\kappa_1} = \leq_{\kappa_2}$.

(o_4) $\leq_\kappa$ is total.

(o_5) $[\kappa \circ \mu] = \min_{\leq_\kappa}[\mu]$, for any propositional formula μ.

(o_N) $w_1 \leq_\kappa w_2$ iff $\rho(w_1) \leq_{\rho(\kappa)} \rho(w_2)$.

An assignment is *basic* if it satisfies properties o_{1-4}. Notice that properties o_{1-2} imply that $\leq_\kappa$ is a preorder on $\mathcal{U}$. Adding property o_4 makes $\leq_\kappa$ total, and o_3 adds an independence of syntax aspect to the assignment. If, on top of this, $\leq_\kappa$ satisfies o_5, we say that the assignment α *represents the revision operator* $\circ$ (and that $\circ$ *is represented by* α). We call an assignment *neutral* if it satisfies property o_N. The overloading of notation ('basic', 'neutral') is intentional: properties $\mathsf{o}_{1-5,\,\mathsf{N}}$ define a class of rankings on interpretations that fully characterize revision operators satisfying axioms $\mathsf{R}_{1-5,\,\mathsf{N}}$. The following results make this precise.

Theorem 1. A revision operator satisfies postulates R_{1-5} iff there exists an assignment from $2^{\mathsf{Prop}}_{\mathsf{cons}}$ to $\mathcal{U}$ representing it such that, for any $\kappa \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$, the preorder $\leq_\kappa$ satisfies properties o_{1-5}.

This result follows from existing work on revision and can be extracted, for example, from the proof of the representation result in (Katsuno and Mendelzon 1992b). Keeping in mind that for any $\mu \in \mathsf{Prop}$ and renaming ρ it holds that $[\rho(\mu)] = \rho([\mu])$, we also get the following result.

Theorem 2. If $\circ$ is a basic revision operator and α is an assignment representing it, then $\circ$ satisfies axiom R_N iff α satisfies property o_N.

Proof. Recall, first, that we denote by $\varphi_{1,2}$ a propositional formula such that $[\varphi_{1,2}] = \{w_1, w_2\}$. ("$\Rightarrow$") Take a basic revision operator $\circ$ that satisfies axiom R_N, and the assignment which represents it. Take w_1 and w_2 and suppose, first, that $w_1 \leq_\kappa w_2$. Then $w_1 \in [\kappa \circ \varphi_{1,2}]$, and hence $\rho(w_1) \in \rho([\kappa \circ \varphi_{1,2}])$. We get that $\rho(w_1) \in [\rho(\kappa \circ \varphi_{1,2})]$ and by postulate R_N it follows that $\rho(w_1) \in [\rho(\kappa) \circ \rho(\varphi_{1,2})]$. This implies that $\rho(w_1) \in \min_{\leq_{\rho(\kappa)}}[\rho(\varphi_{1,2})]$, which implies that $\rho(w_1) \in \min_{\leq_{\rho(\kappa)}} \rho([\varphi_{1,2}])$. Thus, $\rho(w_1) \leq_{\rho(\kappa)} \rho(w_2)$. Conversely, suppose that $\rho(w_1) \leq_{\rho(\kappa)} \rho(w_2)$. This implies that $\rho(w_1) \in \min_{\leq_{\rho(\kappa)}} \rho([\varphi_{1,2}])$. By postulate R_N, we get that $w_1 \leq_\kappa w_2$. ("$\Leftarrow$") Take an assignment that satisfies property o_N and the revision operator $\circ$ represented by it. We have that $[\rho(\kappa \circ \mu)] = \rho([\kappa \circ \mu]) = \rho(\min_{\leq_\kappa}[\mu])$, and $[\rho(\kappa) \circ \rho(\mu)] = \min_{\leq_{\rho(\kappa)}}[\rho(\mu)] = \min_{\leq_{\rho(\kappa)}} \rho([\mu])$. We show that $[\rho(\kappa \circ \mu)] = [\rho(\kappa) \circ \rho(\mu)]$ by double inclusion. Take, first, $\rho(w_1) \in \rho(\min_{\leq_\kappa}[\mu])$, for $w_1 \in \min_{\leq_\kappa}[\mu]$, and $\rho(w_2) \in \rho([\mu])$, for $w_2 \in [\mu]$. Then $w_1 \leq_\kappa w_2$ and by o_N we get that $\rho(w_1) \leq_{\rho(\kappa)} \rho(w_2)$, which implies that $\rho(w_1) \in \min_{\leq_{\rho(\kappa)}} \rho([\mu])$. This shows that $[\rho(\kappa \circ \mu)] \subseteq [\rho(\kappa) \circ \rho(\mu)]$. Next, take $\rho(w_1) \in \min_{\leq_{\rho(\kappa)}} \rho([\mu])$, for $w_1 \in [\mu]$, and $\rho(w_2) \in \rho([\mu])$, for $w_2 \in [\mu]$. We get that $\rho(w_1) \leq_{\rho(\kappa)} \rho(w_2)$, which by o_N implies that $w_1 \leq_\kappa w_2$. Thus, $w_1 \in \min_{\leq_\kappa}[\mu]$ and hence $\rho(w_1) \in \rho(\min_{\leq_\kappa}[\mu])$. □

Theorem 1 tells us that an agent revising beliefs along the lines of postulates R_{1-5} behaves as if it ranks interpretations in $\mathcal{U}$ in a total preorder $\leq_\kappa$ that depends on initial beliefs κ, and always picks the minimal models of μ according to $\leq_\kappa$. Such an agent, then, behaves like a rational agent, in the sense of rational choice theory (Sen 1984; French 1988), choosing the best elements from a given menu of options: the menu, here, would be the models of μ, i.e., the possible worlds the agent is allowed to believe in light of new information, while the best elements are decided with reference to $\leq_\kappa$. Thus, a revision operator can be seen as a *choice function* over sets of interpretations: for instance, postulates R_{5-6} are equivalent to what is known as the *Weak Axiom of Revealed Preference (WARP)* and, taken together, postulates R_{1-5} characterize choice functions *rationalizable* by total preorders. Accordingly, Theorem 1 aligns with standard choice theoretic results (Arrow 1959; Sen 1984; Moulin 1991). That a similar mathematical formalism underlies both belief revision and rational choice is, in itself, not a new insight, the topic having been studied before under various guises (Rott 2001; Bonanno 2009).

Theorem 2 adds invariance of $\leq_\kappa$ under renamings. The ranking $\leq_\kappa$ is usually thought of as a plausibility ranking, i.e., the assessment of an agent believing κ as to which possible worlds are more or less plausible.

Example 4. A doctor knows that the patient has been diagnosed with asthma (a), finds out that the patient is suffering from shortness of breath (b) and infers that chest pain (c) is also present (the two often go together in asthma). In other words, the doctor has initial information $\kappa = a$ and a plausibility ranking $\leq_\kappa$ over possible worlds, depicted in Figure 1. The doctor then revises by $\mu = b$ and settles on a possible state of affairs that is most plausible according to their plausibility ranking $\leq_\kappa$, i.e., $[\kappa \circ \mu] = \min_{\leq_\kappa}[\mu] = \{abc\}$. Note that the doctor, in this case, believes that the situation represented by $a \wedge b \wedge c$ is more likely than $a \wedge b \wedge \neg c$.

One way of thinking of postulates $\mathsf{R}_{1-5,\,\mathsf{N}}$ is that they axiomatize neutral total preorders on interpretations. These preorders nominally depend on κ, but nothing in postulates $\mathsf{R}_{1-5,\,\mathsf{N}}$ touches on how models of κ should influence these preorders. In other words, there is as yet no information about the attitude of an agent towards its initial epistemic state, and postulates $\mathsf{R}_{1-5,\,\mathsf{N}}$ are consistent with arbitrary attitudes towards κ. How should the models of κ stand in

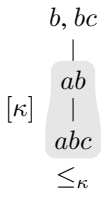


Figure 1: Revision scenario of Example 4, showing the preorder $\leq_\kappa$ on the basis of which the revision result is constructed: only models of $\mu = b$ are depicted; the models of κ among this set are highlighted in grey.

relation to all other interpretations? In Example 4 we catch a glimpse of one possible answer: the agent there starts off with some information κ and differentiates among the possible worlds consistent with κ: some of these interpretations are, *a priori*, more plausible than others. Still, as a whole, models of κ are more plausible than any *other* interpretations consistent with the new information μ—the agent is biased towards the possible worlds consistent with κ, an attitude which fits with the idea of κ being the agent's *belief*. Are there, now, other ways of arranging the models of κ in $\leq_\kappa$, ways that span the space of possible such attitudes? We study this question through the lens of additional axioms.

Attitudes towards initial beliefs. Let $\kappa, \kappa' \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$, $\mu \in \mathsf{Prop}$, and consider the following postulates:

- (R_6) If $\kappa \wedge \mu$ is consistent, then $\kappa \wedge \mu \models \kappa \circ \mu$.
- (R_7) If $\kappa \wedge \mu$ is consistent, then $\kappa \circ \mu \models \kappa \wedge \mu$.
- (R_8) If $\mu \models \overline{\kappa}$, then $\kappa \circ \mu \equiv \mu$.
- (R_9) If $\mu \not\models \overline{\kappa}$, then $(\kappa \circ \mu) \wedge \overline{\kappa}$ is inconsistent.
- (R_{10}) If κ' is complete and $\kappa' \models \kappa \circ \mu$, then $\kappa' \models (\kappa \vee \kappa') \circ \mu$.

Each of these postulates encodes a particular type of attitude towards initial beliefs, and they are intended to be thought of in conjunction with the basic set of postulates R_{1-5}. Some clarification is in order. Postulate R_6 models an agent that incorporates all information in $\kappa \wedge \mu$, and possibly extends this to cover more ground. Postulate R_7 models an agent that reserves itself the right to drop information from κ if it so sees fit, even if that information is consistent with μ: we may imagine this is done on the basis of certain preferences over the information encoded by κ, i.e., the agent is partial towards certain parts of κ to the detriment of others. Taken together, postulates R_{6-7} imply that $\kappa \circ \mu$ is equivalent to $\kappa \wedge \mu$, when $\kappa \wedge \mu$ is consistent. This property models an agent who wants to preserve as much of κ as it can, and does not have any bias towards either of the models of κ. Postulate R_6 can be equated with the *Inclusion* postulate in the AGM formulation and R_7 corresponds to *Vacuity* (Fermé and Hansson 2018), while in the KM axiomatization R_6 and R_7 are packaged together in one postulate (i.e., KM postulate R_2) and presented alongside R_{1-5} as the default set of rational properties for revision (Katsuno and Mendelzon 1992b).

Postulates R_{8-9} focus on the dual knowledge base $\overline{\kappa}$ obtained by replacing every literal in κ with its negated version. If κ is a conjunction of literals, or if it is a complete (i.e., with exactly one model) formula, then $\overline{\kappa}$ will be a formula whose models are complements of the models of κ.

Example 5. If $\mathcal{P} = \{a, b, c\}$ and $\kappa = \{a \wedge b\}$ is a knowledge base over the alphabet $\mathcal{P}$, then $\overline{\kappa} = \{\neg a \wedge \neg b\}$, and we have that $[\kappa] = \{ab, abc\}$, while $[\overline{\kappa}] = \{\emptyset, c\}$

Thus, if κ is reasonably specific (e.g., is a conjunction of literals), then $\overline{\kappa}$ can be thought of as a point of view opposite to that of κ. Of course, this analogy breaks down if κ is a knowledge base such as $a \vee b$, or $a \vee \neg a$, where κ and $\overline{\kappa}$ share models. Our point here is simply that there are situations in which it makes sense to view κ and $\overline{\kappa}$ as embodying opposing stances, and in which one would like to place bounds on the revision function in terms of how it treats information encoded by the opposing point of view $\overline{\kappa}$: this is the case if the agent has, or is required to have, a definite opinion on every item from an agenda, as is typically the case in Judgment Aggregation (Endriss 2016), and when κ will be a complete formula; or if κ is a 'vivid' knowledge base (Levesque 1986), or encodes something like an agent's preferred bundle from a set of available items, in which case κ can be required to be a conjunction of literals.

Postulate R_8 says that if κ undergoes revision by a formula μ embodying such an adverse perspective, then the agent must adopt μ: in other words, the agent has no room for maneuvering towards a more amenable middle ground. Such a revision policy makes more sense when considered alongside postulate R_9, which specifies that if the agent has the option of believing states of affairs *not* compatible with $\overline{\kappa}$, it should wholeheartedly adopt those as the most plausible stance. Taken together, postulates R_{8-9} inform the agent to believe states of affairs compatible with $\overline{\kappa}$ only if it has no other choice in the matter: the models of $\overline{\kappa}$ should be part of a viewpoint one is willing to accept only as a last resort. Postulate R_{10} is best understood through an example.

Example 6. An agent intends to go to an art museum, the beach and a concert, i.e., $\kappa = \{a \wedge b \wedge c\}$. The agent then learns that it only has time for one of these activities and chooses the art museum, i.e., $\kappa \circ \mu \equiv a \wedge \neg b \wedge \neg c$. If the agent's initial intentions were less specific, for instance that it would either go to all three places or only to the art museum (i.e., $\kappa = \{(a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c)\}$), then, faced with the same new information μ, $a \wedge \neg b \wedge \neg c$ should still feature as one of its most preferred options.

A clearer view of postulates R_{6-10} emerges when looking at how they place the models of κ in a total preorder $\leq_\kappa$ ($\kappa, \kappa' \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$, $w_1, w_2, w' \in \mathcal{U}$):

- (o_6) If $w_1 \in [\kappa]$, then $w_1 \leq_\kappa w_2$.
- (o_7) If $w_1 \in [\kappa]$ and $w_2 \notin [\kappa]$, then $w_1 <_\kappa w_2$.
- (o_8) If $w_1 \in [\overline{\kappa}]$, then $w_2 \leq_\kappa w_1$.
- (o_9) If $w_1 \in [\overline{\kappa}]$ and $w_2 \notin [\overline{\kappa}]$, then $w_2 <_\kappa w_1$.
- (o_{10}) If $w' \leq_\kappa w$ and $[\kappa'] = \{w'\}$, then $w' \leq_{\kappa \vee \kappa'} w$.

Properties o_{6-10} turn out to characterize axioms R_{6-10} on the semantic level, as per the following representation result.

Theorem 3. If $\circ$ is a revision operator satisfying postulates R_{1-5} and α is an assignment from $2^{\mathsf{Prop}}_{\mathsf{cons}}$ to $\mathcal{U}$ which satisfies properties o_{1-5} then, for any $\kappa \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$ and $\mu \in \mathsf{Prop}$, the following equivalences hold:

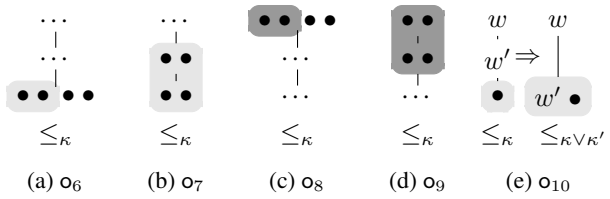
(1) $\circ$ satisfies axiom R_6 iff $\leq_\kappa$ satisfies property o_6;
(2) $\circ$ satisfies axiom R_7 iff $\leq_\kappa$ satisfies property o_7;
(3) $\circ$ satisfies axiom R_8 iff $\leq_\kappa$ satisfies property o_8;
(4) $\circ$ satisfies axiom R_9 iff $\leq_\kappa$ satisfies property o_9;
(5) $\circ$ satisfies axiom R_{10} iff $\leq_\kappa$ satisfies property o_{10}.

Proof. Recall that we denote by $\varphi_{1,2}$ a propositional formula such that $[\varphi_{1,2}] = \{w_1, w_2\}$. For equivalence 1, we show each direction in turn. ("$\Rightarrow$") Take, first, an assignment α satisfying property o_6, and the revision operator $\circ$ represented by it. Let us assume that $\kappa \wedge \mu$ is consistent, and show that for any $w \in [\kappa \wedge \mu]$, it holds that $w \in [\kappa \circ \mu]$ as well. By property o_5, this is equivalent to showing that $w \in \min_{\leq_\kappa}[\mu]$. Take an arbitrary interpretation $w' \in [\mu]$. Since $w \in [\kappa]$, we can apply property o_6 to get that $w \leq_\kappa w'$. Hence $w \in \min_{\leq_\kappa}[\mu]$. ("$\Leftarrow$") Take a basic revision operator $\circ$ satisfying R_6. and the assignment α which represents it. To show that $\leq_\kappa$ satisfies property o_6, take two interpretations w_1 and w_2 such that $w_1 \in [\kappa]$. Then, by axiom R_6, we have that $\kappa \wedge \varphi_{1,2} \models \kappa \circ \varphi_{1,2}$. By property o_5, it holds that $[\kappa \circ \varphi_{1,2}] = \min_{\leq_\kappa}[\varphi_{1,2}]$ and, since $w_1 \in [\kappa \wedge \varphi_{1,2}]$, it follows that $w_1 \in \min_{\leq_\kappa}[\varphi_{1,2}]$. Thus, $w_1 \leq_\kappa w_2$.

For equivalence 2, we show again each direction in turn. ("$\Rightarrow$") Take an assignment α satisfying property o_7 and the revision operator $\circ$ represented by it. Let us assume that $\kappa \wedge \mu$ is consistent. Take $w \in [\kappa \circ \mu]$, and suppose $w \notin [\kappa \wedge \mu]$. By property o_5, we have that $[\kappa \circ \mu] = \min_{\leq_\kappa}[\mu]$, and hence $w \in [\mu]$. Thus, the fact that $w \notin [\kappa \wedge \mu]$ implies that $w \notin [\kappa]$. But, by assumption, it holds that $[\kappa \wedge \mu] \neq \emptyset$. Thus, there exists $w' \in [\kappa \wedge \mu]$ and, by property o_7, it follows that $w' <_\kappa w$. But we also have that $w \in \min_{\leq_\kappa}[\mu]$, which implies that it cannot be the case that $w' <_\kappa w$, which is a contradiction. ("$\Leftarrow$") Take a basic revision operator $\circ$ satisfying R_7 and the preorder $\leq_\kappa$ that represents it. To show that $\leq_\kappa$ satisfies property o_7, take $w_1 \in [\kappa]$ and $w_2 \notin [\kappa]$. We then have that $\kappa \wedge \varphi_{1,2}$ is consistent and hence, by axiom R_7, that $\kappa \circ \varphi_{1,2} \models \kappa \wedge \varphi_{1,2}$. Since $[\kappa \circ \varphi_{1,2}]$ is, by axioms R_{1-2}, a non-empty subset of $[\varphi_{1,2}] = \{w_1, w_2\}$, we have that at least one of w_1 and w_2 is in $[\kappa \circ \varphi_{1,2}]$. Notice, now, that we cannot have $w_2 \in [\kappa \circ \varphi_{1,2}]$, since it would follow that $w_2 \in [\kappa \wedge \varphi_{1,2}]$ and $w_2 \in [\kappa]$, which is a contradiction. Thus, $[\kappa \circ \varphi_{1,2}] = \{w_1\}$. Since $\leq_\kappa$ represents $\circ$, we have by property o_5 that $[\kappa \circ \varphi_{1,2}] = \min_{\leq_\kappa}[\varphi_{1,2}]$. It follows that $w_1 <_\kappa w_2$.

Equivalences 3 and 4 are analogous to 1 and 2, respectively. For equivalence 5, assume first that axiom R_{10} holds, and take interpretations w and w' and a knowledge base κ' such that $w' \leq_\kappa w$ and $[\kappa'] = \{w'\}$. To show that $w' \leq_{\kappa \vee \kappa'} w$, we must show that $w' \in [(\kappa \vee \kappa') \circ \varphi_{w,w'}]$, where $\varphi_{w,w'}$ is a formula such that $[\varphi_{w,w'}] = \{w, w'\}$. This follows immediately by applying axiom R_{10}. Conversely, suppose $[\kappa'] = \{w'\}$, and take $w \in [\kappa \circ \mu]$. Then, we get that $w' \leq_\kappa w$, and we can apply property o_{10} to derive the conclusion. □

Theorem 3 is better understood through an illustration of how such preorders treat models of κ. Property o_6 says that models of κ are minimal elements in κ, i.e., the agent considers possible worlds satisfying its beliefs among the most plausible possible worlds, though possibly not uniquely so (Figure 2-(a)). Property o_7 states that there are no countermodels of κ more plausible than the models of κ, but the models of κ themselves may not be equally plausible (Figure 2-(b)). Properties o_{8-9} say that models of the dual knowledge base κ are the least plausible interpretations in $\leq_\kappa$ (Figure 2-(c,d)), while property o_{10} says that if w' is more plausible than w when the initial beliefs are κ, then w' would still be more plausible than w if it were part of the initial beliefs (Figure 2-(e)).



(a) o_6 (b) o_7 (c) o_8 (d) o_9 (e) o_{10}

Figure 2: Schematic view of prototypical preorders satisfying each of the properties o_{6-9}; models of κ are in the light gray area, models of $\overline{\kappa}$ are in the dark gray area.

Together, properties o_{1-7} define what is more commonly known as a *faithful assignment*, placing all and only models of κ on the lowest level of $\leq_\kappa$. This corresponds to an agent that holds its initial beliefs to be the most plausible states of affairs (Katsuno and Mendelzon 1992b). Consequently, Theorem 1 plus equivalences 1-2 from Theorem 3 make up the classical representation result for belief revision operators (Katsuno and Mendelzon 1992b). Here we have opted for a more fine-grained approach to the placement of models of κ in $\leq_\kappa$, which allows a more diverse representation of the different types of attitudes an agent can have towards initial beliefs. Though operators that do not satisfy the classical KM postulate R_2 have been considered before (Ryan 1996; Benferhat *et al.* 2005), the idea that such deviations correspond to possible epistemic attitudes and can be axiomatized is, to the best of our knowledge, new.

Indifference to already held beliefs. One particular consequence of weakening the KM axiom R_2 (axioms R_{6-7} in the current context) is that the following property is not guaranteed to hold anymore:

($\mathsf{R}_{\mathsf{IDF}}$) $\kappa \circ \kappa \equiv \kappa$.

This property, called here $\mathsf{R}_{\mathsf{IDF}}$ (for *indifference to already held beliefs*), says that revising with information the agent already believes does not change the agent's epistemic state. More generally, the KM standard set of postulates implies that revising by any formula μ such that $\kappa \models \mu$ results in κ. It quickly becomes apparent that axiom R_6 implies $\mathsf{R}_{\mathsf{IDF}}$,[2] but R_7 does not. Thus, if an agent is allowed to rank models of κ unequally, then $\mathsf{R}_{\mathsf{IDF}}$ is not guaranteed to hold.

Example 7. Consider the knowledge base $\kappa = \{a \vee b\}$ and a revision operator that satisfies axiom R_7, and which orders

[2]The converse is not true: $\mathsf{R}_{\mathsf{IDF}}$ enforces only that models of κ are equally plausible, but not where they are placed in $\leq_\kappa$.

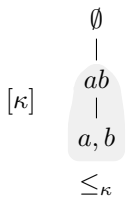


Figure 3: $[\kappa] = \{a, b, ab\}$, but $[\kappa \circ \kappa] = \{a, b\}$

the models of κ as in Figure 3. We get that $[\kappa \circ \kappa] = \{a, b\}$, i.e., $\kappa \circ \kappa \equiv (a \leftrightarrow \neg b)$.

What this fact points to is a more graded view of what it means to believe κ. Thus, an agent might have a certain threshold of plausibility, along the lines of what is known in epistemology as *the Lockean thesis* (Foley 1993), according to which it calibrates its beliefs: anything above the threshold counts as part of the belief κ and anything below counts as disbelief. This then leaves open the possibility that the agent assigns different degrees of plausibility to states of affairs it counts as part of its belief κ: indeed, this is the point of view we endorse here. This is in contrast to more standard approaches, which consider that an agent assigns equal degrees of plausibility to all items of its belief. Thus, new information which confirms an agent's belief might have the effect of *reinforcing* parts that are given more plausibility at the expense of parts that are given less, and this is the kind of phenomenon we take to be modeled by Example 7.

What would be worrying would be a revision policy that makes an agent cycle between different viewpoints when confronted repeatedly with the same type of information: we will see that for revision operators satisfying R_7 this concern is unwarranted, but we must introduce some new notation. We write κ^i for the knowledge base obtained by revising κ by itself an i number of times. Thus, $\kappa^0 = \kappa$ and $\kappa^{i+1} = \kappa^i \circ \kappa$. Consider now the following property:

($\mathsf{R}_{\mathsf{STB}}$) There is $n \geq 1$ such that $\kappa^m \equiv \kappa^n$, for every $m \geq n$.

We say that a revision operator $\circ$ is *stable* if it satisfies property $\mathsf{R}_{\mathsf{STB}}$. Stability implies that repeated revision by κ ultimately settles (or *stabilizes*) on a set of models that does not change anymore through subsequent revisions by κ. The following result proves relevant to the issue of stability.

Proposition 4. If a revision operator $\circ$ satisfies axioms R_1 and R_7, then $\kappa^{i+1} \models \kappa^i$.

Proof. By axiom R_1, we have that $\kappa \circ \kappa \models \kappa$, and thus $\kappa^1 \models \kappa^0$. Applying axiom R_7, we have that $(\kappa \circ \kappa) \circ \kappa \models (\kappa \circ \kappa) \wedge \kappa \models \kappa \circ \kappa$. Thus, $\kappa^2 \models \kappa^1$, and it is straightforward to see how this argument is iterated to get the conclusion. □

If the operator $\circ$ also satisfies axiom R_2 (which, here, says that if the revision formula is consistent, then the revision result is also consistent), it follows that if κ is consistent, then κ_i is consistent, for any $i \geq 0$. Thus, combining this fact and Proposition 4, we get that repeated revision by κ leads to a chain of ever more specific knowledge bases, i.e., $\emptyset \subset \cdots \subseteq [\kappa^{i+1}] \subseteq [\kappa^i] \subseteq \cdots \subseteq [\kappa^0]$. Since a knowledge base has a finite number of models, it falls out immediately from this that there must be a point at which further revision by κ does not change anything.

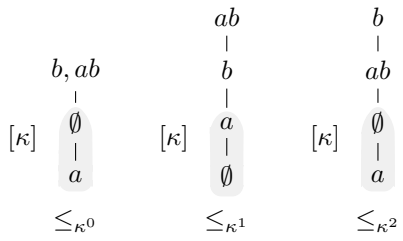


Figure 4: Repeated revision by κ ends up jumping around from $\{a\}$ to $\{\emptyset\}$.

Corollary 5. A basic revision operator $\circ$ satisfying axiom R_7 is stable.

Unfortunately, axioms R_{8-9} do not guarantee stability. Since these axioms require only that the agent places the models of $\overline{\kappa}$ as the least plausible interpretations, it becomes possible that an agent's plausibility ranking does not hold on to a core set of interpretations through successive revisions by κ.

Example 8. Take a knowledge base $\kappa = \{\neg b\}$ and a revision operator satisfying R_{8-9} which orders interpretations as shown in Figure 4. We have that $[\kappa^0] = [\kappa] = \{\emptyset, a\}$, and $[\kappa^1] = [\kappa \circ \kappa] = \{\emptyset\}$ and $[\kappa^2] = [\kappa^1 \circ \kappa] = \{a\}$. By axiom R_3, we get that subsequent revisions by κ jump around between $\{a\}$ and $\{\emptyset\}$, i.e., $[\kappa^3] = \{\emptyset\}$, $[\kappa^4] = \{a\}$, and so on, therefore never settling on a stable answer.

The issue of stability suggests another dimension along which revision operators can be analyzed, with Proposition 5 and Example 8 showing that a revision operator does not satisfy it trivially. Example 8, in particular, shows that there is interplay between the preorders $\leq_\kappa$ and $\leq_{\kappa'}$, where $\kappa' \models \kappa$, which is relevant to the question of whether a revision operator is stable or not. This interplay is reminiscent of issues surrounding iterated revision and kinetic consistency (Darwiche and Pearl 1997; Peppas and Williams 2016) and pursuing it further is worthwhile, though it would take us too far afield of the aims of the current work.

Concrete propositional revision operators

Having characterized revision operators in terms of assignments on interpretations, we ask ourselves what is a natural way to construct such assignments. The usual way of exploiting the insight afforded by Theorem 1 is to use some type of distance d between interpretations, interpreted as a measure of plausibility of one interpretation relative to the other. The distance d is then used to compare interpretations with respect to how plausible (or close) they are with respect to the initial beliefs κ.

Our answer extends this method in a way reminiscent of techniques used to construct belief merging operators (Konieczny *et al.* 2004; Konieczny and Pérez 2011), i.e., by employing *two* main ingredients. The first ingredient is a *distance* between interpretations, defined as a function $d \colon \mathcal{U} \times \mathcal{U} \to \mathbb{R}_+$ such that $d(w_1, w_2) = 0$ iff $w_1 = w_2$

and $d(w_1, w_2) = d(w_2, w_1)$. Given $w \in \mathcal{U}$ and $\kappa \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$ such that $[\kappa] = \{w_1, \ldots, w_n\}$, the *vector of distances from w to κ* is $d(w, \kappa) = (d(w, w_1), \ldots, d(w, w_n))$. If there is no danger of ambiguity we omit commas and simply write $d(w, \kappa)$ as a string of numbers.

Example 9. If $w = a$, $[\kappa] = \{a, b, ab\}$, d is a distance function such that $d(a, a) = 0$, $d(a, b) = 2$ and $d(a, ab) = 1$, then $d(w, \kappa) = (0, 2, 1)$, written as $d(w, \kappa) = (021)$.

We mention here two prominent examples of distance. The first one, called *the drastic distance* d_{D} works by the all-or-nothing rule: $d_{\mathsf{D}}(w_1, w_2) = 0$ if $w_1 = w_2$, and 1 otherwise. The second one is *the Hamming distance* d_{H}, which counts the number of atoms on which two interpretations differ.

The second ingredient is a *comparison function*, which is a family of functions $f \colon \mathbb{R}^n \to \mathbb{R}$ for $n \in \mathbb{N}$, mapping a distance vector $d(w, \kappa)$ to a number and used to compare distance vectors. We write $\overrightarrow{d(w, \kappa)}$ and $\overleftarrow{d(w, \kappa)}$ for the vectors of distances from w to κ ordered in ascending order and descending order, respectively. The lexicographic order between two vectors is denoted by $\leq_{\mathsf{lex}}$. Minimal and maximal elements of $d(w, \kappa)$ are denoted by $\min d(w, \kappa)$ and $\max d(w, \kappa)$, respectively. We define $\sum d(w, \kappa) = \sum_{w_i \in [\kappa]} d(w, w_i)$. The *centrality of w with respect to κ* is $\mathrm{cen}(w, \kappa) = \max d(w, \kappa) - \min d(w, \kappa)$. The *displacement of w with respect to κ* is $\mathrm{dis}(w, \kappa) = \min d(w, \kappa) - \min d(w^*, \kappa)$, where w^* is an interpretation such that $\min d(w^*, \kappa)$ is minimal among all the interpretations w' for which $\mathrm{cen}(w', \kappa) = \mathrm{cen}(w, \kappa)$. Finally, the *agreeability index of w with respect to κ* is defined as $\mathrm{agr}(w, \kappa) = \min\{\min d(w, \kappa),\ \mathrm{cen}(w, \kappa) + \mathrm{dis}(w, \kappa)\}$, while the *disagreeability index of w with respect to κ* is $\mathrm{dagr}(w, \kappa) = n - \mathrm{agr}(w, \overline{\kappa})$, where $n = |\mathcal{P}|$.

Let us now put the two ingredients together. Given a distance d and comparison function f, we write $\leq_\kappa^{d,\,f}$ for the ranking generated using d and f, and $\circ^{d,\,f}$ for the revision operator represented by the assignment generated using d and f, i.e., defined by taking $[\kappa \circ^{d,\,f} \mu] = \min_{\leq_\kappa^{d,\,f}}[\mu]$. Assuming some distance d between interpretations, we will look at the types of rankings defined in Table 1.

$w_1 \leq_\kappa^{d,\,\min} w_2$	iff	$\min d(w_1, \kappa) \leq \min d(w_2, \kappa)$,
$w_1 \leq_\kappa^{d,\,\mathrm{lmin}} w_2$	iff	$\overrightarrow{d(w_1, \kappa)} \leq_{\mathsf{lex}} \overrightarrow{d(w_2, \kappa)}$,
$w_1 \leq_\kappa^{d,\,\mathrm{agr}} w_2$	iff	$\mathrm{agr}(w_1, \kappa) \leq \mathrm{agr}(w_2, \kappa)$,
$w_1 \leq_\kappa^{d,\,\max} w_2$	iff	$\max d(w_1, \kappa) \leq \max d(w_2, \kappa)$,
$w_1 \leq_\kappa^{d,\,\mathrm{lmax}} w_2$	iff	$\overleftarrow{d(w_1, \kappa)} \leq_{\mathsf{lex}} \overleftarrow{d(w_2, \kappa)}$,
$w_1 \leq_\kappa^{d,\,\mathrm{dagr}} w_2$	iff	$\mathrm{dagr}(w_1, \kappa) \leq \mathrm{dagr}(w_2, \kappa)$,
$w_1 \leq_\kappa^{d,\,\mathrm{sum}} w_2$	iff	$\sum d(w_1, \kappa) \leq \sum d(w_2, \kappa)$,

Table 1: Types of rankings, defined for a distance d

We illustrate these operators with the following examples.

Example 10. Take $\kappa = \{\neg a \vee \neg b \vee (a \wedge b \wedge c)\}$, for which we get that $[\kappa] = \{\emptyset, a, b, abc\}$. For the interpretation $w = \emptyset$, we get that $\overrightarrow{d_{\mathsf{H}}(w, \kappa)} = (0113)$, $\overleftarrow{d_{\mathsf{H}}(w, \kappa)} = (3110)$, $\min d_{\mathsf{H}}(w, \kappa) = 0$, $\max d_{\mathsf{H}}(w, \kappa) = 3$ and $\sum d_{\mathsf{H}}(w, \kappa) = 5$. The distances and computed aggregation functions for each interpretation are depicted in Table 2. Notice how the models of κ are distributed when the interpretations are ranked according to the different comparison functions used: we have $\emptyset \approx_\kappa^{\mathsf{H},\,\min} a$, since $\min d_{\mathsf{H}}(a, \kappa) = \min d_{\mathsf{H}}(a, \kappa) = 0$, but $\emptyset <_\kappa^{\mathsf{H},\,\mathrm{lmin}} a$, since $(0113) \leq_{\mathsf{lex}} (0122)$. Also, we have that $c <_\kappa^{\mathsf{H},\,\max} abc$, $c <_\kappa^{\mathsf{H},\,\mathrm{lmax}} abc$ and $ab <_\kappa^{\mathsf{H},\,\mathrm{sum}} abc$, i.e., models of κ are not minimal in $\leq_\kappa^{\mathsf{H},\,\max}$, $\leq_\kappa^{\mathsf{H},\,\mathrm{lmax}}$ and $\leq_\kappa^{\mathsf{H},\,\mathrm{sum}}$. In particular, $\leq_\kappa^{\mathsf{H},\,\max}$ makes the models of $\overline{\kappa}$ (i.e., abc, bc, ac and $\emptyset$) the least plausible interpretations.

	$\emptyset$	a	b	abc	$\overrightarrow{d_{\mathsf{H}}(w,\kappa)}$	$\overleftarrow{d_{\mathsf{H}}(w,\kappa)}$	min	max	$\sum$
$\emptyset$	0	1	1	3	(0113)	(3110)	0	3	5
a	1	0	2	2	(0122)	(2210)	0	2	5
b	1	2	0	2	(0122)	(2210)	0	2	5
c	1	2	2	2	(1222)	(2221)	1	2	7
ab	2	1	1	1	(1112)	(2111)	1	2	5
ac	2	1	3	1	(1123)	(3211)	1	3	7
bc	2	3	1	1	(1123)	(3211)	1	3	7
abc	3	2	2	0	(0223)	(3220)	0	3	7

Table 2: Table of Hamming distances for κ from Example 10

The agreement and disagreement operators ($\circ^{d,\,\mathrm{agr}}$ and $\circ^{d,\,\mathrm{dagr}}$) are simpler than they appear: $\circ^{d,\,\mathrm{agr}}$ works as a scoring rule allowing interpretations other than the models of κ as the minimal elements of the preorder $\leq_\kappa$. Notice that the score of an interpretation in $\leq_\kappa^{d,\,\mathrm{agr}}$ is 0 if it is either a model of κ, or it is equidistant from every model of κ (i.e., its centrality is 0) and it is the 'closest' interpretation to κ with this property. The disagreement operator $\circ^{d,\,\mathrm{dagr}}$ does something similar, by making models of $\overline{\kappa}$ and interpretations minimally equidistant to them the least plausible interpretations in $\leq_\kappa^{d,\,\mathrm{dagr}}$.

Example 11. Take κ such that $[\kappa] = \{a, b, c\}$, and notice that $d_{\mathsf{H}}(\emptyset, \kappa) = (111)$ and $d_{\mathsf{H}}(abc, \kappa) = (222)$, i.e., they are both equidistant to κ, hence their centrality is 0. However, $\emptyset$ is closer to κ than abc (its displacement is 0, compared to abc's displacement of 1), and $\mathrm{agr}(\emptyset, \kappa) = 0$. Thus, what $\leq_\kappa^{\mathsf{H},\,\mathrm{agr}}$ does is to give a minimal score to models of κ and to the minimally equidistant interpretation $\emptyset$. By contrast, $\leq_\kappa^{\mathsf{H},\,\mathrm{dagr}}$ gives a maximal score to the models of $\overline{\kappa}$ and to the maximally equidistant interpretation abc.

All operators proposed here generate a total preorder $\leq_\kappa$ over interpretations, but differ in how they arrange the models of κ in $\leq_\kappa$, corresponding to different attitudes an agent can have towards its beliefs κ prior to any revision. The operator $\circ^{\mathsf{H},\,\min}$, known as Dalal's operator (Dalal 1988), considers all models of κ as the most plausible elements in $\leq_\kappa$ and is the only operator for which $\kappa \circ \mu$ is equivalent to $\kappa \wedge \mu$ when $\kappa \wedge \mu$ is consistent. Similarly, $\circ^{\mathsf{H},\,\mathrm{lmin}}$ also ranks models of κ as more plausible than any other interpretation, but discriminates among models of κ. The operators $\leq_\kappa^{\mathsf{H},\,\max}$ and $\leq_\kappa^{\mathsf{H},\,\mathrm{lmax}}$ work by pushing away models of $\overline{\kappa}$, under the assumption that they are the most implausible possible worlds. They differ as to how they arrange the models of $\overline{\kappa}$: $\leq_\kappa^{\mathsf{H},\,\max}$ considers them equally implausible, whereas $\leq_\kappa^{\mathsf{H},\,\mathrm{lmax}}$ uses the more fine-grained lexicographic approach. The operator $\circ^{\mathsf{H},\,\mathrm{agr}}$ makes models of κ the most plausible elements in $\leq_\kappa$ but does not stop here and allows

$\circ^{d,\,\mathrm{lmin}} \rightarrow \circ^{d,\,\mathrm{min}} \longrightarrow \circ^{d,\,\mathrm{agr}} \quad \circ^{d,\,\mathrm{lmax}} \rightarrow \circ^{d,\,\mathrm{max}} \quad \circ^{d,\,\mathrm{sum}}$

Figure 5: An arrow from x to y indicates that the output of operator x implies output of operator y for the same input

other interpretations on that position, in particular certain interpretations that are equidistant to κ as per Example 11. The intuition here is that an interpretation equally distanced from models of κ is something like a compromise point of view, with good chances of being correct if it is close to κ. The operator $\circ^{\mathsf{H},\,\mathrm{dagr}}$ is the dual of $\circ^{\mathsf{H},\,\mathrm{agr}}$ and, finally, operator $\circ^{\mathsf{H},\,\mathrm{sum}}$ evokes utilitarian approaches by choosing interpretations that minimize the sum of the distances to each model of κ, i.e., are close to κ on an aggregate level.

Plugging in the drastic and Hamming distances in the definition in Table 1 would seem to give us a considerable number of operators, but quick reflection shows that operators obtained with drastic distance d_{D} collapse into two main categories. To get a grasp on this fact, consider first the *drastic revision operator* $\circ^{\mathsf{dr}}$ defined, for $\kappa \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$ and $\mu \in \mathsf{Prop}$, as $\kappa \circ^{\mathsf{dr}} \mu = \kappa \wedge \mu$, if $\kappa \wedge \mu$ is consistent, and μ otherwise, and the *forgetful revision operator* $\circ^{\mathsf{fg}}$ defined as $\kappa \circ^{\mathsf{fg}} \mu = \mu$.

Proposition 6. For any knowledge base κ and formula μ, it holds that $\kappa \circ^{\mathsf{D},\,\mathrm{min}} \mu \equiv \kappa \circ^{\mathsf{D},\,\mathrm{lmin}} \mu \equiv \kappa \circ^{\mathsf{D},\,\mathrm{lmax}} \mu \equiv \kappa \circ^{\mathsf{D},\,\mathrm{sum}} \mu \equiv \kappa \circ^{\mathsf{dr}} \mu$. Moreover, $\kappa \circ^{\mathsf{D},\,\mathrm{agr}} \mu \equiv \kappa \circ^{\mathsf{fg}} \mu$ and:

$$\kappa \circ^{\mathsf{D},\,\mathrm{max}} \mu \equiv \kappa \circ^{\mathsf{D},\,\mathrm{dagr}} \mu \equiv \begin{cases} \kappa \circ^{\mathsf{dr}} \mu, & \text{if } \kappa \text{ is complete,} \\ \kappa \circ^{\mathsf{fg}} \mu, & \text{otherwise.} \end{cases}$$

Proof. If κ is a complete knowledge base and $[\kappa] = \{w_0\}$, then $d_{\mathsf{D}}(w, \kappa) = d_{\mathsf{D}}(w, w_0)$, for any interpretation w. In other words, $\overrightarrow{d_{\mathsf{D}}(w, \kappa)}$ yields (0) if $w = w_0$ and (1) otherwise. Thus, $\max d_{\mathsf{D}}(w, \kappa) = 0$ if $w = w_0$, and $\max d_{\mathsf{D}}(w, \kappa) = 1$, otherwise. It is then straightforward to see that if $w_0 \in [\mu]$, then $[\kappa \circ^{\mathsf{D},\,\mathrm{max}} \mu] = \{w_0\} = [\kappa \wedge \mu]$, and if $w_0 \notin [\mu]$, then $[\kappa \circ^{\mathsf{D},\,\mathrm{max}} \mu] = [\mu]$. If κ is not complete, then $\overrightarrow{d_{\mathsf{D}}(w, \kappa)}$ yields $(01 \ldots 1)$ if $w \in [\kappa]$, and $(1 \ldots 1)$ otherwise. It is now straightforward to see that the remaining statements of Proposition 6 hold. □

With Hamming distance the landscape is more diverse, as the different attitudes the operators assume towards models of κ lead to genuinely different revision strategies. Nonetheless, certain relationships between the operators still hold.

Proposition 7. For any $\kappa \in 2^{\mathsf{Prop}}_{\mathsf{cons}}$ and $\mu \in \mathsf{Prop}$, we have that $\kappa \circ^{\mathsf{H},\,\mathrm{lmin}} \mu \models \kappa \circ^{\mathsf{H},\,\mathrm{min}} \mu \models \kappa \circ^{\mathsf{H},\,\mathrm{agr}} \mu$ and $\kappa \circ^{\mathsf{H},\,\mathrm{lmax}} \mu \models \kappa \circ^{\mathsf{H},\,\mathrm{max}} \mu \models \kappa \circ^{\mathsf{H},\,\mathrm{dagr}} \mu$.

The relationship between the different operators is depicted in Figure 5. One can see that lexicographic operators are the most discriminating, in the sense that they pick formulas with fewer models (i.e., more specific formulas).

Since all operators generate total preorders over interpretations, by Theorem 1 they all satisfy axioms R_{1-5}. Satisfaction with respect to the newly introduced postulates is clarified by the following result.

Proposition 8. For a distance $d \in \{\mathsf{D}, \mathsf{H}\}$ and a comparison function $f \in \{\mathrm{min}, \mathrm{lmin}, \mathrm{max}, \mathrm{lmax}, \mathrm{agr}, \mathrm{dagr}, \mathrm{sum}\}$, the operators $\circ^{d,\,f}$ satisfy postulates R_{6-10} as shown in Table 3.

	R_6	R_7	R_8	R_9	R_{10}	R_{N}	$\mathsf{R}_{\mathsf{IDF}}$	$\mathsf{R}_{\mathsf{STB}}$
$\circ^{\mathsf{H},\,\mathrm{min}}$	✓	✓	×	×	✓	✓	✓	✓
$\circ^{\mathsf{H},\,\mathrm{lmin}}$	×	✓	×	×	✓	✓	×	✓
$\circ^{\mathsf{H},\,\mathrm{agr}}$	✓	×	×	×	✓	✓	✓	✓
$\circ^{\mathsf{H},\,\mathrm{max}}$	×	×	✓	✓	×	✓	×	✓
$\circ^{\mathsf{H},\,\mathrm{lmax}}$	×	×	×	✓	×	✓	×	✓
$\circ^{\mathsf{H},\,\mathrm{dagr}}$	×	×	✓	×	×	✓	✓	✓
$\circ^{\mathsf{H},\,\mathrm{sum}}$	×	×	×	×	✓	✓	×	✓
$\circ^{\mathsf{dr}}$	✓	✓	×	×	✓	✓	✓	✓
$\circ^{\mathsf{fg}}$	✓	×	✓	×	✓	✓	✓	✓

Table 3: Satisfaction of axioms

Proof. It is already known that $\leq_\kappa^{d,\,\mathrm{min}}$ (known as Dalal's operator (Dalal 1988; Katsuno and Mendelzon 1991)) satisfies axioms R_{5-6}. To see why the operator $\circ^{\mathsf{H},\,\mathrm{lmin}}$ satisfies R_7, notice that if $w_1 \in [\kappa]$ and $w_2 \notin [\kappa]$, then the first element in $d(w_1, \kappa)$ is 0, while the first element in $\overrightarrow{d(w_2, \kappa)}$ is strictly greater than 0. This implies that $\overrightarrow{d(w_1, \kappa)} <_{\mathsf{lex}} \overrightarrow{d(w_2, \kappa)}$, which in turn implies that $w_1 <_\kappa^{\mathsf{H},\,\mathrm{lmin}} w_2$. Hence property o_7 is satisfied, which implies that axiom R_7 is satisfied. Additionally, it cannot be the case that $w_3 <_\kappa^{\mathsf{H},\,\mathrm{lmin}} w_1$, for any $w_3 \notin [\kappa]$, which shows that property o_6 (and hence axiom R_6) is satisfied. For $\circ^{\mathsf{H},\,\mathrm{lmin}}$ and axiom R_6, take $[\kappa] = \{a, b, ab\}$ and $[\mu] = \{a, b, ab\}$. We get that $[\kappa \circ^{\mathsf{H},\,\mathrm{lmin}} \mu] = \{a, b\}$. The operator $\circ^{\mathsf{H},\,\mathrm{agr}}$ satisfies axiom R_6 because it makes all models of κ, and potentially other interpretations as well (which is the reason why it does not satisfy axiom R_7), as the equally most plausible interpretations in $\leq_\kappa^{\mathsf{H},\,\mathrm{agr}}$. Since all these operator place the models of κ on the lowest levels of $\leq_\kappa$, they all satisfy axiom R_{10}.

To see why postulates R_{8-9} are not satisfied by $\circ^{d,\,\mathrm{min}}$, $\circ^{d,\,\mathrm{lmin}}$ or $\circ^{d,\,\mathrm{dagr}}$, notice that these operators do not make models of $\overline{\kappa}$ as the least plausible interpretations in $\leq_\kappa$. Thus, if $\kappa = a \vee b$, then $\overline{\kappa}$ shares some models with κ, yet these models (along with all other models of $a \vee b$) will be among the most plausible interpretations in $\leq_\kappa^{d,\,\mathrm{min}}$, $\leq_\kappa^{d,\,\mathrm{lmin}}$ and $\leq_\kappa^{d,\,\mathrm{agr}}$. The one exception is the forgetful operator $\circ^{\mathsf{fg}}$, which satisfies R_8 trivially.

The case for $\circ^{\mathsf{H},\,\mathrm{max}}$, $\circ^{\mathsf{H},\,\mathrm{lmax}}$ and $\circ^{\mathsf{H},\,\mathrm{dagr}}$ is analogous to the one for $\circ^{\mathsf{H},\,\mathrm{min}}$, $\circ^{\mathsf{H},\,\mathrm{lmin}}$ and $\circ^{\mathsf{H},\,\mathrm{agr}}$, as they can be seen as duals of each other. For the operator $\circ^{\mathsf{H},\,\mathrm{sum}}$, take $[\kappa] = \{a, b, c\}$ and $[\mu] = \{\emptyset, a, b, c\}$. We get that $[\kappa \circ^{\mathsf{H},\,\mathrm{sum}} \mu] = \{\emptyset\}$, as $\emptyset$ minimizes the sum of the Hamming distances to the models of κ: this is a counter-example to axioms R_{6-7}. For R_{8-9}, take $[\mu'] = \{\emptyset, ab, ac, bc\}$. For $\circ^{\mathsf{H},\,\mathrm{sum}}$ and R_{10}, notice that adding w' to $[\kappa]$ creates a new column for w' in the table of distances, in which the distance corresponding to w' is 0, i.e., the score assigned to w' in $\leq_{\kappa \vee \kappa'}^{\mathsf{H},\,\mathrm{sum}}$ does not increase with respect to $\leq_\kappa^{\mathsf{H},\,\mathrm{sum}}$. Satisfaction of $\mathsf{R}_{\mathsf{IDF}}$ and $\mathsf{R}_{\mathsf{STB}}$ is straightforward, keeping in mind how the various operators arrange the models of κ in the generated preorders. The neutrality axiom R_{N} is discussed separately. □

With respect to neutrality, Table 3 shows that all operators introduced so far satisfy postulate R_{N}. This is guaranteed by a property of the distances which we will, by an overload of

notation, call by the same name. Thus, a distance d is *neutral* if, for any renaming ρ and interpretations w_1 and w_2, it holds that $d(w_1, w_2) = d(\rho(w_1), \rho(w_2))$. It is straightforward to see that the drastic and Hamming distances are neutral. Furthermore, if d is neutral, then $d(w, \kappa) = d(\rho(w), \rho(\kappa))$, for any $w \in \mathcal{U}$, and $w_1 \leq_\kappa^{d,f} w_2$ iff $\rho(w_1) \leq_{\rho(\kappa)}^{d,f} \rho(w_2)$, for all selection functions introduced so far. Thus, the preorders $\leq_\kappa^{d,f}$ satisfy property $\mathsf{o_N}$ and, by Theorem 2, the operators represented by them satisfy postulate $\mathsf{R_N}$. It should be kept in mind that neutrality is not guaranteed by the other postulates, but the way in which concrete operators are usually defined (i.e., by appeal to neutral distances) indicates that neutrality is part of our basic understanding of how a revision operator should behave. And, in general, there seems to be no *a priori* reason for looking at non-neutral operators. However, it turns out that such operators cannot be avoided when we move to a fragment of propositional logic.

Revision of Horn theories

Discussions of stability and varying attitudes to initial beliefs notwithstanding, one might still question the rationale behind giving up the KM axiom R_2 (axioms R_{6-7} in the present work): indeed, why fix something if it is not broken? In response, we will use this section to argue that there are situations where revision is warranted, but in which axioms R_{6-7} *cannot* occur together.

Recall that a clause is called *Horn* if at most one of its literals is positive, a *Horn formula* is a conjunction of Horn clauses and a *Horn knowledge base* is a finite set of Horn formulas. The set of all Horn formulas is $\mathcal{L}_{\mathsf{Horn}}$. Horn formulas are characterized semantically by closure under intersection, i.e., if W is a set of interpretations then there exists a Horn formula φ such that $[\varphi] = W$ iff $\mathrm{Cl}_\cap(W) = W$.

As mentioned in the introduction, there is good reason to want to do revision on Horn knowledge bases, and to ask that the result is still a Horn knowledge base. We thus consider H-revision operators $\bullet\colon 2^{\mathcal{L}_{\mathsf{Horn}}}_{\mathsf{cons}} \times \mathsf{Prop} \to \mathcal{L}_{\mathsf{Horn}}$, mapping a Horn knowledge base κ and a propositional formula μ to a Horn knowledge base $\kappa \bullet \mu$. The natural next step now would be to adapt the regular propositional postulates to the new setting of H-revision. Postulates R_{1-5} can be adapted seamlessly, and we may denote the adapted postulates $\mathsf{R}^{\mathsf{H}}_{1-5}$, but consider what happens if we try to introduce a postulate that is an adapted version of the standard KM postulate R_2, saying that $\kappa \bullet \mu \equiv \kappa \wedge \mu$, if $\kappa \wedge \mu$ is consistent. For ease of reference, we split this postulate into two weaker postulates:

($\mathsf{R}^{\mathsf{H}}_6$) If $\kappa \wedge \mu$ is consistent, then $\kappa \wedge \mu \models \kappa \bullet \mu$.

($\mathsf{R}^{\mathsf{H}}_7$) If $\kappa \wedge \mu$ is consistent, then $\kappa \bullet \mu \models \kappa \wedge \mu$.

It turns out that postulate $\mathsf{R}^{\mathsf{H}}_6$ cannot be used in this context.

Example 12. Take a Horn knowledge base $\kappa = \{\neg a \vee \neg b\}$ and a $\mu = a \leftrightarrow \neg b$. Clearly, $\kappa \wedge \mu$ is satisfiable and, moreover, $\kappa \wedge \mu \equiv \mu$. However, $[\kappa \wedge \mu] = \{a, b\}$, which is not equal to $\mathrm{Cl}_{\mathsf{Horn}}([\kappa \wedge \mu])$ and thus does not represent any Horn formula. If $\mathsf{R}^{\mathsf{H}}_6$ were true, then with $\mathsf{R}^{\mathsf{H}}_1$ we would have to conclude that $[\kappa \bullet \mu] = [\kappa \wedge \mu] = [\mu]$, a contradiction.

Thus, it seems that H-revision operators cannot be axiomatized in a way that is analogous to propositional operators satisfying axioms R_1 and R_6, or, equivalently: we cannot model an agent who, when revising a Horn knowledge base κ, always makes the models of κ equally plausible. This can be stated as a corollary, following directly from Example 12.

Corollary 9. If an H-revision operator satisfies axiom $\mathsf{R}^{\mathsf{H}}_1$, then it does not satisfy axiom $\mathsf{R}^{\mathsf{H}}_6$.

Since we are not prepared to sacrifice axiom R_1, we are left with satisfying axiom R_7. What about neutrality? It turns out that this is also problematic for H-revision operators: since H-revision by $\mu = a \leftrightarrow \neg b$ must return, by $\mathsf{R}^{\mathsf{H}}_{1,3}$, a consistent result which implies μ and is a Horn formula, such an operator must effectively choose exactly one of the interpretations a and b. This leads to a clash with the adapted neutrality postulate, which we may call $\mathsf{R}^{\mathsf{H}}_{\mathsf{N}}$.

Proposition 10. If an H-revision operator satisfies axioms $\mathsf{R}^{\mathsf{H}}_{1-3}$, then it does not satisfy axiom $\mathsf{R}^{\mathsf{H}}_{\mathsf{N}}$.

Proof. Take $\kappa = a \wedge b$ and $\mu = a \leftrightarrow \neg b$. Suppose $\kappa \bullet \mu \equiv a \wedge \neg b$, and take a renaming ρ such that $\rho(a) = b$ and $\rho(b) = a$ to get a contradiction. □

The move to be explicit about neutrality and to split the standard KM postulate R_2 into two distinct properties (postulates $\mathsf{R}_{6,7}$), either of which can be turned off, finds additional justification here: we can see now that properties taken for granted in the propositional case break down when restricting the language, and a thorough analysis of what are rational, or desirable, properties for revision must take this into account.

Conclusion

We have looked at the classical revision axioms from the point of view of what they assume about an agent's attitude towards its initial beliefs, and argued that this attitude is embedded in a specific axiom (KM axiom R_2), By varying this axiom and calling attention to a commonly overlooked neutrality property, we were able to put forward and characterize a wide range of revision operators, and refine previously entangled intuitions in the process. We also showed that this level of analysis is needed when working in restricted fragments of propositional logic, where the KM axiom R_2 cannot be satisfied and must therefore be broken down into two separate components (axioms R_{6-7} in the current work).

Analysis of the new operators uncovered the principles of indifference to already held beliefs ($\mathsf{R}_{\mathsf{IDF}}$) and stability ($\mathsf{R}_{\mathsf{STB}}$). Further work is needed to link these notions to the other axioms, to map out the interplay between them, and to provide them with semantic characterizations. Following the line of reasoning initiated in the previous section, a natural follow-up would be to consider the proposed postulates in fragments of propositional logic and to look for characterizations in terms of preorders on possible worlds. At the same time, the more fine grained view on the types of attitudes an agent can have towards its initial beliefs raises the question of what these attitudes are good for, i.e., whether they can be put to use in an area such as learning (Kelly 1998; Baltag *et al.* 2011).

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF): P30168-N31, W1255-N23, Y698.

References

Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *J. Symb. Log.*, 50(2):510–530, 1985.

Kenneth J. Arrow. Rational choice functions and orderings. *Economica*, 26(102):121–127, 1959.

Alexandru Baltag, Nina Gierasimczuk, and Sonja Smets. Belief revision as a truth-tracking process. In *Proc. of TARK 2011*, pages 187–190, 2011.

Salem Benferhat, Sylvain Lagrue, and Odile Papini. Revision of Partially Ordered Information: Axiomatization, Semantics and Iteration. In *Proc. of IJCAI 2005*, pages 376–381, 2005.

Giacomo Bonanno. Rational choice and AGM belief revision. *Artif. Intell.*, 173(12-13):1194–1203, 2009.

Nadia Creignou, Odile Papini, Reinhard Pichler, and Stefan Woltran. Belief revision within fragments of propositional logic. *J. Comput. Syst. Sci.*, 80(2):427–449, 2014.

Mukesh Dalal. Investigations into a Theory of Knowledge Base Revision. In *Proc. of IJCAI 1988*, pages 475–479, 1988.

Adnan Darwiche and Judea Pearl. On the Logic of Iterated Belief Revision. *Artif. Intell.*, 89(1-2):1–29, 1997.

James P. Delgrande and Pavlos Peppas. Belief revision in Horn theories. *Artif. Intell.*, 218:1–22, 2015.

James P. Delgrande, Pavlos Peppas, and Stefan Woltran. General Belief Revision. *J. ACM*, 65(5):29:1–29:34, 2018.

Ulle Endriss. Judgment Aggregation. In Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, editors, *Handbook of Computational Social Choice*, pages 399–426. Cambridge University Press, 2016.

Eduardo Fermé and Sven Ove Hansson. *Belief Change: Introduction and Overview*. Springer, 2018.

Richard Foley. *Working Without a Net: A Study of Egocentric Epistemology*. Oxford University Press, 1993.

Simon French. *Decision Theory: An Introduction to the Mathematics of Rationality*. Ellis Horwood Limited, 1988.

Peter Gärdenfors. *Knowledge in Flux: Modelling the Dynamics of Epistemic States*. The MIT Press, Cambridge, MA, 1988.

Sven Ove Hansson. A Survey of non-Prioritized Belief Revision. *Erkenntnis*, 50(2):413–427, 1999.

Adrian Haret, Andreas Pfandler, and Stefan Woltran. Beyond IC Postulates: Classification Criteria for Merging Operators. In *Proc. of ECAI 2016*, pages 372–380, 2016.

Andreas Herzig and Omar Rifi. Propositional Belief Base Update and Minimal Change. *Artif. Intell.*, 115(1):107–138, 1999.

Hirofumi Katsuno and Alberto O. Mendelzon. On the Difference between Updating a Knowledge Base and Revising It. In *Proc. of KR 1991*, pages 387–394, 1991.

Hirofumi Katsuno and Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. In Peter Gärdenfors, editor, *Belief Revision*, pages 183–203. Cambridge University Press, 1992.

Hirofumi Katsuno and Alberto O. Mendelzon. Propositional Knowledge Base Revision and Minimal Change. *Artif. Intell.*, 52(3):263–294, 1992.

Kevin T. Kelly. The Learning Power of Belief Revision. In *Proc. of TARK 1998*, pages 111–124, 1998.

Sébastien Konieczny and Ramón Pino Pérez. Logic Based Merging. *J. Philosophical Logic*, 40(2):239–270, 2011.

Sébastien Konieczny, Jérôme Lang, and Pierre Marquis. DA^2 merging operators. *Artif. Intell.*, 157(1-2):49–79, 2004.

Hector J Levesque. Making believers out of computers. *Artif. Intell.*, 30(1):81–108, 1986.

Pierre Marquis and Nicolas Schwind. Lost in translation: Language independence in propositional logic - application to belief change. *Artif. Intell.*, 206:1–24, 2014.

Hervé Moulin. *Axioms of cooperative decision making*. Number 15. Cambridge university press, 1991.

Pavlos Peppas and Mary-Anne Williams. Kinetic Consistency and Relevance in Belief Revision. In *Proc. of JELIA 2016*, pages 401–414, 2016.

Jörg Rothe, editor. *Economics and Computation*. Springer, 2015.

Hans Rott. *Change, choice and inference: A study of belief revision and nonmonotonic reasoning*. Number 42. Oxford University Press, 2001.

Mark Ryan. Belief Revision and Ordered Theory Presentations. In *Logic, Action, and Information*, pages 129–151, 1996.

Amartya K. Sen. *Collective Choice and Social Welfare*. NY: North-Holland, 2nd edition, 1984.

Fernando R. Velázquez-Quesada. On Subtler Belief Revision Policies. In *Proc. of LORI 2017*, pages 314–329, 2017.

Zhiqiang Zhuang, Maurice Pagnucco, and Yan Zhang. Inter-Definability of Horn Contraction and Horn Revision. *J. Philosophical Logic*, 46(3):299–332, 2017.

Implementing Logic Programs with Ordered Disjunction Using *asprin*

Joohyung Lee and Zhun Yang
School of Computing, Informatics and Decision Systems Engineering
Arizona State University, Tempe, USA
{joolee, zyang90}@asu.edu

Abstract

Logic Programs with Ordered Disjunction (LPOD) is an extension of standard answer set programs to handle preference using the high-level construct of ordered disjunction whereas *asprin* is a recently proposed, general, flexible, and extensible framework that provides low-level constructs for representing preference in answer set programming. We present an encoding of LPOD in the language of *asprin* and the implementation LPOD2ASPRIN based on the encoding. Unlike the known method that applies only to a fragment of LPOD via the translation to Answer Set Optimization (ASO), our translation is general, direct, and simpler. It also leads to more efficient computation of LPOD using *asprin*.[1]

1 Introduction

Logic Programs with Ordered Disjunction (LPOD) (Brewka 2002) is an extension of standard answer set programs to handle preference using the high-level construct of ordered disjunction. *asprin* (Brewka et al. 2015b) is a recently proposed, general, flexible, and extensible framework for expressing and computing preferences in answer set programming, and, as such, the preference specification in the language of *asprin* is in a lower level than LPOD. Representing high-level preference constructs in the language of *asprin* could be verbose, and end-users may find it complicated to use. To alleviate the problem, *asprin* provides a library that implements several preference types, such as `subset`, `less(weight)`, and `ASO`. However, LPOD preference types are not one of them.

In (Brewka, Niemelä, and Syrjänen 2004), LPOD is implemented using SMODELS by interleaving the execution of two ASP programs—a *generator* which produces candidate answer sets and a *tester* which checks whether a given candidate answer set is most preferred or produces a more preferred answer set otherwise. In principle, the encodings in (Brewka, Niemelä, and Syrjänen 2004) can be used with *asprin* to implement LPOD. However, this method introduces a large number of translation rules and auxiliary atoms since it does not utilize the main component of *asprin*, preference statements.

In fact, it is known that using preference statements, some fragment of LPOD can be succinctly represented in the language of *asprin* via the translation into Answer Set Optimization (ASO). Brewka, Niemelä, and Truszczynski (2003) show how to turn LPOD under Pareto-preference into ASO programs, and Brewka et al. (2015a) show that ASO programs can be represented in *asprin*. By combining the two results, the fragment of LPOD can be represented in *asprin*. It is also mentioned that LPOD under inclusion-preference can be turned into "ranked" ASO (Brewka et al. 2015a) but the representation appears quite complicated. Furthermore, it is not known how the results apply to the other LPOD preference criteria.

This paper presents a more direct and simpler translation from LPOD into the language of *asprin*, handling all four preference criteria from (Brewka 2005) in a uniform way. Based on the translation, we implemented the system LPOD2ASPRIN, which translates LPOD programs into the input language of *asprin* and internally invokes the *asprin* system. Our experiments show that the system is more scalable than the other methods of computing LPOD.

The paper is organized as follows. Section 2 reviews LPOD and *asprin*. Section 3 presents a translation that turns LPOD into the language of *asprin*. Section 4 presents the LPOD2ASPRIN system and Section 5 compares its performance with other methods of computing LPOD. Section 6 discusses the related work. Selected proofs are given in the appendix.

2 Review of LPOD and *asprin*

2.1 Review: LPOD

We review the definition of LPOD by (Brewka 2002). As in that paper, for simplicity, we assume the underlying signature is propositional.

Syntax: A (propositional) LPOD Π is $\Pi_{reg} \cup \Pi_{od}$, where its *regular part* Π_{reg} consists of usual ASP rules

$$\mathit{Head} \leftarrow \mathit{Body}$$

and its *ordered disjunction part* Π_{od} consists of *LPOD rules* of the form

$$C^1 \times \cdots \times C^n \leftarrow \mathit{Body} \tag{1}$$

[1] A short version of this paper is to appear in the *16th International Conference on Principles of Knowledge Representation and Reasoning KR 2018*. This paper contains more details about the system LPOD2ASPRIN and includes experimental results.

in which C^i are atoms, n is at least 2, and *Body* is a conjunction of atoms possibly preceded by *not*.[2] Rule (1) says "when *Body* is true, if possible then C^1; if C^1 is not possible then C^2; ...; if all of $C^1, \ldots, C^{n-1}$ are not possible then C^n." It is not the case that none of $C^1, \ldots, C^n$ is true when *Body* is true.

Semantics: For an LPOD rule (1), its *i-th option* ($i = 1, \ldots, n$) is defined as

$$C^i \leftarrow Body, not\ C^1, \ldots, not\ C^{i-1}.$$

A *split program* of an LPOD Π is obtained from Π by replacing each LPOD rule in Π_{od} by one of its options. A set S of atoms is a *candidate answer set* of Π if it is an answer set of a split program of Π.

Example 1 *(From (Brewka 2002)) The following LPOD* Π_1,

$$\begin{array}{lcl} a \times b & \leftarrow & not\ c \\ b \times c & \leftarrow & not\ d, \end{array}$$

has four split programs:

$$\begin{array}{ll} a \leftarrow not\ c & a \leftarrow not\ c \\ b \leftarrow not\ d & c \leftarrow not\ d, not\ b \\ \\ b \leftarrow not\ c, not\ a & b \leftarrow not\ c, not\ a \\ b \leftarrow not\ d & c \leftarrow not\ d, not\ b. \end{array}$$

Each of them has the following answer sets respectively, which are the candidate answer sets of Π_1.

$$\begin{array}{ll} \{a, b\} & \{c\} \\ \{b\} & \{b\}, \{c\}. \end{array}$$

A candidate answer set S of Π is said to *satisfy* rule (1)

- to degree 1 if S does not satisfy *Body*, and
- to degree j ($1 \leq j \leq n$) if S satisfies *Body* and $j = min\{k \mid C^k \in S\}$.

The notion of satisfaction degrees are the basis of defining a preference relation on the candidate answer sets of Π. For a candidate answer set S, let $S^i(\Pi)$ denote the set of rules in Π_{od} satisfied by S to degree i. For candidate answer sets S_1 and S_2 of Π, (Brewka 2005) introduces the following four preference criteria.

1. **Cardinality-Preferred:** S_1 is *cardinality-preferred* to S_2 ($S_1 >^c S_2$) if there is a positive integer i such that $|S_1^i(\Pi)| > |S_2^i(\Pi)|$, and $|S_1^j(\Pi)| = |S_2^j(\Pi)|$ for all $j < i$.
2. **Inclusion-Preferred:** S_1 is *inclusion-preferred* to S_2 ($S_1 >^i S_2$) if there is a positive integer i such that $S_2^i(\Pi) \subset S_1^i(\Pi)$, and $S_1^j(\Pi) = S_2^j(\Pi)$ for all $j < i$.
3. **Pareto-Preferred:** S_1 is *Pareto-preferred* to S_2 ($S_1 >^p S_2$) if there is a rule that is satisfied to a lower degree in S_1 than in S_2, and there is no rule that is satisfied to a lower degree in S_2 than in S_1.
4. **Penalty-Sum-Preferred:** S_1 is *penalty-sum-preferred* to S_2 ($S_1 >^{ps} S_2$) if the sum of the satisfaction degrees of all rules is smaller in S_1 than in S_2.

A candidate answer set S of Π is a *k-preferred* ($k \in \{c, i, p, ps\}$) *answer set* if there is no candidate answer set S' of Π such that $S' >^k S$.

When Π_{od} contains m LPOD rules, the *satisfaction degree list* of a candidate answer set S of Π is $(d_1, \ldots, d_m)$ where d_i is the degree to which S satisfies rule i in Π_{od}.

Example 1 (Continued) Recall that Π_1 has three candidate answer sets: $\{a, b\}$, $\{b\}$, and $\{c\}$. Their satisfaction degree lists are (1,1), (2,1), and (1,2), respectively. One can check that $\{a, b\}$ is the only preferred answer set according to any of the four preference criteria.

The following example shows differences in preferred answer sets depending on the different preference criteria.

Example 2 *To illustrate the difference among the four preference criteria, consider the following LPOD* Π_2 *about picking a hotel near the Grand Canyon.* $hotel(1)$ *is a 2 star hotel but is close to the Grand Canyon,* $hotel(2)$ *is a 3 star hotel and the distance is medium, and* $hotel(3)$ *is a 4 star hotel but is too far.*

$$close \times med \times far \times tooFar$$
$$star4 \times star3 \times star2$$

$$1\{hotel(X) : X = 1..3\}1$$
$$\bot \leftarrow hotel(1),\ not\ close$$
$$\bot \leftarrow hotel(1),\ not\ star2$$
$$\bot \leftarrow hotel(2),\ not\ med$$
$$\bot \leftarrow hotel(2),\ not\ star3$$
$$\bot \leftarrow hotel(3),\ not\ tooFar$$
$$\bot \leftarrow hotel(3),\ not\ star4$$

Π_2 *has* 4×3 *split programs but only the following three programs are consistent (The regular part of* Π_2 *is not listed).*

$$close$$
$$star2 \leftarrow not\ star4, not\ star3$$

$$med \leftarrow not\ close$$
$$star3 \leftarrow not\ star4$$

$$tooFar \leftarrow not\ close, not\ med, not\ far$$
$$star4$$

The candidate answer sets of Π_2 *and their satisfaction degree lists are*

$$\begin{array}{ll} S_1 = \{hotel(1), close, star2, \ldots\}, & (1, 3) \\ S_2 = \{hotel(2), med, star3, \ldots\}, & (2, 2) \\ S_3 = \{hotel(3), tooFar, star4, \ldots\}, & (4, 1) \end{array}$$

By definition, the cardinality-preferred answer set of Π_2 *is* S_1*, the inclusion-preferred answer sets are* S_1 *and* S_3*, the Pareto-preferred answer sets are* S_1*,* S_2 *and* S_3*, while the penalty-sum-preferred answer sets are* S_1 *and* S_2*.*

2.2 Review: *asprin*

asprin computes the most preferred answer sets of an ASP program P according to a preference specification $\hat{F}_s$ by repeated calls to CLINGO as in Figure 1. First, an arbitrary

[2] In (Brewka 2002), a usual ASP rule is viewed as a special case of a rule with ordered disjunction when $n = 1$ but in this paper, we distinguish them. This simplifies the presentation of the translation and also allows us to consider LPOD programs that are more general than the original definition by allowing modern ASP constructs such as aggregates.

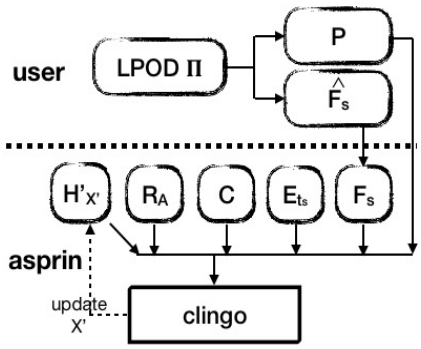


Figure 1: *asprin* Framework

answer set of P is generated as X'. Second, *asprin* tries to find an answer set X of P that is better than (i.e., preferred to) X' by running CLINGO on $P \cup F_s \cup E_{t_s} \cup H'_{X'} \cup R_A \cup C$, each of which is defined below. If CLINGO finds an answer set, which encodes the answer set X of P that is "better" than X', *asprin* replaces X' by X, and repeats the second step until CLINGO finds no answer sets, at which point X' is determined to be a most preferred answer set.

1. P is the *base program*, which consists of usual ASP rules. The answer sets of P are the "candidate answer sets" to apply a preference criterion.
2. $\hat{F}_s$ is the *preference specification* consisting of a single *optimization directive* of the form
$$\#optimize(s) \tag{2}$$
and a single [3] *preference statement* of the form
$$\#preference(s,t)\{e_1;\ldots;e_n\} \tag{3}$$
where $n \geq 0$; and s is the name of the preference statement; and t is its type (i.e., preference criterion). Each e_i is a *preference element* of the form
$$\phi_1 >> \cdots >> \phi_m$$
where $m \geq 1$ and each ϕ_i is a literal (an atom possibly preceded by *not*). [4] Intuitively, each index $1,\ldots,m$ gives the rank of the corresponding literal. The preference statement (3) declares a preference relation named s: each preference element in $\{e_1,\ldots,e_n\}$ gives a ranking of a set of literals while preference type t determines in which case one candidate answer set is better than another given the rankings. The optimization directive (2) tells *asprin* to restrict its reasoning mode to the preference relation declared by the preference statement whose name is s.
3. F_s is obtained from the preference specification $\hat{F}_s$ by turning the optimization directive (2) into an ASP fact
$$optimize(s)$$
and turning the preference statement (3) into an ASP fact
$$preference(s,t)$$
along with
$$preference(s,i,j,for(t_{\phi_j}),())$$
for each j-th literal ϕ_j in the i-th preference element e_i in (3). The term t_{ϕ_j} is defined as a if the literal ϕ_j is an atom a, and is $neg(a)$ if the literal ϕ_j is "*not* a". [5]
4. E_{t_s} is the *preference encoding* for t_s, where t_s is the type of the preference statement named s. It defines a reserved predicate $better(s)$, which is true iff there exists a candidate answer set X that is preferred to X' according to preference type t_s and the facts in F_s. In Section 3.3, we show four preference encodings $E_{lpod(c)}$, $E_{lpod(i)}$, $E_{lpod(p)}$, and $E_{lpod(ps)}$ for each of the four preference types (i.e., criteria) for LPOD.
5. $H'_{X'}$ is the set of ASP facts
$$\{holds'(a) \mid a \in X'\}$$
which reifies the atoms in X' in the form of $holds'(\cdot)$.[6]
6. R_A is the set of ASP rules
$$\{holds(a) \leftarrow a \mid a \text{ is an atom in } P\}$$
which reifies the atoms in any candidate answer set X in the form of $holds(\cdot)$.
7. C is a set of (domain-independent) ASP rules as follows.[7]
$$\bot \leftarrow \textit{not } better(S), optimize(S). \tag{4}$$
$$holds(neg(A)) \leftarrow \textit{not } holds(A), \quad preference(\_,\_,\_,for(neg(A)),\_). \tag{5}$$
$$holds'(neg(A)) \leftarrow \textit{not } holds'(A), \quad preference(\_,\_,\_,for(neg(A)),\_). \tag{6}$$
Rule (4) instructs the *asprin* system to find an answer set X that is better than X' according to the preference statement S. Rule (5) is about X, which is reified in the form of $holds(\cdot)$: for the literal of the form "*not* A" in the preference statement (3), it says $holds(neg(A))$ is true if $holds(A)$ is false in the reified X (i.e., $X \not\models A$). Similarly, rule (6) is about X', which is reified in the form of $holds'(\cdot)$.

Given a program P and a preference specification $\hat{F}_s$, we say an answer set X of P is a *preferred answer set* of P w.r.t. $\hat{F}_s$ if $P \cup F_s \cup E_t \cup H'_X \cup R_A \cup C$ has no answer set, where t is the type of the preference statement s declared in $\hat{F}_s$.

[3] *asprin* allows multiple preference statements in the input but for simplicity of the presentation we assume a single preference statement.

[4] In general, *asprin* allows for a more general syntax of preference specification and preference element. For the purpose of this paper, it is sufficient to consider this simple fragment.

[5] The last term is empty because we consider ϕ_j as a non-weighted formula.

[6] Note that this is based on the definition of H'_X, which is the set of ASP facts $\{holds'(a) \mid a \in X\}$.

[7] In general, C contains more rules such as the rule to define $holds(or(A,B))$. They are omitted because they are not related to our translation.

3 Representing LPOD in *asprin*

Let Π be an LPOD where Π_{od} consists of m propositional rules as follows.

$$\begin{aligned} 1: &\quad C_1^1 \times \cdots \times C_1^{n_1} \leftarrow \mathit{Body}_1 \\ &\quad \cdots \\ m: &\quad C_m^1 \times \cdots \times C_m^{n_m} \leftarrow \mathit{Body}_m \end{aligned} \tag{7}$$

where $1, \ldots, m$ are rule indices; $n_i \geq 2$ for $1 \leq i \leq m$.

In the following subsections, we present the component programs of *asprin* that encode LPOD Π, namely, P, $\hat{F}_s$, E_{t_s}. The other components, F_s, H'_X, R_A and C are generated as described above.

3.1 Base Program P

For the LPOD program $\Pi = \Pi_{reg} \cup \Pi_{od}$, the base program P contains all rules in Π_{reg} and, for each LPOD rule

$$C_i^1 \times \cdots \times C_i^{n_i} \leftarrow \mathit{Body}_i$$

in Π_{od}, P contains

$$\mathit{body}_i \leftarrow \mathit{Body}_i \tag{8}$$

$$\{C_i^1\} \leftarrow \mathit{body}_i \tag{9}$$

$$\cdots$$

$$\{C_i^{n_i-1}\} \leftarrow \mathit{body}_i, \mathit{not}\ C_i^1, \ldots, \mathit{not}\ C_i^{n_i-2} \tag{10}$$

$$C_i^{n_i} \leftarrow \mathit{body}_i, \mathit{not}\ C_i^1, \ldots, \mathit{not}\ C_i^{n_i-1} \tag{11}$$

Rule (8) defines the case when the body of rule i is true. Rules (9)–(10) say that if the body of rule i is true and each C_i^j is false ($j \in \{1, \ldots, k-1\}$), then C_i^k is possibly true. Rule (11) says that if the body of rule i is true and C_i^j is false for all $j \in \{1, \ldots, n_i - 1\}$, then $C_i^{n_i}$ must be true.

The above method of generating candidate answer sets using choice rules is from (Cabalar 2011). It is not difficult to check that the answer sets of this program P are the candidate answer sets of LPOD Π (ignoring body_i atoms).

Proposition 1 *For any LPOD Π and any set X of atoms in Π, X is a candidate answer set of Π iff $X \cup \{\mathit{body}_i \mid X$ satisfies the body of rule i in $\Pi_{od}\}$ is an answer set of P.*

Example 2 (Continued) For LPOD Π_2, the P-component of the *asprin* program is as follows.

```
body_1.
{close} :- body_1.
{med} :- body_1, not close.
{far} :- body_1, not close, not med.
tooFar :- body_1, not close, not med, not far.

body_2.
{star4} :- body_2.
{star3} :- body_2, not star4.
star2 :- body_2, not star4, not star3.

1{hotel(X): X=1..3}1.
:- hotel(1), not close.      :- hotel(1), not star2.
:- hotel(2), not med.        :- hotel(2), not star3.
:- hotel(3), not tooFar.     :- hotel(3), not star4.
```

The answer sets of the P-component are

$$\begin{array}{lllll} \{\mathit{hotel}(1), & \mathit{close}, & \mathit{star}2, & \mathit{body}_1, & \mathit{body}_2\} \\ \{\mathit{hotel}(2), & \mathit{med}, & \mathit{star}3, & \mathit{body}_1, & \mathit{body}_2\} \\ \{\mathit{hotel}(3), & \mathit{tooFar}, & \mathit{star}4, & \mathit{body}_1, & \mathit{body}_2\} \end{array}$$

which are exactly the unions of the candidate answer sets of Π and $\{\mathit{body}_1, \mathit{body}_2\}$.

3.2 Preference Specification $\hat{F}_s$

$\hat{F}_s$ contains an optimization directive

$$\#\mathit{optimize}(s)$$

and a preference statement

$$\begin{aligned} &\#\mathit{preference}(s, \mathit{lpod}(s))\ \{ \\ &\quad \mathit{not}\ \mathit{body}_1 >> C_1^1 >> \cdots >> C_1^{n_1}; \\ &\quad \cdots \\ &\quad \mathit{not}\ \mathit{body}_m >> C_m^1 >> \cdots >> C_m^{n_m} \\ &\} \end{aligned} \tag{12}$$

where $s \in \{c, i, p, ps\}$ denotes one of the four preference criteria for LPOD, and each line of (12) is associated with each LPOD rule. Intuitively, to check the satisfaction degree of an LPOD rule i, we check the truth value of the literals in the order specified in the i-th preference element. We first check whether *not* body_i is true. If *not* body_i is true, i.e., the body of rule i is false, the satisfaction degree must be 1 and we do not have to check further; and if it is not the case, check whether C_i^1 is true, and so on.

Example 2 (Continued) For LPOD Π_2 which contains LPOD rules

$$\begin{array}{l} \mathit{close} \times \mathit{med} \times \mathit{far} \times \mathit{tooFar} \\ \mathit{star}4 \times \mathit{star}3 \times \mathit{star}2 \end{array}$$

to find its cardinality-preferred answer sets, we set the preference criterion s to c, and let $\hat{F}_s$ be the following.

```
#optimize(c).

#preference(c, lpod(c)) {
  not body_1 >> close >> med >> far >> tooFar ;
  not body_2 >> star4 >> star3 >> star2
}.
```

asprin internally turns $\hat{F}_s$ into F_s as follows.

```
optimize(c).

preference(c, lpod(c)).

preference(c, 1, 1, for(neg(body_1)), ()).
preference(c, 1, 2, for(close), ()).
preference(c, 1, 3, for(med), ()).
preference(c, 1, 4, for(far), ()).
preference(c, 1, 5, for(tooFar), ()).

preference(c, 2, 1, for(neg(body_2)), ()).
preference(c, 2, 2, for(star4), ()).
preference(c, 2, 3, for(star3), ()).
preference(c, 2, 4, for(star2), ()).
```

The facts $optimize(c)$ and $preference(c, lpod(c))$ assert that we optimize according to the preference statement c of type $lpod(c)$ (inclusion preference). The fact $preference(c, 2, 1, for(neg(body\_2)), ())$ asserts that the first literal of the second preference element of the preference statement c is "*not* $body\_2$".

3.3 Preference Encoding E_{t_s}

The aim of E_{t_s} is to find an answer set X (reified in the form of $holds(\cdot)$) that is better than (i.e., preferred to) the current answer set X' (reified in the form of $holds'(\cdot)$) with respect to the preference type t_s.

We introduce the preference encodings E_{t_s} for each $t_s \in \{lpod(c),\ lpod(i),\ lpod(p),\ lpod(ps)\}$. Each E_{t_s} contains the common rules Deg as defined below.

Degree The aim of Deg is to find the satisfaction degree to which each LPOD rule R is satisfied by X or X'. [8]

Deg consists of the following two rules.

$$\begin{aligned} degree(R, D) \leftarrow\ & optimize(S), preference(S, lpod(\_)), \\ & preference(S, R, I, \_, \_),\ D = \#max\{1; I-1\}, \\ & I = \#min\{J : holds(A), preference(S, R, J, for(A), \_)\}. \end{aligned} \tag{13}$$

$$\begin{aligned} degree'(R, D) \leftarrow\ & optimize(S), preference(S, lpod(\_)), \\ & preference(S, R, I, \_, \_),\ D = \#max\{1; I-1\}, \\ & I = \#min\{J : holds'(A), preference(S, R, J, for(A), \_)\}. \end{aligned} \tag{14}$$

Rule (13) records the degree D to which rule R is satisfied by X (X is reified in the form of $holds(\cdot)$). It asserts that if we want to optimize according to preference statement S whose type is one of the four $lpod(\cdot)$ types, then we need to calculate the satisfaction degree D for each rule R: D is the maximum value of 1 and $I-1$ where I is the index of the first literal in the preference element R that is true in X. Rule (14) is similar to rule (13) except that it finds the satisfaction degree D of rule R for X'.

Cardinality-Preferred $E_{lpod(c)}$ contains Deg and the following two rules:

$$\begin{aligned} worse2degree(S, D) \leftarrow\ & optimize(S), preference(S, lpod(c)), \\ & degree'(\_, D), \\ & \#sum\{\ \ 1, R : degree(R, D); \\ & \qquad -1, R : degree'(R, D)\} < 0. \end{aligned} \tag{15}$$

$$\begin{aligned} better(S) \leftarrow\ & optimize(S), preference(S, lpod(c)), \\ & degree(\_, D), \\ & \#sum\{\ \ 1, R : degree(R, D); \\ & \qquad -1, R : degree'(R, D)\} > 0, \\ & not\ worse2degree(S, J) : J = 1..D-1. \end{aligned} \tag{16}$$

Rule (15) defines the case when X is worse than, i.e., less preferred to, X' at degree D: X satisfies less LPOD rules to degree D than X'. In this case, there must be at least one LPOD rule that is satisfied to degree D by X', which is guaranteed by $degree'(\_, D)$. Rule (16) says that X is better than X' according to the preference type $lpod(c)$ if there exists a degree D such that X is preferred to X' at degree D (i.e., X satisfies more rules to degree D than X') and X is not worse than X' at all lower degrees. Note that "*not* $worse2degree(S, J) : J = 1..D-1$" is a *conditional literal*, and is equivalent to the conjunction of literals "*not* $worse2degree(S, J)$" for all $J \in \{1, \ldots, D-1\}$.

Inclusion-Preferred $E_{lpod(i)}$ contains Deg and two rules:

$$\begin{aligned} prf2degree(S, D) \leftarrow\ & optimize(S), preference(S, lpod(i)), \\ & degree(\_, D), \\ & \#count\{J : degree(J, D), not\ degree'(J, D)\} > 0, \\ & degree(J, D) : degree'(J, D). \end{aligned} \tag{17}$$

$$\begin{aligned} better(S) \leftarrow\ & preference(S, lpod(i)), \\ & prf2degree(S, D), \\ & degree(R, J) : degree'(R, J), J < D. \end{aligned} \tag{18}$$

Rule (17) defines the case when X is preferred to X' at degree D: (i) X satisfies at least one rule to degree D; (ii) there is a rule J that is satisfied by X, but not by X', to degree D; and (iii) all rules J that are satisfied by X' to degree D are also satisfied by X to the same degree. Rule (18) says that X is better than X' according to preference type $lpod(i)$ if there exists a degree D such that X is preferred to X' at degree D, and any rule R that is satisfied by X' to a lower degree than D should also be satisfied by X to the same degree.

Pareto-Preferred $E_{lpod(p)}$ contains Deg and two rules:

$$\begin{aligned} equ(S) \leftarrow\ & optimize(S), preference(S, lpod(p)), \\ & D1 = D2 : degree(R, D1), degree'(R, D2). \end{aligned} \tag{19}$$

$$\begin{aligned} better(S) \leftarrow\ & optimize(S), preference(S, lpod(p)), \\ & not\ equ(S), \\ & D1 \leq D2 : degree(R, D1), degree'(R, D2). \end{aligned} \tag{20}$$

Rule (19) defines that X and X' are "equivalent" if they satisfy each LPOD rule to the same degree. Rule (20) says that X is better than X' according to preference type $lpod(p)$ if X is not "equivalent" to X', and X satisfies each LPOD rule R to a degree that is the same or lower than the degree to which X' satisfies R.

Penalty-Sum-Preferred $E_{lpod(ps)}$ contains Deg and one rule:

$$\begin{aligned} better(S) \leftarrow\ & optimize(S), preference(S, lpod(ps)), \\ & \#sum\{D, R : degree(R, D); \\ & \qquad -D, R : degree'(R, D)\} < 0. \end{aligned} \tag{21}$$

Rule (21) says that X is better than X' according to preference type $lpod(ps)$ if the sum of the degrees to which the LPOD rules are satisfied by X is lower than the sum of the degrees to which the LPOD rules are satisfied by X'.

Theorem 1 *For any LPOD* Π*, X is an s-preferred answer set ($s \in \{c, i, p, ps\}$) of* Π *in the sense of LPOD iff* $X \cup \{body_i \mid X$ *satisfies the body of rule* i *in* $\Pi_{od}\}$ *is a preferred answer set of P w.r.t. $\hat{F}_s$ in the sense of asprin, where P and $\hat{F}_s$ are obtained from* Π *as above.*

[8] Note that each preference element denotes an LPOD rule. We use symbol R to denote the index of the preference element in predicate "$preference(S, R, I, \_, \_)$" because R is also the index of the denoted LPOD rule.

4 LPOD2ASPRIN System

We implement system LPOD2ASPRIN as in Figure 2. The system first translates an LPOD program Π into a base program P and a preference specification $\hat{F}_s$ in the language of *asprin* as described in Sections 3.1 and 3.2, which are fed into the *asprin* system along with other component programs. We put the encodings $E_{lpod(c)}$, $E_{lpod(i)}$, $E_{lpod(p)}$, and $E_{lpod(ps)}$ in the *asprin* library. The encodings are exactly the same as those in Section 3.3 except that we eliminate the use of $\#min$ and $\#max$ by replacing rule (13) (and rule (14) accordingly) with

```
degree(R,1) :- preference(S, lpod(_)),
   preference(S,R,1,for(A),_), holds(A).
degree(R,D-1) :- preference(S, lpod(_)),
   preference(S,R,D,for(A),_), holds(A), D>1,
   not holds(B) : preference(S,R,J,for(B),_), 0<J, J<D.
```

The reason for this change is because our experiments show significant speed-up with the alternative encoding.

Finally, an s-preferred answer set of Π is obtained from the output of *asprin* by removing the auxiliary atoms $body_i$.

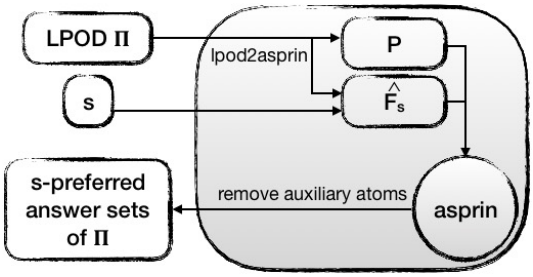


Figure 2: LPOD2ASPRIN System Overview

The LPOD2ASPRIN system homepage is

`http://reasoning.eas.asu.edu/lpod2asprin/`

which contains the source code, the tutorial, examples and some experimental results.

To find an s-preferred ($s \in \{c, i, p, ps\}$) answer set of an LPOD Π, one may execute the command

```
python lpod2asprin.py -i input.txt -type s
```

where

- `input.txt` stores the LPOD Π written in the input format of CLINGO except that the symbol `>>` is used to denote the ordered disjunction symbol; and
- `s` is one of the preference criteria in {`c`, `i`, `p`, `ps`}.

Example 2 (Continued) In the language of LPOD2ASPRIN, Π_2 is written as

```
dom(1..3).

1{hotel(X) : dom(X)}1.
:- hotel(1), not close.
:- hotel(1), not star2.
:- hotel(2), not med.
:- hotel(2), not star3.
:- hotel(3), not tooFar.
:- hotel(3), not star4.

close >> med >> far >> tooFar.

star4 >> star3 >> star2.
```

If we save this program in file hotel.txt and want to find the i-preferred answer set of Π_2, we can execute

```
python lpod2asprin.py -i hotel.txt -type i
```

which outputs

```
Input LPOD program: hotel.txt
Type of LPOD preference criterion: i

asprin version 3.0.2
Reading from /asprin-3.0.2/asprin/lpod.lp ...
Solving...
Answer: 1
dom(1) dom(2) dom(3) hotel(3) tooFar star4
OPTIMUM FOUND
Answer: 2
dom(1) dom(2) dom(3) hotel(1) close star2
OPTIMUM FOUND

Models       : 2
  Optimum    : yes
  Optimal    : 2
```

The output says that *asprin* finds two i-preferred answer sets of Π_2: $\{hotel(1), close, star2, \dots\}$ and $\{hotel(3), tooFar, star4, \dots\}$, which is as expected.

5 Experiments

Since there is no benchmarks available in the existing literature for LPOD, we designed two benchmarks and compare the run-time of LPOD2ASPRIN with two other methods: PSMODELS from (Brewka, Niemelä, and Syrjänen 2004) and using *asprin* via reduction to ASO (Brewka et al. 2015a). The latter method does not have a dedicated solver for LPOD, so we manually translate the LPOD programs into ASO programs and then into the input language of *asprin*. We only compare w.r.t. Pareto preference because it is not known how the *asprin* via ASO method can be used to represent cardinality preference and penalty-sum preference, and the representation for inclusion preference appears to be complicated.

5.1 Benchmark: abc

This benchmark is used to test how the system LPOD2ASPRIN performs with an increasing number of LPOD rules.

The **abc** example, as shown below, contains n LPOD rules, each of which contains two atoms in its head. The program generates one or two $a(i)$ and one or two $c(i)$ ($i \in \{1, \dots, n\}$), and restricts that $a(i)$ is true iff $b(i)$ is false for any $i \in \{1, \dots, n\}$. There is also a preference of $a(i)$ over $b(i)$ if $c(i)$ is true.

n	PSMODELS (2004)	*asprin* via ASO (2015a)	LPOD2ASPRIN
10	1.244s	3.977s	3.432s
15	10.088s	14.282s	11.374s
20	47.689s	45.860s	34.677s
25	2m47.842s	2m6.501s	1m32.202s
30	8m12.839s	5m1.220s	3m40.439s
35	18m26.952s	10m39.620s	7m42.605s
40	42m6.830s	20m25.556s	14m41.012s

Table 1: Experiment on the **abc** Example

```
dom(1..n).
1{a(X): dom(X)}2.                    1{c(X): dom(X)}2.
b(X) :- dom(X), not a(X).            :- a(X), b(X).

a(X) >> b(X) :- c(X).
```

Table 1 shows the run-time comparison of finding Pareto-preferred answer sets of this program with different values of n. We compare LPOD2ASPRIN with the implementation from (Brewka, Niemelä, and Syrjänen 2004) and the method via ASO reduction from (Brewka et al. 2015a) as we mentioned earlier.

In comparison, *asprin* using the reduction via ASO is more scalable than PSMODELS. However, since the semantics of ASO programs is analogous to the semantics of LPOD under Pareto preference, the reduction of LPOD into ASO programs is only straightforward under this preference. On the other hand, the LPOD2ASPRIN system works for any of the four preference criteria and scales better than the other methods.

5.2 Benchmark: n-Hotel

The **n-Hotel** example is about choosing a hotel from n candidate hotels based on the preferences over the prices, distances, and ratings of services. The input program contains three LPOD rules, each of which has n atoms in the head. The program is automatically generated with a parameter n, denoting the number of candidate hotels. Below is a program generated when $n = 4$ with random orders of price, distance, and service.

```
1{hotel(X): dom(X)}1.

:- hotel(X), price(Y), X!=Y, dom(X), dom(Y).
:- hotel(X), distance(Y), X!=Y, dom(X), dom(Y).
:- hotel(X), service(Y), X!=Y, dom(X), dom(Y).

dom(1..4).

price(3) >> price(4) >> price(1) >> price(2).

distance(4) >> distance(3) >> distance(1) >>
distance(2).

service(2) >> service(1) >> service(3) >> service(4).
```

The run-time of finding the Pareto-preferred answer sets of this program for different values of n is shown in Table 2.

n	PSMODELS (2004)	LPOD2ASPRIN
10	0.060s	0.438s
50	2.485s	1.961s
60	4.825s	1.788s
70	Terminate with No Result	3.147s
100	Terminate with No Result	6.730s
200	Terminate with No Result	47.868s
300	Terminate with No Result	1m45.006s

Table 2: Experiment on the **n-Hotel** Example

[9]

As we see, the LPOD2ASPRIN system is much more scalable than the original LPOD implementation. Besides, the original LPOD implementation cannot find any answer set if the number (n) of atoms in the head of an LPOD rule exceeds 70 whereas our system can find the preferred answer set even when $n = 800$ (800 is not an upper bound for our system, but is the biggest number we have tested).

In summary, the experimental results on the benchmarks show that the LPOD2ASPRIN system is more scalable than PSMODELS (Brewka, Niemelä, and Syrjänen 2004). In comparison with the method via ASO reduction, the method is more general to cover all preference types, and the encoding is more compact leading to more scalable computation.

6 Related Work and Conclusion

We already mentioned the method of (Brewka et al. 2015a) works under the Pareto preference. However, the reduction under inclusion preference requires a translation from LPOD to "ranked" ASO programs, which further requires a more complex reduction to *asprin*. Besides, the reductions from LPOD to ASO programs under cardinality and penalty-sum preferences were not shown. In comparison, our method reduces LPOD directly to *asprin*, which yields a simpler and uniform method that applies to all preference criteria for LPOD.

Asuncion *et al.* (2014) present a first-order semantics of logic programs with ordered disjunction by a translation into second-order logic.

Lee and Yang (2018) show a reduction from LPOD to answer set programs, where the semantics of each preference type is also represented by standard ASP rules. Their reduction is one-pass: the preferred answer sets are computed by calling an answer set solver one time by generating all candidate answer sets to which preference criteria are applied. The computation is not as scalable as LPOD2ASPRIN which makes iterative calls to CLINGO.

asprin has a library of built-in preference types, but LPOD preference is not one of them. Our preference encodings may be included in the *asprin* library to benefit the end-users.

Acknowledgments: We are grateful to the anonymous referees for their useful comments. This work was partially

[9]We did not test the *asprin* via ASO reduction because we did not implement a compiler for this method while a manual translation takes too much efforts.

supported by the National Science Foundation under Grants IIS-1526301 and IIS-1815337.

References

Asuncion, V.; Zhang, Y.; and Zhang, H. 2014. Logic programs with ordered disjunction: first-order semantics and expressiveness. In *Proceedings of the Fourteenth International Conference on Principles of Knowledge Representation and Reasoning*, 2–11. AAAI Press.

Brewka, G.; Delgrande, J.; Romero, J.; and Schaub, T. 2015a. Implementing preferences with asprin. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, 158–172. Springer.

Brewka, G.; Delgrande, J. P.; Romero, J.; and Schaub, T. 2015b. asprin: Customizing answer set preferences without a headache. In *AAAI*, 1467–1474.

Brewka, G.; Niemelä, I.; and Syrjänen, T. 2004. Logic programs with ordered disjunction. *Computational Intelligence* 20(2):335–357.

Brewka, G.; Niemelä, I.; and Truszczynski, M. 2003. Answer set optimization. In *IJCAI*, volume 3, 867–872.

Brewka, G. 2002. Logic programming with ordered disjunction. In *AAAI/IAAI*, 100–105.

Brewka, G. 2005. Preferences in answer set programming. In *CAEPIA*, volume 4177, 1–10. Springer.

Cabalar, P. 2011. A logical characterisation of ordered disjunction. *AI Communications* 24(2):165–175.

Lee, J., and Yang, Z. 2018. Translating LPOD and CR-Prolog2 into standard answer set programs. *Journal of Theory and Practice of Logic Programming (TPLP)*, 18(3–4): 589–606.

Appendix: Proof of Theorem 1

Let X be a set of atoms and let σ be a signature. By $X|_\sigma$, we denote the projection of X onto σ.

Lemma 1 *Let Π be an answer set program, let X be an answer set of Π, and let $constraint$ be a rule of the form "$\bot \leftarrow$ Body". If $X \models constraint$, X is an answer set of $\Pi \cup \{constraint\}$.*

Lemma 2 *Let Π be an answer set program of signature σ. Let X be a set of atoms in σ, let Body be a conjunction of atoms (possibly preceded by not) in σ, and let a be an atom not in σ. X is an answer set of Π iff $X \cup \{a \mid X \models \text{Body}\}$ is an answer set of $\Pi' \cup \{a \leftarrow \text{Body}\}$, where Π' is obtained from Π by replacing the occurrence of Body in the body of some (i.e., from zero to all) rules in Π with a.*

Theorem 1 *For any LPOD Π and any set X of atoms in Π, X is an s-preferred answer set ($s \in \{c, i, p, ps\}$) of Π according to LPOD iff $X \cup \{body_i \mid X$ satisfies the body of rule i in $\Pi_{od}\}$ is an s-preferred answer set of P according to $asprin$, where P is the base program obtained from Π. In other words, (let $\phi(X)$ be $X \cup \{body_i \mid X$ satisfies the body of rule i in $\Pi_{od}\}$)*

(a) *X is a candidate answer set of Π and*

(b) *there is no candidate answer set X' of Π that is s-preferred to X according to LPOD*

iff

(c) *$\phi(X)$ is an answer set of P and*

(d) *$P \cup F_s \cup E_{t_s} \cup H'_{\phi(X)} \cup R_A \cup C$ has no answer set.*

Proof. **($\rightarrow$)** Let Π be an LPOD of signature σ with m LPOD rules. Let X be a candidate answer set of Π such that there is no candidate answer set X' of Π that is s-preferred to X according to LPOD. By Proposition 1, $\phi(X)$ is an answer set of P.

Assume for the sake of contradiction that $P \cup F_s \cup E_{t_s} \cup H'_{\phi(X)} \cup R_A \cup C$ has an answer set S. Let σ' be $\sigma \cup \{body_i \mid i \in \{1, \ldots, m\}\}$. Note that σ' is the signature of P. By the splitting theorem, $S|_{\sigma'}$ is an answer set of P. By Proposition 1, $S|_\sigma$ is a candidate answer set of Π. We will prove that $S|_\sigma$ is s-preferred to X, which contradicts with bullet **(b)**.

Let a be an atom in σ. Here we list some facts that will be used in the proof.

1. By R_A, $S \models holds(a)$ iff $S|_{\sigma'} \models a$. Since a is an atom in σ, $S \models holds(a)$ iff $S|_\sigma \models a$.
2. By $H'_{\phi(X)}$, $S \models holds'(a)$ iff $\phi(X) \models a$. Since a is an atom in σ, $S \models holds'(a)$ iff $X \models a$.
3. By F_s, S satisfies $optimize(s)$ and S satisfies $preference(s, lpod(s))$.
4. By F_s, S satisfies $preference(s, r, j, for(a), ())$ iff the $(j-1)$-th atom ($j \geq 2$) in the head of LPOD rule r is a; S satisfies $preference(s, r, 1, for(neg(body_r)), ())$ iff Π contains an LPOD rule r.
5. By rules (5) and (6) in C, for $i \in \{1, \ldots, m\}$, S satisfies $holds(neg(body_i))$ iff S does not satisfy $holds(body_i)$; S satisfies $holds'(neg(body_i))$ iff S does not satisfy $holds'(body_i)$.
6. By rule (13) in E_{t_s}, S satisfies $degree(R, D)$ iff $S|_\sigma$ satisfies LPOD rule R to degree D. This is because for any LPOD rule r, in case $S \models degree(r, 1)$,
   - $S \models degree(r, 1)$

   iff (by rule (13), bullet 3, and bullet 4)
   - in the case when $I = 1$:
     - $S \models preference(s, r, 1, for(neg(body_r)), \_)$, and
     - $S \models holds(neg(body_r))$
   - or in the case when $I = 2$: there exists an atom a such that
     - $S \models preference(s, r, 2, for(a)), \_)$, and
     - $S \models holds(a)$, and
     - $S \not\models holds(neg(body_r))$

   iff (by bullets 4 and 5)
   - in the case when $I = 1$:
     - Π contains an LPOD rule r, and
     - S does not satisfy $holds(body_r)$
   - or in the case when $I = 2$: there exists an atom a such that
     - the 1-st atom in the head of LPOD rule r is a, and

– $S \models holds(a)$, and
– $S \models holds(body_r)$

iff (by bullet 1 and rule (8))

- in the case when $I = 1$:
– $S|_\sigma$ does not satisfy the body of LPOD rule r
- or in the case when $I = 2$:
– $S|_\sigma$ satisfies the first atom in the head of LPOD rule r, and
– $S|_\sigma$ satisfies the body of LPOD rule r

iff (by definition)

- $S|_\sigma$ satisfies LPOD rule r to degree 1;

and in case $S \models degree(r, d)$ where d is greater than 1,

- $S \models degree(r, d)$, $d \geq 2$

iff (by rule (13), bullet 3, and bullet 4)

- there exists an atom a such that
– $S \models preference(s, r, d + 1, for(a)), \_)$, and
– $S \models holds(a)$, and
- $S \not\models holds(b)$ for any b and d' such that $S \models preference(s, r, d' + 1, for(b)), \_)$ and $d' < d$

iff (by bullet 4)

- there exists an atom a such that
– the d-th atom in the head of LPOD rule r is a, and
– $S \models holds(a)$, and
- $S \not\models holds(b)$ for any d'-th atom b in the head of LPOD rule r where $d' < d$

iff (by bullet 1)

- $S|_\sigma$ satisfies the d-th atom in the head of LPOD rule r, and
- $S|_\sigma$ does not satisfy the d'-th atom in the head of LPOD rule r for any $d' < d$

iff (by definition)

- $S|_\sigma$ satisfies LPOD rule r to degree d.

7. Similarly, by rule (14) in E_{t_s}, S satisfies $degree'(R, D)$ iff X satisfies LPOD rule R to degree D. The proof is analogous to that in bullet 6.

Now, we will prove that $S|_\sigma$ is s-preferred to X for $s \in \{i, p, ps\}$. The proof for the case when s is c (cardinally preference) is not shown due to the limit of space.

- **Inclusion-Preferred** By bullet 3, S satisfies $optimize(i)$ and $preference(i, lpod(i))$. Since S satisfies rule (4), S must satisfy $better(i)$. Besides,
– $S \models better(i)$

iff (by rule (18))

– there exists an integer d such that
∗ $S \models prf2degree(i, d)$, and
∗ for any LPOD rule $r \in \{1, \ldots, m\}$ and any degree j such that $j < d$, if $S \models degree'(r, j)$, then $S \models degree(r, j)$

iff (by rule (17))

– there exists an integer d such that
∗ there exists at least one LPOD rule j such that $S \models degree(j, d)$ and $S \not\models degree'(j, d)$, and for any LPOD rule j, if $S \models degree'(j, d)$, then $S \models degree(j, d)$, and
∗ for any LPOD rule $r \in \{1, \ldots, m\}$ and any degree j such that $j < d$, if $S \models degree'(r, j)$, then $S \models degree(r, j)$

iff (by bullet 6 and bullet 7)

– there exists an integer d such that
∗ the set of LPOD rules that are satisfied to degree d by X is a proper subset of the set of LPOD rules that are satisfied to degree d by $S|_\sigma$, and
∗ for any degree j such that $j < d$, the set of LPOD rules that are satisfied to degree j by X is a subset of the set of LPOD rules that are satisfied to degree j by $S|_\sigma$

iff (by definition)

– $S|_\sigma$ is i-preferred to X according to LPOD.

So $S|_\sigma$ is i-preferred to X according to LPOD. Contradiction.

- **Pareto-Preferred** Since S satisfies rule (4), and satisfies $optimize(p)$ (in F_s), S must satisfy $better(p)$. Besides,
– $S \models better(p)$

iff (by rule (20))

– $S \not\models equ(p)$, and
– for any LPOD rule $r \in \{1, \ldots, m\}$, if $S \models degree(r, d_1)$ and $S \models degree'(r, d_2)$ for any d_1 and d_2, then $d_1 \leq d_2$

iff (by rule (19))

– it is not the case that for all LPOD rule $r \in \{1, \ldots, m\}$, $S \models degree(r, d)$ iff $S \models degree'(r, d)$ for any d, and
– for any LPOD rule $r \in \{1, \ldots, m\}$, if $S \models degree(r, d_1)$ and $S \models degree'(r, d_2)$ for any d_1 and d_2, then $d_1 \leq d_2$

iff (by bullet 6)

– there is an LPOD rule that is satisfied to a lower degree in $S|_\sigma$ than in X, and there is no rule that is satisfied to a lower degree in $\phi(X)$ than in $S|_\sigma$

iff (by definition)

– $S|_\sigma$ is p-preferred to X according to LPOD.

So $S|_\sigma$ is p-preferred to X according to LPOD. Contradiction.

- **Penalty-Sum-Preferred** Since S satisfies rule (4), and satisfies $optimize(ps)$ (in F_s), S must satisfy $better(ps)$. Besides,
– $S \models better(ps)$

iff (by rule (21))

– $\sum_{R:S \models degree(R,D)} D < \sum_{R:S \models degree'(R,D)} D$

iff (by bullet 6)

– the sum of the satisfaction degrees of all rules is smaller in $S|_\sigma$ than in X

iff (by definition)

– $S|_\sigma$ is ps-preferred to X according to LPOD.

So $S|_\sigma$ is ps-preferred to X according to LPOD. Contradiction.

(←) Let X' be a set of atoms in σ such that $\phi(X')$ is an answer set of P, and $P \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup R_A \cup C$ has no answer set. By Proposition 1, X' is a candidate answer set of Π.

Assume for the sake of contradiction that there is a candidate answer set X of Π that is s-preferred to X' according to LPOD. By Proposition 1, $\phi(X)$ is an answer set of P. We will prove that $P \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup R_A \cup C$ has some answer set S, which contradicts with bullet **(d)**. Since $\phi(X)$ is an answer set of P, it is sufficient to prove $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$ has some answer set S, where $H_{\phi(X)}$ reifies the atoms in $\phi(X)$ into the form of $holds(\cdot)$.

First, let Π_{cur} be $H_{\phi(X)} \cup F_s \cup H'_{\phi(X')}$, and consider the answer set of Π_{cur}. Let $a(H_{\phi(X)})$ be $\{holds(a) \mid a \in \phi(X)\}$, let $a(H'_{\phi(X')})$ be $\{holds'(a) \mid a \in \phi(X')\}$, and let $a(F_s)$ denote all the atoms in F_s. It is obvious that $S_1 = a(H_{\phi(X)}) \cup a(F_s) \cup a(H'_{\phi(X')})$ is the only answer set of Π_{cur}.

Second, let's include rule (5) and rule (6) in C into Π_{cur}. By Lemma 2, $S_2 = S_1 \cup \{holds(neg(body_r)) \mid r \in \{1, \ldots, m\}, holds(body_r) \notin a(H_{\phi(X)})\} \cup \{holds'(neg(body_r)) \mid r \in \{1, \ldots, m\}, holds'(body_r) \notin a(H'_{\phi(X')})\}$ is the only answer set of Π_{cur}.

Third, let's include Deg (i.e., rule (13) and rule (14)) into Π_{cur}. By Lemma 2, $S_3 = S_2 \cup \{degree(R, D) \mid S_2$ satisfies the body of rule (13)$\} \cup \{degree'(R, D) \mid S_2$ satisfies the body of rule (14)$\}$ is the only answer set of Π_{cur}. Indeed, analogous to bullet 6 and bullet 7 in the previous direction ($\rightarrow$) of the proof, S_3 satisfies $degree(R, D)$ iff X satisfies LPOD rule R to degree D; S_3 satisfies $degree'(R, D)$ iff X' satisfies LPOD rule R to degree D. In other words, $S_3 = S_2 \cup \{degree(r, d) \mid r \in \{1, \ldots, m\}$, X satisfies rule r to degree $d\} \cup \{degree'(r, d) \mid r \in \{1, \ldots, m\}, X'$ satisfies rule r to degree $d\}$ is the only answer set of Π_{cur}.

Fourth, let's include the rules in E_{t_s} (i.e., $E_{lpod(s)}$ in Section 3.3) into Π_{cur} for each preference criterion $s \in \{i, p, ps\}$. The proof for the case when s is c is not shown due to the limit of space.

- **Inclusion-Preferred** Let's include the first rule (17) in $E_{lpod(i)}$ into Π_{cur}. Since X is i-preferred to X' according to LPOD, there exists an integer d such that
  - the set of LPOD rules that are satisfied to degree d by X' is a proper subset of the set of LPOD rules that are satisfied to degree d by X, and
  - for any degree j such that $j < d$, the set of LPOD rules that are satisfied to degree j by X' is a subset of the set of LPOD rules that are satisfied to degree j by X.

  In other words,there exists an integer d such that
  - there exists at least one LPOD rule j that is satisfied by X to degree d but is not satisfied by X' to degree d, and for any LPOD rule j', if it is satisfied by X' to degree d, it must be satisfied by X to degree d, and
  - for any degree j such that $j < d$, if an LPOD rule is satisfied by X' to degree j, it must by satisfied by X to degree j.

  Since S_3 satisfies $degree(R, D)$ iff X satisfies LPOD rule R to degree D, and S_3 satisfies $degree'(R, D)$ iff X' satisfies LPOD rule R to degree D, there exists an integer d such that

  (i) there exists at least one LPOD rule j such that $S_3 \models degree(j, d)$ and $S_3 \not\models degree'(j, d)$, and for any LPOD rule j', if $S_3 \models degree'(j', d)$, then $S_3 \models degree(j', d)$, and

  (ii) for any LPOD rule $r \in \{1, \ldots, m\}$ and any degree j such that $j < d$, if $S_3 \models degree'(r, j)$, then $S_3 \models degree(r, j)$

  Since S_3 satisfies all atoms in $a(F_s)$, according to the translation, S_3 satisfies $optimize(i)$ an $preference(i, lpod(i))$. By rule (17) and bullet **(i)** above, S_3 satisfies the body of rule (17) (where S is i and D is d). By Lemma 2, $S_3 \cup \{prf2degree(i, d)\}$ is an answer set of Π_{cur}.

  Let's include the second rule (18) in $E_{lpod(i)}$ into Π_{cur}. By Lemma 2 and bullet **(ii)** above, $S_4 = S_3 \cup \{prf2degree(i, d), better(i)\}$ is an answer set of Π_{cur}.

  Let's include rule (4) into Π_{cur}. It is easy to see that S_4 satisfies rule (4). By Lemma 1, S_4 is an answer set of $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$. Contradiction.

- **Pareto-Preferred** Let's include the first rule (19) in $E_{lpod(p)}$ into Π_{cur}. Since X is p-preferred to X' according to LPOD, there is an LPOD rule that is satisfied to a lower degree in X than in X', and there is no rule that is satisfied to a lower degree in X' than in X. By rule (19), S_3 does not satisfy the body of rule (19) (where S is p). By Lemma 2, S_3 is an answer set of Π_{cur}.

  Let's include the second rule (20) in $E_{lpod(p)}$ into Π_{cur}. By Lemma 2, $S_4 = S_3 \cup \{better(p)\}$ is an answer set of Π_{cur}.

  Let's include rule (4) into Π_{cur}. It is easy to see that S_4 satisfies rule (4). By Lemma 1, S_4 is an answer set of $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$. Contradiction.

- **Penalty-Sum-Preferred** Let's include $E_{lpod(ps)}$ into Π_{cur}. Since X is ps-preferred to X' according to LPOD, the sum of the satisfaction degrees of all rules is smaller in X than in X'. Thus $\sum_{R:S_3 \models degree(R,D)} D < \sum_{R:S_3 \models degree'(R,D)} D$. By Lemma 2, $S_4 = S_3 \cup \{better(ps)\}$ is an answer set of Π_{cur}.

  Let's include rule (4) into Π_{cur}. It is easy to see that S_4 satisfies rule (4). By Lemma 1, S_4 is an answer set of $H_{\phi(X)} \cup F_s \cup E_{t_s} \cup H'_{\phi(X')} \cup C$. Contradiction.

Epistemic states, fusion and strategy-proofness

Amílcar Mata Díaz and **Ramón Pino Pérez***
Departamento de Matemáticas, Facultad de Ciencias
Universidad de Los Andes. Mérida, Venezuela.
{amilcarmata, pino}@ula.ve

Abstract

In this work we study the problem of manipulability in belief merging. We adopt the framework of logic-based merging introduced by Konieczny and Pino Pérez (2002) and recently extended by Mata Díaz and Pino Pérez (2017) considering complex epistemic states. We state conditions under which a merging operator of epistemic states admits a manipulator agent, that is, an agent which can change his true beliefs in order to obtain a result which is more convenient for him. Our results adapt some techniques introduced by Mata Díaz and Pino Pérez (2017) concerning representation and impossibility in belief merging of complex epistemic states. These results, together with tools of lifting preferences, allow us to establish a general theorem of manipulability.

Introduction

The aim of belief merging (Konieczny and Pino Pérez 2002; 2011) is to give rational processes for producing a coherent and pertinent piece of information when many sources, which may be mutually in conflict, are present. This kind of process appears naturally in many important domains, for instance decision making, medical diagnosis, policy planning, automatic integration of data, etc. Thus, it is important to understand well the model, its behavior and limitations in order to develop future applications and know in which domains this can be valid.

In the belief merging framework of Konieczny and Pino Pérez the agents' basic piece of information is encoded in propositional logic and the group information is a *bag*. Nevertheless, more complex representations are necessary in many situations. This has been revealed by Darwiche and Pearl (1997) in the case of revision operators with a good behavior with respect to iteration. Mata Díaz and Pino Pérez (2017) have shown the necessity of these representations. The following example illustrates a scenario in which complex representations are very natural:

Example 1 *Anne and Bob have to travel from point A to point B. There are four paths to accomplish the travel: w_1, w_2, w_3 and w_4. Anne thinks that the best way to carry out the travel is w_1. Actually she thinks that w_i is better than w_{i+1} for $i = 1, 2, 3$. Bob thinks that the best way to travel is through w_4 and the other possibilities are equally good. A common consensual result in this situation is to take path w_1, the path that Anne prefers and which is not so bad for Bob, whereas path w_4 is very far from Anne's preferred path. Indeed, the consensual preference is path w_1 as the first option, then path w_2. The last options are indifferently paths w_3 and w_4. Now suppose that Anne and Bob hate heavy traffic. The last news about the traffic inform that there are enormous traffic jams in the paths w_1 and w_4, whereas on paths w_2 and w_3 there is light traffic. With this new information, which can play the role of an integrity constraint, the global preference should be: path w_2 is the first option; the second option is path w_3, followed by path w_1 and finally w_4 is the last option.*

Note also that in the previous example the use of complex integrity constraints can be useful to model the problem. Thus, in this work we adopt this view: the basic pieces of information are some states, called epistemic states, which have attached a logical information, actually a propositional formula. Moreover, the group information is vectorial (or functional, see Section *Preliminaries*).

Some similarities have been pointed out between belief merging and social choice theory (Arrow 1963; Suzumura 2002). These similarities have been established in early works in belief merging (Konieczny and Pino Pérez 1999; 2002; 2005). As a matter of fact, the representation theorem for belief merging operators evoques the methods for defining social choice functions.

Actually, some aspects of social choice are explored in belief merging (Chopra, Ghose, and Meyer 2006; Everaere, Konieczny, and Marquis 2007), mainly impossibility (Arrow 1963; Campbell and Kelly 2002) and strategy-proofness (Gibbard 1973; Satterthwaite 1975). The operators considered by Chopra, Ghose, and Meyer (2006) satisfy the postulate **(IC3)** (the syntax independence postulate) of Konieczny and Pino Pérez (2002). As a matter of fact, this postulate implies a very strong form of anonymity and therefore it is incompatible with the notion of dictator; for this reason they don't have an impossibility result. Moreover, they impose certain conditions to their operators which force strategy-proofness. Everaere, Konieczny, and Marquis (2007) present some results on manipulation of some specific merging operators with respect to certain indexes. Other works about

*Current address: School of Mathematics and Computer Science, Yachay Tech University, Urcuquí, Ecuador. E-mail: rpino@yachaytech.edu.ec

manipulation have been done in the related domain of judgement aggregation by Dietrich and List (2007) [see also (Grossi and Pigozzi 2014)]. Let us note that in judgement aggregation the representation of information is simply propositional. Unlike this, in this work, we continue the study of manipulation in the extended framework of belief merging of epistemic states.

We have to note three important features of the framework considered here. First, that the view of epistemic states considered here is more general than the propositional view. This abstract view of epistemic states, introduced by Benferhat et al. (2000), is indeed a formalization of the concept established by Darwiche and Pearl (1997) [see also Mata Díaz and Pino Pérez (2011)]. Second, the representation of profiles will be also more general; we adopt the functional view. Third, we enrich the set of rational postulates by introducing social postulates inspired on the classical Arrow's postulates in social choice theory (Arrow 1963). All these postulates are formulated in a logical setting.

We have a manipulability situation in merging information, when an agent can change his true information in order to obtain a result that is more convenient for him. We establish some conditions over merging operators which imply the existence of agents who can manipulate. Our first result about manipulation concerns complete operators, that is, operators having as output complete information. Our second result about manipulation concerns general operators. In order to obtain it, we adapt to a logical setting, the lifting technique introduced by Leal and Pino Pérez (2007; 2017) in the framework of social choice theory.

This work is organized as follows: the *Preliminaries* section is devoted to defining the concepts used throughout the paper. The *Epistemic states merging operators* section is devoted to state the syntactical postulates and their semantical counterparts and then to establish the basic representation theorem. In *Social behavior of ES merging operators* section, we present the social postulates in a concise way. In this Section we also present Impossibility results which are key tools for obtaining the results in what follows. In *Manipulation of merging operators* section, we introduce other new social postulates and we establish general results of manipulability – the main results of the paper. In *Concrete examples and properties* section, we give a constructive method for building operators and we analyze its properties. Finally, in *Concluding remarks and perspectives* section, we make some observations about our work and give some paths for future work. For space reasons, proofs of results are not included in detail, but we sketch the main steps.

Preliminaries

A *preorder* over a set A is a binary relation $\succeq$ over A which is reflexive and transitive. We define the strict relation, $\succ$ and the indifference relation, $\simeq$, associated to a preorder $\succeq$ over A as follows: $a \succ b$ iff $a \succeq b$ & $b \not\succeq a$ and $a \simeq b$ iff $a \succeq b$ & $b \succeq a$.

Given a preorder $\succeq$ over A and a subset C of A, we say that c in C is a maximal element of C, with respect to $\succeq$, if there is no $x \in C$ such that $x \succ c$. The set of maximal elements of C with respect to $\succeq$ will be denoted by $\max(C, \succeq)$. We will write $\max(\succeq)$ instead of $\max(A, \succeq)$ to denote the set of maximal elements of the whole set A with respect to $\succeq$. We will denote by $\succeq_{\restriction C}$ the restriction of $\succeq$ to C, and $\mathbb{P}(A)$ will denote the set of preorders over a set A.

A *total preorder* is a binary relation over a set A which is total (therefore reflexive) and transitive. Thus, any total preorder over a set A is also a preorder over A. Let $\succeq_1$ and $\succeq_2$ be two total preorders over A. We define $\succeq^{\text{lex}(\succeq_1,\succeq_2)}$ over A by putting $a \succeq^{\text{lex}(\succeq_1,\succeq_2)} b$ iff $a \succ_1 b$ or $a \simeq_1 b$ and $a \succeq_2 b$.

The set of propositional formulas built over a finite set $\mathcal{P}$ of atomic propositions will be denoted $\mathcal{L}_\mathcal{P}$. $\mathcal{L}_\mathcal{P}{}^*$ will denote the set of non contradictory formulas in $\mathcal{L}_\mathcal{P}$, while $\mathcal{W}_\mathcal{P}$ will be the set of all the interpretations. Note that $\mathcal{W}_\mathcal{P}$ is a finite set. If φ is a formula in $\mathcal{L}_\mathcal{P}$, we denote by $[\![\varphi]\!]$ the set of its models, *i.e.* $[\![\varphi]\!] = \{w \in \mathcal{W}_\mathcal{P} : w \models \varphi\}$. If φ_i is a formula in $\mathcal{L}_\mathcal{P}$, for each i in a finite set of indexes I, then we denote by $\bigwedge_{i\in I} \varphi_i$ the conjunction of all the formulas φ_i. If I is a nonempty set of interpretations, we denote by φ_I a formula such that $[\![\varphi_I]\!] = I$. When I is $i\{w\}$ or $\{w, w'\}$, φ_I will be denoted φ_w or $\varphi_{w,w'}$ respectively.

An epistemic space is a triple $(\mathcal{E}, B, \mathcal{L}_\mathcal{P})$ such that $\mathcal{E}$ is a nonempty set, $\mathcal{L}_\mathcal{P}$ is the set of propositional formulas over $\mathcal{P}$ and B is a function from $\mathcal{E}$ into $\mathcal{L}_\mathcal{P}$, such that the image of B, modulo logical equivalence, is all the set $\mathcal{L}^*_\mathcal{P}$. The elements of $\mathcal{E}$ are called *epistemic states*, B is called the *belief function*, while $B(E)$ is called the *belief base* (or *the most entrenched beliefs*) of E, for each E in $\mathcal{E}$. Note that if I is a finite nonempty set of interpretations, then there exists an epistemic state E such that $B(E) \equiv \varphi_I$.

Extending a binary relation over interpretations to binary relations over propositional formulas can be performed, via the semantics, in a way which is similar to the extension of an ordering over a set A to an ordering over subsets of A [see for instance (Fishburn 1972; Gärdenfors 1976; Kelly 1977) for manipulability issues, (Barberà, Bossert, and Pattanaik 2004) for general properties and more recently the works (Brandt 2011; Brandt and Brill 2011)]. We call this kind of process *liftings over formulas*. More precisely, a *lifting over formulas* is an application $\succeq \mapsto \sqsupseteq_\succeq$ which maps any total preorder $\succeq$ over $\mathcal{W}_\mathcal{P}$ into a preorder $\sqsupseteq_\succeq$ over $\mathcal{L}_\mathcal{P}$, which satisfies the following: $\varphi_w \sqsupseteq_\succeq \varphi_{w'}$ iff $w \succeq w'$, and if $\varphi \equiv \psi$ then $\varphi \simeq \psi$ (with respect to $\sqsupseteq_\succeq$).

Given a preorder $\sqsupseteq^{\mathcal{W}_\mathcal{P}}_\succeq$ over subsets of $\mathcal{W}_\mathcal{P}$, it is possible to establish an order over $\mathcal{L}_\mathcal{P}$, $\sqsupseteq^{\mathcal{L}_\mathcal{P}}_\succeq$, as follows:

$$\varphi \sqsupseteq^{\mathcal{L}_\mathcal{P}}_\succeq \varphi' \Leftrightarrow [\![\varphi]\!] \sqsupseteq^{\mathcal{W}_\mathcal{P}}_\succeq [\![\varphi']\!]$$

Thus, through liftings over sets, like the Fishburn lifting (Fishburn 1972), the Gärdenfors lifting (Gärdenfors 1976), the Kelly lifting (Kelly 1977), the *Possibilistic lifting* introduced by Dubois, Prade, and Lang (1994), and the *Precise leximax lifting*[1], defined by Leal and Pino Pérez (2017), it is possible to define liftings over formulas.

Barberà, Bossert, and Pattanaik (2004) have characterized many natural liftings through their properties. Among these

[1]The Precise leximax lifting is a variant of the leximax lifting presented by Barberà, Bossert, and Pattanaik (2004).

properties we can find a couple of very basic properties that together are called *Simple dominance*. We adapt these properties to liftings over formulas.

Simple Dominance 1: $\succeq \mapsto \sqsupseteq_{\succeq}$ satisfies *simple dominance 1 iff* for each total preorder $\succeq$ over $\mathcal{W}_{\mathcal{P}}$ and any pair w, w' in $\mathcal{W}_{\mathcal{P}}$: $w \succ w' \Rightarrow \varphi_{\{w,w'\}} \sqsupset_{\succeq} \varphi_{\{w'\}}$

Simple Dominance 2: $\succeq \mapsto \sqsupseteq_{\succeq}$ satisfies *simple dominance 2 iff* for each total preorder $\succeq$ over $\mathcal{W}_{\mathcal{P}}$ and any pair w, w' in $\mathcal{W}_{\mathcal{P}}$: $w \succ w' \Rightarrow \varphi_{\{w\}} \sqsupset_{\succeq} \varphi_{\{w,w'\}}$

These properties of companionship have a very natural interpretation: simple dominance 1 means that the good company improves the group; simple dominance 2 means that the bad company worsens the group. Its is worth noting that all the previously cited liftings satisfy both instances of simple dominance, except the possibilistic lifting [cf. Leal and Pino Pérez (2017)].

In order to introduce the notion of *epistemic state profiles*, from now on we consider a given epistemic space $(\mathcal{E}, B, \mathcal{L}_{\mathcal{P}})$ and a well ordered set $(\mathcal{S}, <)$ the elements[2] of which are called *agents*. A *finite society of agents* is a finite and nonempty set N of $\mathcal{S}$. $\mathcal{F}^*(\mathcal{S})$ will denote the set of finite societies of agents. From now on, we suppose $N = \{i_1, i_2, \ldots, i_n\}$ and also assume that its elements are disposed in increasing way, *i.e.* $i_k < i_m$ whenever $k < m$. A partition of N is a finite family $\{N_1, N_2, \ldots, N_k\}$ of pairwise disjoint sets such that their union is N.

Given a finite society of agents N, an *N-profile of epistemic states* (also called *N-profile* or *epistemic profile* by abuse of notation) is a function $\Phi : N \longrightarrow \mathcal{E}$. We think of $\Phi(i)$ as the epistemic state of the agent i, for each agent i in N. If Φ is an N-profile, for each agent i in N, E_i will denote $\Phi(i)$. Thus, if $N = \{i_1, i_2, \ldots, i_n\}$ is a finite society of agents, it can be seen as an ordered tuple: $\Phi = (E_{i_1}, E_{i_2}, \ldots, E_{i_n})$. Thus, an N-profile Φ collects in an ordered way the information expressed by those agents in N. From now on, if N is a singleton, suppose $N = \{i\}$, by abuse of notation, E_i will denote the N-profile $\Phi = (E_i)$. In that case we write i-profile instead of $\{i\}$-profile. The set of all the epistemic profiles will be denoted $P(\mathcal{S}, \mathcal{E})$.

Let $N = \{i_1, i_2, \ldots, i_n\}$ and $M = \{j_1, j_2, \ldots, j_m\}$ be two finite societies of agents, and consider the profiles $\Phi = (E_{i_1}, E_{i_2}, \ldots, E_{i_n})$ and $\Phi' = (E'_{j_1}, E'_{j_2}, \ldots, E'_{j_m})$. We say that Φ and Φ' are equivalent, denoted $\Phi \equiv \Phi'$, if $n = m$ and $E_{i_k} = E_{j_k}$ for $k = 1, \ldots, n$. Thus, for any pair of agents i, j in $\mathcal{S}$, if we consider an i-profile E_i and a j-profile E_j, then $E_i \equiv E_j$ *iff* , seen as epistemic states, we have $E_i = E_j$. Thus, by abuse of notation and being clear from the context, we write $E_i = E_j$ and $E_i \neq E_j$ instead of $E_i \equiv E_j$ and $E_i \not\equiv E_j$ respectively.

When N and M are disjoint, Φ is an N-profile and Φ' is an M-profile, we define a new $(N \cup M)$-profile, the joint of Φ and Φ', denoted $\Phi \sqcup \Phi'$, as follows: $(\Phi \sqcup \Phi')(i)$ is $\Phi(i)$ if $i \in N$, otherwise it is $\Phi'(i)$. Moreover, if $M \subseteq N$, then $\Phi \restriction_M$ will denote the M-profile obtained by the restriction of Φ to M. Thus, if $\{N_1, N_2, \ldots, N_k\}$ is a partition of a finite society of agents N, then we have $\Phi = \Phi \restriction_{N_1} \sqcup \cdots \sqcup \Phi \restriction_{N_k}$, for each N-profile Φ.

For a given epistemic state E^* and an N-profile Φ, we denote by $\Phi[E^*/i]$ the N-profile obtained from Φ by replacing E_i, the epistemic state associated with the agent i through Φ, with E^*. More precisely: $\Phi[E^*/i](j)$ is $\Phi(j)$ if $j \neq i$, otherwise it is E^*.

From now on, we will suppose that N is the finite society $\{i_1, i_2, \ldots, i_n\}$, while the N-profiles Φ and Φ' will be denoted by $(E_{i_1}, E_{i_2}, \ldots, E_{i_n})$ and $(E'_{i_1}, E'_{i_2}, \ldots, E'_{i_n})$ respectively.

i

Epistemic states merging operators

Let us fix an epistemic space $(\mathcal{E}, B, \mathcal{L}_{\mathcal{P}})$ and a set of agents $\mathcal{S}$. An *epistemic state operator* is a function of the form $\nabla : P(\mathcal{S}, \mathcal{E}) \times \mathcal{E} \longrightarrow \mathcal{E}$, also called an *ES operator* for short. $\nabla(\Phi, E)$ represents the result of combining the epistemic states in Φ under an *integrity constraint* E.

Now we establish the rationality postulates of merging in the framework of epistemic states. Most of them are adapted from IC merging postulates proposed by Konieczny and Pino Pérez (1999; 2002; 2005; 2011). The postulates in terms of epistemic states, were first proposed and widely studied by Mata Díaz and Pino Pérez (2011; 2017). In order to introduce such postulates, we consider N and M a couple of finite societies of agents in $\mathcal{S}$, an N-profile Φ, an M-profile Φ', and a triple of epistemic states E, E', E'' in $\mathcal{E}$; let j, k be a pair of agents in $\mathcal{S}$, N_1, N_2 be any partition of N, E_j be any j-profile, E_k be any k-profile.

(ESF1) $B\big(\nabla(\Phi, E)\big) \vdash B(E)$.

(ESF2) If $\Phi \equiv \Phi'$ and $B(E) \equiv B(E')$ then $B\big(\nabla(\Phi, E)\big) \equiv B\big(\nabla(\Phi', E')\big)$.

(ESF3) If $B(E) \equiv B(E') \wedge B(E'')$, then $B\big(\nabla(\Phi, E')\big) \wedge B(E'') \vdash B\big(\nabla(\Phi, E)\big)$.

(ESF4) If $B(E) \equiv B(E') \wedge B(E'')$ and $B\big(\nabla(\Phi, E')\big) \wedge B(E'') \not\vdash \bot$, then $B\big(\nabla(\Phi, E)\big) \vdash B\big(\nabla(\Phi, E')\big) \wedge B(E'')$.

(ESF5) If $E_j \neq E_k$, then there exists E' in $\mathcal{E}$ such that $B\big(\nabla(E_j, E')\big) \not\equiv B\big(\nabla(E_k, E')\big)$

(ESF6) If $\bigwedge_{i \in N} B(E_i) \wedge B(E) \not\vdash \bot$, then $B\big(\nabla(\Phi, E)\big) \equiv \bigwedge_{i \in N} B(E_i) \wedge B(E)$

(ESF7) $B\big(\nabla(\Phi \restriction_{N_1}, E)\big) \wedge B\big(\nabla(\Phi \restriction_{N_2}, E)\big) \vdash B\big(\nabla(\Phi, E)\big)$

(ESF8) If $B\big(\nabla(\Phi \restriction_{N_1}, E)\big) \wedge B\big(\nabla(\Phi \restriction_{N_2}, E)\big) \not\vdash \bot$, then $B\big(\nabla(\Phi, E)\big) \vdash B\big(\nabla(\Phi \restriction_{N_1}, E)\big) \wedge B\big(\nabla(\Phi \restriction_{N_2}, E)\big)$

(ESF8W) If $B(\nabla(\Phi \restriction_{N_1}, E)) \wedge B(\nabla(\Phi \restriction_{N_2}, E)) \not\vdash \bot$ then $B(\nabla(\Phi, E)) \vdash B(\nabla(\Phi \restriction_{N_1}, E)) \vee B(\nabla(\Phi \restriction_{N_2}, E))$

The first four postulates, **(ESF1)**-**(ESF4)**, called *basic merging postulates*, are considered the minimal requirements of rationality that the operators have to satisfy. These postulates allow us to introduce an important class of ES operators, namely the *ES basic merging operators*. They

[2] The set $(\mathcal{S}, <)$ can be identified with the set of natural numbers $\mathbb{N}$ with the usual order.

are generalizations of ICO, IC3, IC7 and IC8 respectively[3]. **(ESF6)**, **(ESF7)** and **(ESF8)** are generalizations of IC2, IC5 and IC6 respectively. **(ESF8W)** is a weakening of **(ESF8)**. **(ESF5)** says that the operators have a sort of injectivity for profiles of size one. The last four postulates describe mainly the relationships between the results of merging as a whole society and the results of merging in its subsocieties.

The only postulate that we do not consider from the propositional belief merging framework is IC4, the so called fairness postulate. The reason is that IC4 imposes heavy restrictions over the representation of epistemic states. Moreover, there are a lot of interesting merging operators which don't satisfy IC4. Thus, including IC4 should lead to exclude them.

Definition 1 *An ES operator ∇ is said to be an epistemic state basic merging operator (*ES basic merging operator *for short) if it satisfies* **(ESF1)**-**(ESF4)**.

We will say that an ES basic merging operator ∇ is complete *if , $B(\nabla(\Phi, E))$ has a single model, for every N in $\mathcal{F}^*(\mathcal{S})$, each N-profile Φ and each epistemic state E in $\mathcal{E}$.*

An ES basic merging operator ∇ is said to be a merging operator of epistemic states *(*quasi-merging operator of epistemic states*)* -ES merging operator (ES quasi-merging operator) *for short- if it satisfies the postulates* **(ESF5)**-**(ESF8)** *(***(ESF5)**-**(ESF7)** *and* **(ESF8W)***).*

Now we present some semantic aspects. An *assignment* is a function $\Phi \mapsto \succeq_\Phi$ which maps epistemic states profiles into total preorders over interpretations. The intended meaning of these mappings is coding semantically, in some sense, the group preference.

Definition 2 *A* basic assignment *is an application $\Phi \mapsto \succeq_\Phi$ which maps each epistemic profile Φ into a total preorder $\succeq_\Phi$ over $\mathcal{W}_\mathcal{P}$, and it is such that $\succeq_\Phi = \succeq_{\Phi'}$, for any pair Φ, Φ' of equivalent epistemic profiles.*

A basic assignment $\Phi \mapsto \succeq_\Phi$ is said to be linear *if $\succeq_\Phi$ is a linear order over $\mathcal{W}_\mathcal{P}$, for each epistemic profile Φ.*

A (linear) basic assignment $\Phi \mapsto \succeq_\Phi$ is called a (linear) faithful assignment *if it satisfies the following properties for any pair j, k of agents in $\mathcal{S}$, any finite society of agents N, any partition of N, $\{N_1, N_2\}$, any N-profile Φ, any j-profile E_j, any k-profile E_k, and any pair of interpretations w, w' in $\mathcal{W}_\mathcal{P}$:*

1 *If $E_j \neq E_k$, then $\succeq_{E_j} \neq \succeq_{E_k}$*

2 *If $\bigwedge_{i \in N} B(E_i) \not\vdash \bot$, then $[\![\bigwedge_{i \in N} B(E_i)]\!] = \max(\succeq_\Phi)$*

3 *If $w \succeq_{\Phi\restriction_{N_1}} w'$ and $w \succeq_{\Phi\restriction_{N_2}} w'$ then $w \succeq_\Phi w'$*

4 *If $w \succeq_{\Phi\restriction_{N_1}} w'$ and $w \succ_{\Phi\restriction_{N_2}} w'$, then $w \succ_\Phi w'$*

A (linear) basic assignment which satisfies **1**-**3** *plus the following property*

4' $w \succ_{\Phi\restriction_{N_1}} w'$ & $w \succ_{\Phi\restriction_{N_2}} w' \Rightarrow w \succ_\Phi w'$

is called a (linear) quasi-faithful assignment.

For any epistemic profile Φ, the total preorder $\succeq_\Phi$ can be seen as a global plausibility measure over worlds:

- If $w \succeq_\Phi w'$, we will say that w is *at least as plausible as* w', for the agents group in Φ
- If $w \succ_\Phi w'$, we will say that w is *more plausible than* w', for the agents group in Φ

From Property **2** follows that the most entrenched preferences represent the entrenchment beliefs of any agent. More precisely, for any agent i in $\mathcal{S}$ and for each i-profile E_i, we have straightforwardly from Property **2** that the following equality holds, which is called *maximality condition*: $[\![B(E_i)]\!] = \max(\succeq_{E_i})$.

Thus, any assignment which satisfies Property **2**, satisfies the maximality condition, but the converse is not true [cf. Mata Díaz and Pino Pérez (2011; 2017)]. However, in presence of Properties **3** and **4**, we have that Property **2** is equivalent to the maximality condition, as was shown by Mata Díaz and Pino Pérez (2017).

Now we present some results that help to understand the behavior of the ES merging operators. They allow us to describe such operators at the level of the entrenched beliefs.

Theorem 1 *(i) An ES operator ∇ is an ES basic merging operator* iff *there exists a unique basic assignment $\Phi \mapsto \succeq_\Phi$ such that:*

$$[\![B(\nabla(\Phi, E))]\!] = max([\![B(E)]\!], \succeq_\Phi) \qquad \text{(B-Rep)}$$

(ii) ∇ is a complete ES basic merging operator iff the assignment $\Phi \mapsto \succeq_\Phi$, associated to ∇ by (i), is a linear basic assignment.

(iii) An ES operator ∇ is an ES (complete) (quasi) merging operator iff there exists a unique (linear) (quasi) faithful assignment $\Phi \mapsto \succeq_\Phi$ satisfying (B-Rep).

Sketch of proof: (i) is proved by Mata Díaz and Pino Pérez (2017). Actually they proved also (iii) without the parts between the parentheses. (ii) follows from (i) and the fact that completeness corresponds exactly to the fact that the assignment is linear. Now (iii) with the parentheses follows from (ii) and (iii) without the parts between the parentheses. ∎

The parts of the previous result concerning complete operators are new.

Social behavior of ES merging operators

We present some properties of the merging process which capture the following social principles appearing in the seminal work of Arrow (1963): *Domain Standard, Pareto Condition, Independence of Irrelevant Alternatives and Existence of a Dictator*.

In order to give a formulation of these principles in logic terms using epistemic states, we will suppose from now on that $\mathcal{P}$ has at least two propositional variables. Thus, there will be available at least four interpretations in $\mathcal{W}_\mathcal{P}$.

The first property, called *Standard Domain*, states some "richness" in the set of results of the merging process.

(ESF-SD) For any agent i in $\mathcal{S}$, for every triple w, w' and w'' in $\mathcal{W}_\mathcal{P}$, and each pair of epistemic states $E_{w,w'}$ and $E_{w',w''}$, such that $[\![B(E_{w,w'})]\!] = \{w, w'\}$ and $[\![B(E_{w',w''})]\!] = \{w', w''\}$, the following conditions hold:

[3]The well known postulates of belief merging appearing in (Konieczny and Pino Pérez 2002)

(i) There exists an i-profile E_i such that $B\big(\nabla(E_i, E_{w,w'})\big) \equiv \varphi_{w,w'}$ and $B\big(\nabla(E_i, E_{w',w''})\big) \equiv \varphi_{w'}$

(ii) There exists an i-profile E_i such that $B\big(\nabla(E_i, E_{w,w'})\big) \equiv \varphi_w$ and $B\big(\nabla(E_i, E_{w',w''})\big) \equiv \varphi_{w',w''}$

(iii) There exists an i-profile E_i such that $B\big(\nabla(E_i, E_{w,w'})\big) \equiv \varphi_w$ and $B\big(\nabla(E_i, E_{w',w''})\big) \equiv \varphi_{w'}$

It is worth to note that complete ES operators can only satisfy (iii) in **(ESF-SD)**. Thus, by abuse of notation and the context being clear, we will say that a complete ES operator ∇ satisfies **(ESF-SD)** if ∇ satisfies such instance. It is also important to note the following:

Observation 1 *For basic merging operators, the satisfaction of this postulate is equivalent to the fact that, for any agent i in $\mathcal{S}$, any triple of interpretations w, w' and w'' in $\mathcal{W}_\mathcal{P}$, and any total preorder $\succeq$ over interpretations (except the flat order), there is an i-profile E_i such that $\succeq = \succeq_{E_i} \restriction_{\{w,w',w''\}}$.*

The following property tries to capture the meaning of the unanimity principle. It is given in terms of similarity between the epistemic states of the agents involved in the merging process. This is called *unanimity condition*:

(ESF-U) For each finite society of agents N, for every N-profile Φ and for each epistemic state E, if $E_i = E_j$, for any pair i, j in N, then, for each i in N, $B\big(\nabla(\Phi, E)\big) \equiv B\big(\nabla(E_i, E)\big)$.

Any ES operator ∇ satisfying **(ESF2)**, **(ESF7)** and **(ESF8)** also satisfies **(ESF-U)**. Thus, any ES merging operator satisfies unanimity condition. However, in general the converse of this result is not true [cf. Mata Díaz and Pino Pérez (2017)].

Now we give a syntactical formulation of the *Pareto condition*: if all the agents reject a given information, and all the agents have some consensus, then this information will be rejected in the result of merging.

(ESF-P) For every finite society N in $\mathcal{F}^*(\mathcal{S})$, any N-profile Φ and any couple of epistemic states E, E' in $\mathcal{E}$, if $\bigwedge\!\!\!\bigwedge_{i\in N} B\big(\nabla(E_i, E)\big) \not\vdash \bot$ and, for all i in N, $B\big(\nabla(E_i, E)\big) \wedge B(E') \vdash \bot$, then $B\big(\nabla(\Phi, E)\big) \wedge B(E') \vdash \bot$.

Any ES merging operator satisfies the Pareto Condition. This is because the satisfaction of **(ESF7)** and **(ESF8)** entails that **(ESF-P)** holds. However, there exist some ES operators showing that the converse of this result is not true [cf. Mata Díaz and Pino Pérez (2017)].

We continue by stating the syntactical postulate aiming to capture the principle of *independence of irrelevant alternatives*: the merging process depends only on how the restrictions in the individual epistemic states are related. Our postulate, called *independence condition*, tries to capture this.

(ESF-I) For any finite society N in $\mathcal{F}^*(\mathcal{S})$, for any couple of N-profiles Φ and Φ', for every epistemic state E in $\mathcal{E}$ we have $B\big(\nabla(\Phi, E)\big) \equiv B\big(\nabla(\Phi', E)\big)$, if for every epistemic state E' in $\mathcal{E}$ such that $B(E') \vdash B(E)$ we have $B\big(\nabla(E_j, E')\big) \equiv B\big(\nabla(E'_j, E')\big)$, for all j in N.

Next, we state the postulate related with the fourth principle, *Absence of Dictator*: the group belief base, obtained as the result of the merging process, does not depend on one unique agent. Actually, we establish the negative form: there is an agent that imposes his will, a *dictatorial agent*. This is the postulate that the *good* operators should avoid.

(ESF-D) For any finite society N in $\mathcal{F}^*(\mathcal{S})$ there exists an agent d_N in N such that, for any N-profile Φ and for any epistemic state E in $\mathcal{E}$, $B\big(\nabla(\Phi, E)\big) \vdash B\big(\nabla(E_{d_N}, E)\big)$.

Operators satisfying the previous postulate are called *dictatorial operators*. Thus, if ∇ is a dictatorial operator, given a finite society N, an agent d_N testifying that **(ESF-D)** holds, is called a *dictator in N* or simply an *N-dictator* for ∇.

It is worth to note that the social postulates here presented have a semantical characterization [cf. Mata Díaz and Pino Pérez (2017)]. Actually, Observation 1 can be seen as the semantical characterization of the standard domain condition.

Now we present a general result of Impossibility, similar to Arrow's Theorem (Arrow 1963; Kelly 1978; Taylor 2005):

Theorem 2 (Main impossibility theorem) *If ∇ is an ES basic (complete) merging operator that satisfies the properties* **(ESF-SD)**, **(ESF-P)** *and* **(ESF-I)**, *then ∇ also satisfies* **(ESF-D)**.

Note that the proof of this result was presented by Mata Díaz and Pino Pérez (2017) for operators which are not complete. An analysis of that proof reveals that this result is true, even if the operator is complete.

Because of the relationships between the merging postulates and the Pareto condition pointed out before, we have:

Corollary 1 *(i) Any ES basic merging operator that satisfies* **(ESF-SD)**, **(ESF7)**, **(ESF8W)** *and* **(ESF-I)**, *is a dictatorial operator.*
(ii) Any ES basic merging operator that satisfies **(ESF-SD)**, **(ESF-U)** *and* **(ESF-I)**, *is a dictatorial operator.*

Manipulation of merging operators

In this section we deal with another interesting problem which is present in the framework of logic-based merging, namely, manipulability of merging information processes.

We establish three new classes of ES operators, after introducing some other natural social properties: *Non-imposition* , *stability* and *strategy-proofness*.

We first introduce *Non-imposition* property. This property expresses that any complete information (formula) can be obtained as the beliefs of the output if it satisfies the restriction of the merging process.

(ESF-NI) For all finite society N in $\mathcal{F}^*(\mathcal{S})$, for each epistemic state E and every model w of $B(E)$, there exists an N-profile Φ such that $B\big(\nabla(\Phi, E)\big) \equiv \varphi_w$.

The following result shows that, in addition to the basic merging postulates and the maximality condition, the Pareto condition entails non-imposition.

Proposition 1 *Let ∇ be an ES basic merging operator and suppose that $\Phi \mapsto \succeq_\Phi$ (the basic assignment associated to ∇ by Theorem 1) satisfies the maximality condition. If ∇ satisfies* **(ESF-P)***, then ∇ satisfies* **(ESF-NI)**.

Sketch of proof: Consider an N-profile Φ where, for each agent $i \in N$, we have $B(E_i) = \varphi_w$ for a fixed w, which is a model of $B(E)$. Then, applying the representation, the maximality and Pareto conditions, we have $B(\nabla(\Phi, E)) = \varphi_w$. ∎

Another natural property is given in terms of *stability* with respect to changes in the pieces of information. More precisely, at level of entrenched beliefs, the stability property expresses that if a new piece of information (epistemic state) is "revised" with the integrity constraints and it coincides with the result obtained from a merging process of a profile and the integrity constraint, then any agent can change its mind with this new piece of information and the result obtained is the same as the initial result.

(ESF-St) For each finite society N in $\mathcal{F}^*(\mathcal{S})$, every N-profile Φ, any agent i in N and every couple of epistemic states E, E^* in $\mathcal{E}$, if $B\big(\nabla(\Phi, E)\big) \equiv B\big(\nabla(E^*, E)\big)$ then $B\big(\nabla(\Phi[E^*/i], E)\big) \equiv B\big(\nabla(\Phi, E)\big)$.

An ES basic merging operator is manipulable by an agent if this agent, knowing the restriction of the merging process and the information that will be expressed by the other agents, changes its epistemic state in order to obtain a result that "fits" better his true beliefs. More formally:

Definition 3 (Manipulable ES operators) *Let $\succeq \mapsto \sqsupseteq_\succeq$ be a lifting over formulas in $\mathcal{L}_\mathcal{P}$. An ES basic merging operator ∇ is said to be* manipulable *with respect to $\succeq \mapsto \sqsupseteq_\succeq$, if there exists a finite society of agents N in $\mathcal{F}^*(\mathcal{S})$, an N-profile Φ, an agent i in N and a couple of epistemic states E, E^*, such that:*

$$B\big(\nabla(\Phi[E^*/i], E)\big) \sqsupset_{\succeq_{E_i}} B\big(\nabla(\Phi, E)\big)$$

If ∇ does not satisfy the previous statement, we will say that ∇ is strategy-proof with respect to the lifting $\succeq \mapsto \sqsupseteq_\succeq$.

An ES operator ∇ is said to be strategy-proof if it is strategy-proof with respect to any lifting.

The next result shows interesting links between some of the previous properties.

Proposition 2 *A complete ES basic merging operator is strategy-proof iff it satisfies* **(ESF-St)**.

Sketch of proof: If the property **(ESF-St)** doesn't hold we have a situation of manipulability. Conversely, if we have a situation of manipulability, then we can build a counter-example to the property **(ESF-St)**. ∎

Now we introduce a variant of dictatorial ES merging operators through a new property. These operators will be called *Weak dictatorial ES operators*. As we will see, operators of this type admit an agent that, in a similar way to dictators, imposes its beliefs in the results of the merging process, but in a weak sense. More precisely, given any complete formula, the weak dictator can adopt an epistemic state such that, independently of the epistemic states expressed by the other agents, the complete formula will satisfy the entrenched beliefs resulting from the merging process. More precisely:

(ESF-WD) For every finite society N in $\mathcal{F}^*(\mathcal{S})$, there exists an agent d_N in N such that for each interpretation w in $\mathcal{W}_\mathcal{P}$, there exists an epistemic state E^w in $\mathcal{E}$ where $w \models B\big(\nabla(\Phi[E^w/d_N], E)\big)$, for any N-profile Φ and any epistemic state E in $\mathcal{E}$, with $w \models B(E)$.

If ∇ is a weak dictatorial operator, an agent d_N in a finite society N that testifies the satisfaction of such property is called a *weak dictador* in N for ∇, or simply a *weak N-dictator* for ∇.

Note that the notion of weak dictator is an inclusive one: he can force to have a world in the result under the condition that his beliefs has this world as unique model.

The following result shows that, under some rational conditions, dictatorial operators are also weak dictatorial operators.

Proposition 3 *Let ∇ be an ES basic merging operator and suppose that $\Phi \mapsto \succeq_\Phi$ (the basic assignment associated to ∇ by Theorem 1) satisfies the maximality condition. If ∇ is a dictatorial operator then ∇ is also a weak dictatorial operator. Moreover, for each finite society N, if d_N is an N-dictator for ∇, then d_N is also a weak N-dictator for ∇.*

Sketch of proof: Given w, a model of $B(E)$, it is enough to check that if the dictator d_N has an epistemic state with belief φ_w, the output of the merging will have w among its models. ∎

By Theorem 2 and Proposition 3, we have that any ES basic merging operator that satisfies **(ESF-SD)**, **(ESF-P)** and **(ESF-I)** is a weak dictatorial operator.

Now, we present two results concerning manipulability. The first one is similar to the result presented in social choice theory by Gibbard (1973) and Satterthwaite (1975); the second one is similar to that presented by Leal and Pino Pérez (2017). Actually, that result is a weak version of Barberà-Kelly's Theorem (Barberà 1977; Kelly 1977) [see (Leal and Pino Pérez 2017)].

Our first general result is a characterization of strategy-proofness for complete and non-imposed operators: they are exactly the dictatorial ones.

Theorem 3 (Strategy-proofness of complete operators) *Let ∇ be a complete ES basic merging operator satisfying* **(ESF-SD)** *and suppose that $\Phi \mapsto \succeq_\Phi$, the basic assignment associated to ∇ by Theorem 1, satisfies the maximality condition, restricted to single-profiles with complete beliefs. Then, ∇ is a non-imposed and strategy-proof operator if, and only if, ∇ is a dictatorial operator.*

Sketch of proof: The if part is easy. For the only if part, we prove that non-imposed and non-manipulable complete basic ES merging operators satisfy **(ESF-P)** and **(ESF-I)**, then we can apply Theorem 2 and conclude that there is a dictator. ∎

Theorem 4 (Strategy-proofness through liftings) *Let ∇ be an ES basic merging operator satisfying* **(ESF-SD)***, such*

that the basic assignment $\Phi \mapsto \succeq_\Phi$, associated to ∇ by Theorem 1, satisfies the maximality condition, restricted to single-profiles with complete beliefs. Let $\succeq \mapsto \sqsupseteq_\succeq$ be a lifting over formulas that satisfies both instances of simple dominance. If ∇ is non-imposed and strategy-proof with respect to $\succeq \mapsto \sqsupseteq_\succeq$, then ∇ is a weak dictatorial operator.

Sketch of proof: The first step is to transform ∇ into a complete ES basic merging operator ∇'. This is done by transforming the basic assignment $\Phi \mapsto \succeq_\Phi$ associated to ∇ into a linear assignment via a fixed linear order over interpretations. More precisely, fix $\geq$, a linear order over $\mathcal{W}_\mathcal{P}$ and consider the assignment $\Phi \mapsto \succeq^{\geq}_\Phi$, where $w \succeq^{\geq}_\Phi w'$ iff $w \succ_\Phi w'$ or $w \simeq_\Phi w'$ and $w \geq w'$. For each model w in $\mathcal{W}_\mathcal{P}$, consider an epistemic state E_w in $\mathcal{E}$ such that $[\![B(E_w)]\!] = \{w\}$. Then define ∇' as follows:

$$\nabla'(\Phi, E) = E_w \textit{ iff } \max([\![B(E)]\!], \succeq^{\geq}_\Phi) = \{w\}$$

Then we establish the following:
Claim 1. ∇' satisfies **(ESF-SD)**.
Claim 2. ∇' is non-imposed.
Claim 3. ∇' is strategy-proof.
Claim 4. ∇' is dictatorial.
Claim 5. A dictator for ∇' is a weak dictator for ∇.

Claim 1 follows from the fact that ∇ satisfies **(ESF-SD)**. Claim 2 follows from the fact that ∇ is non-imposed. To prove Claim 3, we suppose that ∇' is manipulable. Then we use the manipulation for ∇' and the properties of simple dominance to create a manipulation situation for ∇. Claim 4 follows from the three first claims and Theorem 3. Finally, Claim 5 is easy to see because of definition of ∇'. ∎

Concrete examples and properties

In this section, we are going to present some specific ES-merging operators in order to illustrate manipulability situations. Moreover, the study of properties satisfied by the operators summarized in Table 1 is important because it allows to see some relations between these properties, in particular when a given property is not entailed by a set of other properties. Unfortunately we don't have the place to include all the justifications. In some cases Theorems 3 and 4 are very useful to establish some properties.

We assume that the epistemic states are total preorders over interpretations – that is, $\mathcal{E} = \mathbb{P}(\mathcal{W}_\mathcal{P})$ – and we have that $[\![B(\succeq)]\!] = \max(\succeq)$ for any epistemic state $\succeq$. Some of the operators presented herein were introduced in (Mata Díaz and Pino Pérez 2017), others are new.

We are going to present a class of ES basic merging operators using aggregation functions. These operators are called *aggregation-based ES merging operators*. This kind of merging operators was introduced by Konieczny and Pino Pérez (1999; 2002; 2005; 2011; 2013) and more formally studied by Konieczny, Lang, and Marquis (2004) in the context of belief bases. They have been extendeded to the ESF framework by Mata Díaz and Pino Pérez (2017). Let us recall the definition of aggregation functions.

Definition 4 (Aggregation functions) *An (symmetric) aggregation function F is a total function associating a non-negative real number to every finite tuple of nonnegative integers such that for any $x_1, ..., x_n, x, y$ in $\mathbb{Z}^+$:*

Monotony $F(x_1, \ldots, x, \ldots, x_n) \geq F(x_1, \ldots, y, \ldots, x_n)$, *if* $x \geq y$

Minimality $F(x_1, x_2, \ldots, x_n) = 0$ *if and only if* $x_1 = x_2 = \cdots = x_n = 0$

Identity $F(x) = x$

Symmetry *For any permutation* σ, $F\big(x_1, x_2, \ldots, x_n\big) = F\big(\sigma(x_1, x_2, \ldots, x_n)\big)$

Two examples of aggregation functions are *sum* and *max*. They are classical examples in the study of logic-based merging (Konieczny and Pino Pérez 1999; 2002; 2005; 2011; 2013; Mata Díaz and Pino Pérez 2017).

sum $\sum(x_1, x_2, \cdots, x_n) = \sum x_i$

max $\max(x_1, x_2, \ldots, x_n) = \max\{x_1, x_2, \ldots, x_n\}$

An aggregation function F induces a total preorder over $\mathcal{W}_\mathcal{P}$, for any profile of epistemic states in the following way:

For any $N = \{i_1, i_2, \ldots, i_n\}$ in $\mathcal{F}^*(\mathcal{S})$, any N-profile $\Phi = (\succeq_{i_1}, \succeq_{i_2}, \ldots, \succeq_{i_n})$, and any couple of interpretations w, w' in $\mathcal{W}_\mathcal{P}$:

$$w \succeq^F_\Phi w' \textit{ iff } F(r_{i_1}(w), \ldots, r_{i_n}(w)) \geq F(r_{i_1}(w'), \ldots, r_{i_n}(w')) \tag{3}$$

where $r_i(x)$ is the rank of an interpretation x in the total preorder $\succeq_i$, for each agent i in N.

In the context of epistemic states given by total preorders over interpretations[4], we will say that an assignment $\Phi \mapsto \succeq_\Phi$ is *structure preserving* if $\succeq_{\succeq_i} = \succeq_i$ for any agent i in $\mathcal{S}$ and any i-profile $\succeq_i$. It is worth to note that such assignments are trivially basic assignments that satisfy Property **1** in Definition 2 [cf. Mata Díaz and Pino Pérez (2017)]. Thus, given an aggregation function F, it is possible to define an assignment $\Phi \mapsto \succeq^F_\Phi$ which is *structure preserving*, that is, the identity function for profiles of size one.

From the equivalence (3) and (B-Rep), it is possible to build an ES basic merging operator, ∇^F, which is associated to an aggregation function F, as follows:

$$\nabla^F(\Phi, \succeq) = \succeq^{\mathrm{lex}(\succeq, \succeq^F_\Phi)} \tag{4}$$

In particular, from the above equation it is possible to define two aggregation-based ES basic merging operators: ∇^Σ (*sum*) and $\nabla^{\max}$ (*max*), defined after the aggregation functions sum and max respectively.

Some variants of these operators can be defined. For instance we present three operators derived from sum: the Σ-*inverted ES basic merging operator*, the Σ-*quasi-inverted ES basic merging operator* and the *pseudo-sum ES basic merging operator*. These operators are presented for the first time in this work.

Σ-inverted ES basic merging operator:

$$\nabla^{\overleftarrow{\Sigma}}(\Phi, \succeq) = \succeq^{\mathrm{lex}(\succeq, \succeq^{\overleftarrow{\Sigma}}_\Phi)}$$

where $\succeq^{\overleftarrow{\Sigma}}_\Phi$ satisfies the following: for every pair w, w' in $\mathcal{W}_\mathcal{P}$; $w \succeq^{\overleftarrow{\Sigma}}_\Phi w'$ iff $w' \succeq^\Sigma_\Phi w$.

[4] That is $\mathcal{E} = \mathbb{P}(\mathcal{W}_\mathcal{P})$

Operator	Max. Cond.	(ESF5)	(ESF6)	(ESF7)	(ESF8)	(ESF8W)	(ESF-SD)	(ESF-U)	(ESF-P)	(ESF-I)	(ESF-D)	(ESF-NI)	(ESF-St)	(ESF-WD)	Str.-Proof
General Operators															
∇^{Σ}	✓	✓	✓	✓	✓	✓	✓	✓	✓	x	x	✓	x	x	x
$\nabla^{\max}$	✓	✓	✓	✓	x	✓	✓	✓	✓	x	x	✓	x	x	x
$\nabla^{\overleftarrow{\Sigma}}$	x	x	x	✓	✓	✓	✓	✓	✓	x	x	✓	x	x	x
$\nabla^{\overleftrightarrow{\Sigma}}$	✓	✓	x	x	x	x	✓	x	x	x	x	✓	x	x	x
$\nabla^{\mathrm{P}\Sigma}$	✓	✓	✓	✓	x	x	✓	✓	x	x	x	✓	x	x	x
∇^{π}	✓	✓	x‡	✓	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
$\nabla^{\mathrm{Q}\pi_{\geq}}$	✓	✓	x‡	x	x	✓	✓	x	✓	✓	✓	✓	✓	✓	✓
$\nabla^{\Sigma-\mathrm{P}\pi}$	✓	✓	✓	x	x	x	✓	✓	✓	x	✓	✓	x	✓	x
$\nabla^{\Sigma-\mathrm{W}\pi}$	✓	✓	✓	x	x	x	✓	✓	✓	x	x	✓	x	✓	x
$\nabla^{\overleftarrow{\pi}}$	x	✓	x	✓	x	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Complete Operators															
$\nabla^{\Sigma_{\geq}}$	x†	x	x‡	✓	✓	✓	✓	✓	✓	x	x	✓	x	x	x
$\nabla^{\Sigma-\mathrm{W}\pi_{\geq}}$	x†	x	x‡	x	x	x	✓	✓	✓	x	x	✓	x	✓	x
$\nabla^{\pi^{w*}_{\geq}}$	x	x	x	✓	✓	✓	x	✓	✓	✓	✓	x	✓	x	✓

Table 1: Landscape of satisfaction of the merging properties and strategy-proofness with respect precise-leximax liftings: ✓ means the property holds; x means the property doesn't hold; x† the property holds for single profiles with complete beliefs and x‡ means that the property holds for profiles where the beliefs of each agent are complete.

Because of the fact that the *sum* assignment is structure preserving, it is easy to see that the Σ-inverted assignment is not structure preserving.

The Σ-*pseudoinverted operator* has the same behavior of the Σ-*inverted operator* for profiles with more that one component.

Σ-pseudoinverted ES basic merging operator:

$$\nabla^{\overleftrightarrow{\Sigma}}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq^{\overleftarrow{\Sigma}}_{\Phi})}$$

where $\succeq^{\overleftrightarrow{\Sigma}}_{\Phi}=\succeq_i$ *if* Φ *is a singletone, namely* $\Phi=\succeq_i$*, and* $\succeq^{\overleftrightarrow{\Sigma}}_{\Phi}=\succeq^{\overleftarrow{\Sigma}}_{\Phi}$ *otherwise.*

It is easy to see that the Σ-*pseudoinverted assignment* is structure preserving.

The *pseudo-sum* operator, defined below, has the same behavior of the sum operator when the agents have a consensus, otherwise all the alternative are equally preferred.

Pseudo-sum ES basic merging operator:

$$\nabla^{\mathrm{P}\Sigma}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq^{\mathrm{P}\Sigma}_{\Phi})}$$

where $\succeq^{\mathrm{P}\Sigma}_{\Phi}=\succeq^{\Sigma}_{\Phi}$ *if* $\bigwedge\!\!\bigvee_{i\in N} B(\succeq_i)$ *is consistent, otherwise is* $\simeq$*, the flat order over* $\mathcal{W}_{\mathcal{P}}$.

Like the *sum* operator, the *pseudo-sum* operator is structure preserving. This is due to the fact that, for each single-profile $\succeq_i$, $B(\succeq_i)$ is consistent.

Another class of ES operators are called *projective-based ES merging operators*. Some of this kind of operators presented herein was introduced and studied by Mata Díaz and Pino Pérez (2017) (the *projective*, *quasiprojective* and the Σ-*pseudoprojective*). The remaining of them are new (the Σ-*weak-projective* operator and the *projective inverted operator*). This class of operators is given in terms of projections or variants of this. In order to present them, from now on, d_N denotes the maximal[5] of each finite society N of agents in $\mathcal{S}$ *i.e.* $d_N = \max(N)$, for each N in $\mathcal{F}^*(\mathcal{S})$.

[5]In order to define projective-based ES merging operators, we can also suppose that $d_N = \max(N)$, for each finite society of agents N in $\mathcal{F}^*(\mathcal{S})$.

The first projective-based operator to be presented is the *projective ES basic merging operator*, ∇^{π} (also called *projective operator* for short). This is totally determined by projections over profiles of epistemic states: the output of a merging process completely depends on the beliefs of the agent d_N and the integrity constraints.

Projective ES basic merging operator:

$$\nabla^{\pi}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq_{d_N})}$$

As we can see through its definition, the projective operator actually is an ES basic merging operator, being its basic assignment the *projective assignment*,$\Phi \mapsto \succeq_{d_N}$, which associates each epistemic profile Φ to its last component $\succeq_{d_N}$. Thus, the *projective assignment* is structure preserving.

Another class of projective-based operators to be presented is the *quasilinearized projective ES basic merging operators* (*quasilinearized projective operators* in short). In order to define them, let us consider a fixed linear order $\geq$ over $\mathcal{W}_{\mathcal{P}}$.

Quasilinearized projective ES basic merging operator:

$$\nabla^{\mathrm{Q}\pi_{\geq}}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq^{\mathrm{Q}\pi_{\geq}}_{\Phi})}$$

where $\succeq^{\mathrm{Q}\pi_{\geq}}_{\Phi}=\succeq_i$ *if* Φ *has a single input, namely* $\succeq_i$*, and* $\succeq^{\mathrm{Q}\pi_{\geq}}_{\Phi}=\succeq^{\mathrm{lex}(\succeq_{d_N},\geq)}$ *otherwise.*

Quasilinearized projective operators are ES basic merging operators too. Their basic assignments, $\Phi \mapsto \succeq^{\mathrm{Q}\pi_{\geq}}_{\Phi}$, associate each epistemic profile Φ to a linearizing of the maximal agent's epistemic state (preorder), excepting for single-profiles who are preserved. Thus, *Quasilinearized projective assignments* are structure preserving.

The following *projective-based ES merging operator* is defined using sum and projections. This operator is called Σ-*pseudoprojective ES merging operator* (Σ-*pseudoprojective operator* in short), $\nabla^{\Sigma-\mathrm{P}\pi}$.

Σ-pseudoprojective ES merging operator:

$$\nabla^{\Sigma-\mathrm{P}\pi}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq^{\Sigma-\mathrm{P}\pi}_{\Phi})}$$

where $\succeq_\Phi^{\Sigma-\mathrm{P}\pi}=\succeq^{\mathrm{lex}(\succeq_{d_N},\succeq_\Phi^{\Sigma})}$.

The Σ-*pseudoprojective operator* allows a very weak participation of all agents involved in the merging process. It is not hard to see that the Σ-*pseudoprojective assignment*, $\Phi \mapsto \succeq_\Phi^{\Sigma-\mathrm{P}\pi}$, is structure preserving.

The next operator, called Σ-weak-projective ES basic merging operator (Σ-*weak-projective operator* in short) is also new.

Σ-weak-projective ES basic merging operator:

$$\nabla^{\Sigma-\mathrm{W}\pi}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq_\Phi^{\Sigma-\mathrm{W}\pi})}$$

where $\succeq_\Phi^{\Sigma-\mathrm{W}\pi}=\succeq_{d_N}$, *if* $|\max(\succeq_{d_N})| = 1$, *and* $\succeq_\Phi^{\Sigma-\mathrm{W}\pi}=\succeq_\Phi^{\Sigma}$, *otherwise.*

The output of this operator is exactly that of the sum operator, except when the entrenched belief of the agent d_N is complete *i.e.* $|\max(\succeq_{d_N})| = 1$. In this case the output of the merging process depends only on the epistemic states of d_N and on the integrity constraints.

Similar to the *sum operator*, we can define some reverse variants of the projective operators. Some instances of such variants are those operators which we call *inverted projective ES basic merging operators* (*inverted projective operators* in short). Such operators are new to our knowledge.

Inverted projective ES basic merging operator:

$$\nabla^{\overleftarrow{\pi}}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\overleftarrow{\succeq}_{d_N})}$$

where $\overleftarrow{\succeq}_{d_N}$ *is defined as follows:* $w \overleftarrow{\succeq}_{d_N} w'$ *iff* $w' \succeq_{d_N} w$.

Like the Σ-*inverted* assignment, the inverted projective assignment, $\Phi \mapsto \overleftarrow{\succeq}_{d_N}$, is not structure preserving. This is because the reversing of the total preorders (on single-profiles) over $\mathcal{W}_\mathcal{P}$.

Now we present some families of complete merging operators built through some of merging operators defined above. Except for the *linearized projective operators*, all of these ES basic merging operators are new. In order to define them, we will considerer a fixed linear order $\geq$ over $\mathcal{W}_\mathcal{P}$. Let us note the following:

Observation 2 If $\Phi \mapsto \succeq_\Phi$ is an assignment, we have that $\succeq_\Phi^{\geq}=\succeq^{\mathrm{lex}(\succeq_\Phi,\geq)}$ is a linear order over $\mathcal{W}_\mathcal{P}$, for every epistemic profile Φ in $\mathcal{P}(\mathcal{S},\mathcal{E})$. Thus, the new assignment $\Phi \mapsto \succeq_\Phi^{\geq}$ actually is a linear assignment (that is not structure preserving). Moreover, from Theorem 1 we have that the following is indeed a complete ES basic merging operator:

$$\nabla^{\geq}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq_\Phi^{\geq})} \tag{5}$$

The first family of complete operators that we present is a linearizing of the sum operator. These operators are called *linearized sum ES basic merging operators* (*linearized sum operators* in short).

Linearized sum ES basic merging operator:

$$\nabla^{\Sigma\geq}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq_\Phi^{\Sigma\geq})}$$

where $\succeq_\Phi^{\Sigma\geq}=\succeq^{\mathrm{lex}(\succeq_\Phi^{\Sigma},\geq)}$.

The operators in the following family of complete operators are called *linearized projective ES basic merging operators* (*linearized projective operators* in short). Each operator in this class is the linearized of a projective operator.

Linearized projective ES basic merging operator:

$$\nabla^{\pi\geq}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq_{d_N}^{\geq})}$$

where $\succeq_{d_N}^{\geq}=\succeq^{\mathrm{lex}(\succeq_{d_N},\geq)}$.

Other linearized variants of the sum and projective operators are those obtained by forcing a fixed interpretation in the top of the output from their linear assignment. These operators are called *impose-linearized operators*, and are introduced for the first time in this work. In order to define them let us consider a fixed interpretation w^* in $\mathcal{W}_\mathcal{P}$.

The first *impose-linearized operator* to be presented is the *impose-linearized sum ES basic merging operator* (*impose-linearized sum operator* in short).

Impose-linearized sum ES basic merging operator:

$$\nabla^{\Sigma^{w^*}\geq}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq_\Phi^{\Sigma^{w^*}\geq})}$$

where $\succeq_\Phi^{\Sigma^{w^*}\geq}$ *is defined as follows:* $\max(\succeq_\Phi^{\Sigma^{w^*}\geq}) = \{w^*\}$ *and* $\succeq_\Phi^{\Sigma^{w^*}\geq}{\restriction}_{\mathcal{W}_\mathcal{P}\setminus\{w^*\}}=\succeq_\Phi^{\Sigma\geq}{\restriction}_{\mathcal{W}_\mathcal{P}\setminus\{w^*\}}$.

Another *impose-linearized operator* that we present is the *impose-linearized projective ES basic merging operator* (also called *impose-linearized projective operator* in short).

Impose-linearized projective ES basic merging operator:

$$\nabla^{\pi^{w^*}\geq}(\Phi,\succeq) = \succeq^{\mathrm{lex}(\succeq,\succeq_{d_N}^{w^*\geq})}$$

where $\succeq_{d_N}^{w^*\geq}$ *is defined as follows:* $\max(\succeq_{d_N}^{w^*\geq}) = \{w^*\}$ *and* $\succeq_{d_N}^{w^*\geq}{\restriction}_{\mathcal{W}_\mathcal{P}\setminus\{w^*\}}=\succeq_{d_N}^{\geq}{\restriction}_{\mathcal{W}_\mathcal{P}\setminus\{w^*\}}$.

Since these operators are strongly defined through their linear assignments, namely, *impose-linearized sum assignments*, $\Phi \mapsto \succeq_\Phi^{\Sigma^{w^*}\geq}$, and *impose-linearized projective assignments*, $\Phi \mapsto \succeq_{d_N}^{w^*\geq}$, these operators are actually complete ES basic merging operators.

We leave the reader the task of extracting some interrelations between the properties. It is interesting to observe that we can see in Table 1 that the converse of Theorem 4 doesn't hold. Actually, the Σ-*pseudoprojective operator* is a weak dictatorial and non-imposed operator. However, it is manipulable with respect to the precise-leximax lifting.

Concluding remarks and perspectives

The framework presented here is a generalization of the propositional logic merging framework. The only postulate that we do not consider from that framework is IC4.

We have established general results of manipulability for ES merging operators. One very interesting feature of our approach is that it gives interesting instantiations of the Manipulability Theorem (Theorem 4) for different representations of epistemic states. Thus, this applies to Ordinal Conditional Functions, rational relations, and of course total preorders. However, with the representation of epistemic states as formulas, our results do not hold because it is impossible to have Standard Domain in presence of a good representation of beliefs, namely the Maximality Condition [cf. Mata Díaz and Pino Pérez (2017)]. This fact highlights the necessity of using complex epistemic states if we want to have properties like **(ESF-SD)**.

Theorem 4 can be seen as a weak version of Barberà-Kelly's Theorem [cf. Leal and Pino Pérez (2017)] in the framework of the epistemic merging. It would be interesting to investigate if a complete version of Barberà-Kelly's Theorem holds in this epistemic framework.

An interesting open question is to give a complete characterization of the operators which are strategy-proof. A future work is to investigate, in the framework of logic based merging, other notions of manipulability coming from Social Choice Theory.

Acknowledgments

Thanks to the CDCHTA-ULA for its financial support through the Project N° C-1855-13-05-A.

References

Arrow, K. 1963. *Social choice and individual values*. Yale University Press.

Barberà, S.; Bossert, W.; and Pattanaik, P. K. 2004. *Ranking sets of objects*, volume 2 of *Handbook of Utility Theory*. Kluwer Publisher. chapter 17, 893–978.

Barberà, S. 1977. Manipulation of social decision functions. *J. Econom. Theory* 15(2):266–278.

Benferhat, S.; Konieczny, S.; Papini, O.; and Pino Pérez, R. 2000. Iterated revision by epistemic states: Axioms, semantics and syntax. In Horn, W., ed., *ECAI*, 13–17. IOS Press.

Brandt, F., and Brill, M. 2011. Necessary and sufficient conditions for the strategyproofness of irresolute social choice functions. In *Proceedings of the 13th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2011), Groningen, The Netherlands, July 12-14, 2011*, 136–142.

Brandt, F. 2011. Group-strategyproof irresolute social choice functions. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 79–84.

Campbell, D. E., and Kelly, J. S. 2002. Impossibility theorems in the arrovian framekork. In Arrow, K. J.; Sen, A. K.; and Suzumura, K., eds., *Handbook of Social Choice and Welfare, Volume 1 (Handbooks in Economics)*. North-Holland. 35–94.

Chopra, S.; Ghose, A. K.; and Meyer, T. A. 2006. Social choice theory, belief merging, and strategy-proofness. *Information Fusion* 7(1):61–79.

Darwiche, A., and Pearl, J. 1997. On the logic of iterated belief revision. *Artificial intelligence* 89:1–29.

Dietrich, F., and List, C. 2007. Strategy-proof judgment aggregation. *Economics and Philosophy* 23:269–300.

Dubois, D.; Prade, H.; and Lang, J. 1994. Possibilistic logic. In D.M. Gabbay et al., ed., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3. Oxford, UK: Oxford University Press. 439–513.

Everaere, P.; Konieczny, S.; and Marquis, P. 2007. The strategy-proofness landscape of merging. *Journal of Artificial Intelligence Research (JAIR)* 28:49–105.

Fishburn, P. C. 1972. Even-chance lotteries in social choice theory. *Theory and Decision* 3(1):18–40.

Gärdenfors, P. 1976. Manipulation of social choice functions. *Journal of Economic Theory* 13(2):217 – 228.

Gibbard, A. 1973. Manipulation of voting schemes: A general result. *Econometrica* 41(4):587–601.

Grossi, D., and Pigozzi, G. 2014. *Judgement Agrgregation: A Primer*. Morgan & Claypool Publishers.

Kelly, J. S. 1977. Strategy-proofness and social choice functions without singlevaluedness. *Econometrica* 45(2):439–446.

Kelly, J. S. 1978. *Arrow impossibility theorems.* New York Academic Press.

Konieczny, S., and Pino Pérez, R. 1999. Merging with integrity constraints. In Hunter, A., and Parsons, S., eds., *ESCQARU*, volume 1638 of *Lecture Notes in Computer Science*, 233–244. Springer.

Konieczny, S., and Pino Pérez, R. 2002. Merging information under constraints: A logical framework. *J. Log. Comput.* 12(5):773–808.

Konieczny, S., and Pino Pérez, R. 2005. Propositional belief base merging or how to merge beliefs/goals coming from several sources and some links with social choice theory. *European Journal of Operational Research* 160(3):785–802.

Konieczny, S., and Pino Pérez, R. 2011. Logic based merging. *J. Philos. Logic* 40(2):239–270.

Konieczny, S., and Pino Pérez, R. 2013. Confluence operators and their relationships with revision, update and merging. *Annals of Mathematics and Artificial Intelligence* 69(1):73–101.

Konieczny, S.; Lang, J.; and Marquis, P. 2004. DA^2 merging operators. *Artificial Intelligence* 157(1-2):49–79. Nonmonotonic Reasoning.

Leal, J., and Pino Pérez, R. 2007. A notion of manipulability based on lifting preferences. *Notas de Matemáticas* 3(1):73–94.

Leal, J., and Pino Pérez, R. 2017. A weak version of Barberà-Kelly's theorem. *Revista Colombiana de Matemáticas* 51(2):173–194. In press.

Mata Díaz, A., and Pino Pérez, R. 2011. Logic-based fusion of complex epistemic states. In Liu, W., ed., *ECSQARU*, volume 6717 of *Lecture Notes in Computer Science*, 398–409. Springer.

Mata Díaz, A., and Pino Pérez, R. 2017. Impossibility in belief merging. *Artificial Intelligence* 251:1 – 34.

Satterthwaite, M. A. 1975. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory* 10(2):187–217.

Suzumura, K. 2002. Introduction. In Arrow, K.; Sen, A. K.; and Suzumura, K., eds., *Handbook of Social Choice and Welfare, Volume 1 (Handbooks in Economics)*. North-Holland. 1–32.

Taylor, A. D. 2005. *Social Choice and the Mathematics of Manipulation (Outlooks).* Cambridge University Press.

A simple qualitative framework for resource allocation*

Franklin Camacho[1] and **Gerardo Chacón**[2] and **Ramón Pino Peréz**[1,3]

[1]School of Mathematical Sciences and Information Techonology, Yachay Tech University, Urcuquí, Ecuador
[2]Department of Science, Technology and Mathematics, Gallaudet University, Washington DC, USA.
[3]Departamento de Matemáticas, Facultad de Ciencias, Universidad de Los Andes - Mérida, Venezuela.

fcamacho@yachaytech.edu.ec gerardo.chacon@gallaudet.edu pino@ula.ve

Abstract

We use an algebraic viewpoint, namely a matrix framework to deal with the problem of resource allocation under uncertainty in the context of a qualitative approach. Our basic qualitative data are a plausibility relation over the resources, a hierarchical relation over the agents and, of course, the preference that the agents have over the resources. With this data we propose a qualitative binary relation $\trianglerighteq$ between allocations such that $\mathcal{F} \trianglerighteq \mathcal{G}$ has the following intended meaning: the allocation $\mathcal{F}$ produces more or equal social welfare than the allocation $\mathcal{G}$. We prove that there is a family of allocations which are maximal with respect to $\trianglerighteq$. We prove also that there is a notion of simple deal such that optimal allocations can be reached by sequences of simple deals. Finally, we introduce some mechanism for discriminating optimal allocations.

Keywords: Resource allocation - qualitative optimal allocations - qualitative social welfare.

Introduction

The task of distribution of a divisible or indivisible set of goods among a set of agents is an important issue at the crossroads of many areas of knowledge. This has been studied mainly in Economics. The mathematical models of fair distributions, in particular when the goods are divisible, have been studied by some mathematicians and economists such as Banach and Steinhaus (Steinhaus 1948; 1949) before the middle of the Twentieth century and by Brams, Taylor, Zwicker and others towards the end of the past century (Brams, Taylor and Zwicker 1994; Brams and Taylor 1995; Brams 1995; Brams, Taylor and Zwicker 1997) (for a good survey see (Brams and Taylor 1996)). Some axiomatic analysis about resource allocation problems have been done by Moulin, Thomson and Beviá among others(Moulin and Thomson 1997; Thomson 2011; Beviá 1996). In more recent years the mathematical models of satisfactory distributions have occupied different communities of researchers besides economists. Among them, computer scientists working in the area of Artificial Intelligence and Multi-agent Systems, see for instance the works of Sandholm (Sandholm 1998; 1999), Endriss et al. (Endriss et al. 2006) and Chevaleyre et al. (Chevaleyre et al. 2007; Chevaleyre, Endriss, and Maudet 2017).

The problem concerning this paper is the distribution of a finite set of indivisible goods (resources) into a finite set of individuals (agents). This is known as the problem of resource allocation. There are many concerns about this problem. The first one is how to consider a determined allocation satisfactory. To deal with this aspect some concepts of social welfare are used, most of them coming from aggregation of numerical functions of individual utility (Moulin 1988; Endriss et al. 2006). Thus, the notion of optimal allocation arises naturally as the one producing the maximal social welfare.

Another important aspect studied is the rationality of negotiation between agents in order to reach better resource allocations. Again here the notion of *being better* is closely connected to the improvement of the social welfare (Sandholm 1998; Endriss et al. 2006). From the algorithmic point of view it is important that these negotiations be simple and that any sequence of these simple negotiations negotiation processes end in an optimal allocation.

In this work we continue an investigation of these problems in which we don't use numbers. That is, we propose a qualitative notion of preference between the allocations that is based only on the knowledge of the set of resources that agents desire, a plausibility relation between set of resources which encodes the uncertainty, and a hierarchical relation between agents. Let us give an example of simple ideas we want to capture.

Example 1 *Suppose there are three research teams in a new Artificial Intelligence Center of an University Computer Science Department. The teams are working in Knowledge Representation, Robotics and Natural Language Processing, respectively. There are two candidates to realize PhD studies in this AI Center. The first one, John -a brilliant one-, has interesting job offers in Google and in Face Book. Thus, it is less plausible that he remains at the University for the PhD studies. The other candidate, Peter, doesn't like the work environment of private companies. He prefers, for sure, the Universities scholar environment. In any case both students are goods in the three topics developed in the center.*

The Knowledge Representation Team would have Peter as PhD student; the Robotics and the Natural Processing Language Teams would like both students. At the moment of the

*The third author is supported by the Project CDCHTA-ULA C-1855-13-05-A.

decision, the team working in Natural Language Processing has 15 PhD students; the team working in Robotics has 6 PhD students and the team working in Knowledge representation has 1 student. In a Council of the Center, in order to create a balanced development of the teams and the research topics, some priorities have been established. Thus, the Knowledge Representation Team is the one having the first priority for receiving PhD students. After that, the Robotics Team has the second priority and in the third place the priority is given to the Natural Processing Language Team. All the teams agree about that.

This situation is a typical and simple problem of resource allocation that we would like to model. There are two resources r_1 *and* r_2 *(Peter and John, respectively) and three agents* 1, 2 *and* 3 *(the Knowledge Representation Team, Robotics Team and Natural Language Processing Team, respectively). There is priority relation* $\succ_h$ *between the agents:* $1 \succ_h 2 \succ_h 3$ *and a plausibility relation between resources (coding uncertainty):* $r_1 \succ_{\mathcal{R}} r_2$*. With the pieces of information we have, a very "natural" resource allocation is to give the resource* r_1 *to agent* 1 *and the resource* r_2 *to the agent* 2*, that is the Knowledge Representation Team (the least favored team so far) receives Peter (the student having more chance to come) and the Robotics Team receives John (the student having less chance to come). In this allocation the Natural Language Team (the most developed team) will not receive PhD students.*

This work will explain in a precise way why a resource allocation as one in previous example is good. Actually, we will propose a notion of qualitative welfare (a relation) for which the resource allocation of Example 1 is optimal.

In the work of Pino Pérez and colleagues (Pino Peréz, Varela Montilva and Camacho 2016) three strong hypothesis were made. First, it was assumed that there was a linear order over the resources. Second, that the plausibility relation between the resources was the possibilistic relation associated to this linear order. Third, it was supposed that there was a linear order between agents encoding a hierarchy between them. In this work we generalize these three assumptions. We suppose now that the relation between the resources and the relation between the agents are total preorders a more general natural structure for encoding preferences and eve uncertainty in a qualitative way. We suppose also that the plausibility relation between the sets of resources is a lifting (an extension, see (Barberà, Bossert, and Pattanaik 2004)) of the relation between the resources with very few properties.

With these tools we use the *dominance plausible rule* -a sort of qualitative expected utility introduced by Dubois et al. (Dubois et al. 2002; Dubois, Fargier, and Perny 2003)- in order to compare two allocations. We show with easy matricial representations that there is a family of optimal allocations. We propose a notion of *simple rational deal* and we prove that we can reach optimal allocations by sequences of simple rational deals.

This work is structured as follows. Section *Preliminaries* contains the basic notions. Section *Classifying allocations* contains the notion of preference between allocations based on a plausibility relation between sets of resources which in turn is based on a qualitative notion of social welfare. Section **Simple and rational deals** is devoted to the notion of simple resource interchange guided by the relationships between the agents. Section *Good allocations* is devoted to define a class of resource allocations considered as well behaved and to see that they can be reached by sequences of simple rational deals. In Section *Optimal resource allocations* we see that good resource allocations are, indeed, optimal with respect to the notion of qualitative welfare. Section *Discriminating good allocations* is devoted to give a method to discriminate optimal good allocations. We finish with a section in which we make some concluding remarks and present some lines of future work.

Preliminaries

In this section, we state the preliminary concepts and notations that we will use throughout the article. Fix a nonempty set X and a binary relation $\succeq$ defined on X. *The strict relation* $\succ$ associated to $\succeq$ is defined as

$$a \succ b \Leftrightarrow [(a \succeq b) \wedge \neg(b \succeq a)], \tag{1}$$

the *indifferent relation* $\preceq$ is defined as

$$a \preceq b \Leftrightarrow [(a \succeq b) \wedge (b \succeq a)]. \tag{2}$$

If $A \subseteq X$, we denote the maximal[1] elements of A as $\max(A)$ and analogously we denote the minimal elements of A as $\min(A)$.

If X is finite, $X = \{x_1, \cdots, x_n\}$, the relation $\succeq$ defines a *relation matrix* as:

$$X_{ij} = \begin{cases} 1, & \text{if } x_i \succeq x_j; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Note that the previous matrix depends upon the order of the elements of X. But any reordering of the elements of X has an associated relation matrix with respect to $\succeq$ closely related to the first matrix. More precisely, if $(X_{ij})_{n\times n}$ and $(Y_{ij})_{n\times n}$ are matrices associated to $\succeq$ with two different orders of X, then there exists a permutation P, of the identity matrix I_n of orden n, such that

$$(Y_{ij})_{n\times n} = P(X_{ij})_{n\times n}$$

A relation is a total pre-order on X if it is total and transitive. A relation is a linear order if it is irreflexive and transitive.

We start by the defining the two main sets to be studied. These are the set of agents and the set of resources. We will denote the set of agents by $\mathcal{A}$ which is defined as a finite, nonempty set with $|\mathcal{A}| = q$, i.e, $\mathcal{A} = \{1, 2, \cdots, q\}$. We will assume there exists a total pre-order $\succeq_h$ defined on $\mathcal{A}$. Given two agents $i, j \in \mathcal{A}$, the symbol $i \succeq_h j$ will be interpreted as: *agent* i *has a higher or equal hierarchy than agent* j. The strict relation $\succ_h$ associated to $\succeq_h$ defined as (1) can be thought as: *agent* i *has greater hierarchy than agent* j. The

[1]The maximal elements of A with respect the relation $\succeq$ are defined as the set $\{a \in A : \nexists b \text{ such that } b \succ a\}$. Analogously, the minimal elements of A with respect the relation $\succeq$ are defined as the set $\{a \in A : \nexists b \text{ such that } a \succ b\}$.

indifferent relation $\sim_h$ defined as in (2) can be interpreted as: *agent i has the same hierarchy as agent j*. Sometimes we will abuse the notation and denote by $i \in \mathcal{A}$ an agent and at the same time a natural number between 1 and q. A relation matrix associated to $\succeq_h$, defined as in equation (3) will be called *a hierarchy matrix* and denoted by $\mathcal{A}_h = (a_{ij})_{q\times q}$.

The set of *resources* will be denoted as $\mathcal{R} = \{r_1, \cdots, r_k\}$. We will assume that a total pre-order $\succeq_{\mathcal{R}}$, is defined on $\mathcal{R}$. The strict and indifferent relations $\succ_{\mathcal{R}}$ and $\simeq_{\mathcal{R}}$, are defined in similar way as in equations (1) and (2), respectively. Based on the relation $\succeq_{\mathcal{R}}$ we will later define a binary relation $\sqsupseteq$ on the set of subsets of $\mathcal{R}$, denoted as $\mathcal{P}(\mathcal{R})$. Such relation should be a *lifting* of $\succeq_{\mathcal{R}}$, in the sense that it should preserve the order $\succeq_{\mathcal{R}}$ when restricted to the the singletons. Such relation will be understood as a 'plausibility' relation on $\mathcal{P}(\mathcal{R})$.

Once defined the sets $\mathcal{A}$ and $\mathcal{R}$, we will assume that each agent is interested in some resources. This relation can be recorded in the following *matrix of request* $\mathcal{P} = (P_{in})_{q\times k}$ that contains the information about the resources each agent requests.

$$P_{in} = \begin{cases} 1, & \text{if agent } i \text{ requests the resource } r_n; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Remark 1 *Sometimes, we will refer to a resource $r \in \mathcal{R}$ without specifying the subindex. In such cases, P_{ir} denotes the entry of the matrix $\mathcal{P}$ in the i-th row and the column associated to the resource r.*

We make the assumptions that the agents either request or do not request a resource, that this is known beforehand, and that it is invariant. Also, we assume that an agent does not have different levels of preference over the resources. This is a subject that could be addressed in future research. Under this conditions, we notice that the $i-$th row of $\mathcal{P}$ contains complete information about the resources that agent i requests, whereas the $n-$th column determines all the agents that request the resource r_n.

We are now ready to define the main concept of the article. A *resource allocation* is a distribution of the resources among the agents where each resource is granted to one and only one agent. We introduce a matrix notation that will be used throughout the rest of the article.

Definition 1 *Let $\mathcal{A}$ and $\mathcal{R}$ be nonempty sets, with $k = |\mathcal{R}|$ and $q = |\mathcal{A}|$. A resource allocation is a matrix*

$$\mathcal{F} = (f_{in})_{q\times k}$$

such that for each $1 \leq n \leq k$, there exists a unique $1 \leq i \leq q$ for which $f_{in} = 1$ and $f_{jn} = 0$ for $j \neq i$.

Notice that if $f_{in} = 1$ then it is understood that agent i is the only one who was granted the resource r_n.

Similarly to the notation used with the matrix of requests, we will write f_{ir} to denote the entry of the matrix $\mathcal{F}$ corresponding to the i-th row and the column associated to the resource r.

Sometimes we will use a set of resource allocations $\mathcal{F}_1, \ldots, \mathcal{F}_m$. In such case, we will use the notation $\mathcal{F}_l = (f^l_{in})_{q\times k}$ when necessary.

Given a fixed resource $r \in \mathcal{R}$, we define a "priority relation" on $\mathcal{A}$ associated to r in the following way: *agent i has a higher priority than agent j* if and only if *agent i has equal or grater hierarchy than agent j and agent i requests the resource r, or agent i does not have equal or greater hierarchy than agent j and agent j does not request the resource r.*

Definition 2 *Given $r \in \mathcal{R}$,* the priority matrix associated to *r, is the matrix $(a^r_{ij})_{q\times q}$, where*

$$a^r_{ij} = a_{ij}P_{ir} + (1 - a_{ij})(1 - P_{jr}) \tag{5}$$

Notice that each term of equation (5) is either one or zero. If $a_{ij}P_{ir} = 1$ then agent i has a higher hierarchy than agent j and agent i requests resource r. On the other hand, if $(1 - a_{ij})(1 - P_{jr}) = 1$, then $a_{ij} = 0$ and $P_{jr} = 0$. This implies agent i doest not have a higher hierarchy than agent j and agent j does not request resource r. This definition captures that on (Pino Peréz, Varela Montilva and Camacho 2016) where the hierarchy order were assumed to be a linear order.

Classifying allocations

As a first step in classifying resource allocations, we want to define an equivalence relation among allocations. Then we will define the set where one resource allocation dominates another.

Definition 3 *Let $r \in \mathcal{R}$ and let c and c^* be two canonical vectors of size q. Suppose that $i, j \in \mathcal{A}$ are such that the i-th entry of c is equal to 1 and the j-th entry of c^* is equal to 1. We will say that c and c^* are r-equivalent, denoted as $c \equiv_r c^*$ if agents i and j have the same hierarchy and the same preference over the resource r. In other words,*

$$c \equiv_r c^* \Leftrightarrow (a_{ij} = a_{ji} = 1) \wedge (P_{ir} = P_{jr}) \tag{6}$$

Let $\mathcal{F}$ and $\mathcal{G}$ be two resource allocations and define the set of all non-r-equivalent columns as

$$D_{\mathcal{FG}} = \{r \in \mathcal{R} : c^{\mathcal{F}}_r \not\equiv_r c^{\mathcal{G}}_r\} \tag{7}$$

where $c^{\mathcal{F}}_r$ and $c^{\mathcal{G}}_r$ denote the columns of $\mathcal{F}$ and $\mathcal{G}$, respectively, associated to the resource r. Notice that $D_{\mathcal{FG}} = D_{\mathcal{GF}}$.

We will define a set where the resource allocation $\mathcal{F}$ *dominates* the resource allocation $\mathcal{G}$. Roughly, this will be the set of resources that are assigned under $\mathcal{F}$ to an agent with higher priority than if assigned under $\mathcal{G}$.

Definition 4 *Let $\mathcal{F} = (f_{in})$ and $\mathcal{G} = (g_{in})$ be two resource allocations. The set $[\mathcal{F} \succ \mathcal{G}]$ where $\mathcal{F}$ dominates $\mathcal{G}$, is defined as:*

$$[\mathcal{F} \succ \mathcal{G}] = \{r \in D_{\mathcal{FG}} : a^r_{ij} = 1, \text{ where } f_{ir} = 1 \text{ and } g_{jr} = 1\} \tag{8}$$

In other words, $r \in [\mathcal{F} \succ \mathcal{G}]$ if and only if for the unique agents i and j such that $f_{ir} = 1$ and $g_{jr} = 1$ it holds that $a^r_{ij} = 1$ and the columns $(f_{ir})_{1\leq i\leq q}$ and $(g_{ir})_{1\leq i\leq q}$ are not r-equivalent.

Once we have defined dominant sets as subsets of $\mathcal{R}$, we will define a way of comparing resource allocations. The idea is to capture a notion of social welfare (Moulin 1988;

2003; Endriss et al. 2006) used to compare allocations via a notion of qualitative welfare. The notion we use next is inspired of the *dominance plausible rule* proposed by Dubois el al. (Dubois et al. 2002; Dubois, Fargier, and Perny 2003).

We are going to consider a relation $\sqsupseteq$ on the set $\mathcal{P}(\mathcal{R})$ which is a *lifting* of $\succeq_{\mathcal{R}}$, that is for any resources r, r' we have

$$\{r\} \sqsupseteq \{r'\} \quad \text{iff} \quad r \succeq_{\mathcal{R}} r'$$

The idea is that the relation $\sqsupseteq$ encodes the plausibility relation overs subsets of resources given by the plausibility relation $\succeq_{\mathcal{R}}$ between resources in the same way that we can construct a finite probability measure over events if we know the probability on states.

Definition 5 *Let $\sqsupseteq$ a plausibility relation on the set $\mathcal{P}(\mathcal{R})$. We say that the resource allocation $\mathcal{F}$ produces more social welfare than a resource allocation $\mathcal{G}$, denoted as $\mathcal{F} \trianglerighteq \mathcal{G}$, if*

$$[\mathcal{F} \succ \mathcal{G}] \sqsupseteq [\mathcal{G} \succ \mathcal{F}]. \tag{9}$$

Moreover, we will say that $\mathcal{F} \triangleright \mathcal{G}$ if $\mathcal{F} \trianglerighteq \mathcal{G}$ and $\neg(\mathcal{G} \trianglerighteq \mathcal{F})$.

Notice that the previous definition depends on a binary relation on $\mathcal{P}(\mathcal{R})$ that, for now, could be an arbitrary lifting of $\succeq_{\mathcal{R}}$. A similar approach was studied in (Pino Peréz, Varela Montilva and Camacho 2016) where a strict possibilistic relation was considered for the case in which $\succeq_{\mathcal{R}}$ was a linear order. Indeed our relation $\sqsupseteq$ will be an extension (see (Barberà, Bossert, and Pattanaik 2004) for a study of the ways to do extensions and their properties) of the relation $\succeq_{\mathcal{R}}$ having very few properties.

The relation $\triangleright$ is capturing our notion of qualitative welfare under uncertainty, the uncertainty of truly disposing of all resources. The fact $\mathcal{F} \triangleright \mathcal{G}$ has to be understood as follows: it is more plausible that the allocation $\mathcal{F}$ produces more welfare than the allocation $\mathcal{G}$.

We are ready to define the notion of optimality among resource allocations. Notice that such definition will depend on the binary relation $\sqsupseteq$.

Definition 6 *Given a binary relation $\sqsupseteq$ on the set $\mathcal{P}(\mathcal{R})$, which is a lifting of $\succeq_{\mathcal{R}}$. We will say that a resource allocation $\mathcal{F}^*$ is optimal if $\mathcal{F}^* \trianglerighteq \mathcal{G}$ for every resource allocation $\mathcal{G}$.*

Here it is relevant to say that optimal resource allocations need not to be unique since the previous definition does not distinguish among equivalent allocations.

Simple and rational deals

In order to search for an optimal resource allocation, we will first introduce negotiation concepts. By a deal we will understand a pair of allocations that are equal except for only one column. We will define the concept in a more rigorous way by making use of permutation matrices.

Denote as I_q the identity matrix of size $q \times q$. A permutation matrix E_{ij} is the result of interchanging the rows i and j of I_q. Clearly, $E_{ij} = E_{ji}$.

Definition 7 *Given two resource allocations $\mathcal{F}$ and $\mathcal{G}$ such that $\mathcal{F} \neq \mathcal{G}$, we say that the pair $(\mathcal{F}, \mathcal{G})$ is a simple deal if there exists a unique $r \in \mathcal{R}$ and a permutation matrix $E_{ij} \neq I_q$, such that if $\mathcal{F} = [c_1, \cdots, c_{r-1}, c_r, c_{r+1} \cdots c_k]$, where $c_1, \ldots, c_k$ denote the columns of F, then*

$$\mathcal{G} = [c_1, \cdots, c_{r-1}, E_{ij}c_r, c_{r+1} \cdots c_k].$$

Lemma 1 *If $(\mathcal{F}, \mathcal{G})$ is a simple deal, then the permutation matrix E_{ij} is unique.*

Proof: Suppose that $\mathcal{F} = (f_{in})_{q\times k}$, and $\mathcal{G} = (g_{in})_{q\times k}$ are two resource allocations such that $(\mathcal{F}, \mathcal{G})$ is a simple deal, then there exists a unique resource r such that $\mathcal{F}$ and $\mathcal{G}$ differ only in the column associated to resource r. Let $i \in \mathcal{A}$ and $j \in \mathcal{A}$ be such that $f_{ir} = 1$ and $g_{jr} = 1$, then $f_{i^*r} = 0$ for all $i^* \neq i$ and $g_{j^*r} = 0$ for all $j^* \neq j$.

Now if $E_{i^*j^*}$, is another permutation matrix, with $i^* \neq i$ or $j^* \neq j$, then either $f_{i^*r} = 0$ or $g_{j^*r} = 0$ and consequently $E_{i^*j^*}c_r \neq E_{ij}c_r$, where c_r is the r-th column of $\mathcal{F}$. ∎

Using the notation of the previous lemma, we will say that the permutation matrix E_{ij} *represents* the simple deal $(\mathcal{F}, \mathcal{G})$ associated to the resource $r \in \mathcal{R}$.

Definition 8 *Given two resource allocations $\mathcal{F}$ and $\mathcal{G}$, we will say that $(\mathcal{F}, \mathcal{G})$ is a rational deal if it is a simple deal represented by the permutation matrix E_{ij}, and if one of the following conditions holds.*

$$(a_{ji} = 1) \wedge (P_{jr} = 1) \tag{10}$$

or

$$(a_{ij} = 1) \wedge (\forall t, \, (a_{ti} = 1 \Rightarrow P_{tr} = 0)) \tag{11}$$

Condition (10) means that agent j has a higher or equal hierarchy than i and moreover, requests resource r. Condition (11) says that agent i has a higher or equal hierarchy than j, that i does not request the resource r and no other agent t with higher or equal hierarchy as i requests the resource r.

We have to note that in our setting there is not compensation when a deal is realised. Thus our notion of rational deal can be seen more as an altruistic notion, a *cession*: an agent i cedes a resource r to an agent j because the agent j *needs more* this resource[2]. As a matter of fact, the hierarchical relation between the agents, $\succeq_h$, tries to code this necessity. Thus, for instance, in Example 1, the relation $\succeq_h$ encodes this necessity. For instance, $1 \succ_h 2$ means that the Knowledge Representation Team needs more a PhD student than the Robotics Team.

Definition 9 *A sequence of resource allocations $\mathcal{F}_1$,$\mathcal{F}_2, \ldots, \mathcal{F}_m$ is called* sequence of rational deals *if for every $1 \leq i \leq m-1$, the pair $(\mathcal{F}_i, \mathcal{F}_{i+1})$ is a rational deal.*

The following lemma and the corollary that follows will be needed in Section *Optimal resources allocation*.

Lemma 2 *Suppose that the permutation matrices E_{ij} y $E_{i^*j^*}$ represent, respectively, the rational deals $(\mathcal{F}_1, \mathcal{F}_2)$*

[2]Of course, from an abstract point of view, the relation $\succeq_h$ could encode a power relation and not a necessity relation. In such a case the notion of welfare between allocations is not very fair: a "rational deal" becomes an expropiation act and the model, instead of altruistic becomes a rapacity model. Therefore, our model is interesting when the hierarchical relation beteens agents is altruistic.

and $(\mathcal{F}_2, \mathcal{F}_3)$ associated to the same resource $r \in \mathcal{R}$. Then the product $E_{ij^*} = E_{i^*j^*}E_{ij}$ also represents a rational deal associated to r.

Proof: We will show that the matrix $E = E_{i^*j^*}E_{ij}$ represents the rational deal $(\mathcal{F}_1, \mathcal{F}_3)$ associated to r. First, notice that $i^* = j$ and $E = E_{ij^*}$.

Suppose that $\mathcal{F}_1 = [c_1, \ldots, c_r, \ldots, c_k]$, then since $(\mathcal{F}_1, \mathcal{F}_2)$ and $(\mathcal{F}_2, \mathcal{F}_3)$ are simple deals, we can write $\mathcal{F}_2 = [c_1, \ldots, E_{ij}c_r, \ldots, c_k]$ and $\mathcal{F}_3 = [c_1, \ldots, E_{i^*j^*}E_{ij}c_r, \ldots, c_k] = [c_1, \ldots, E_{ij^*}c_r, \ldots, c_k]$. In consequence, $(\mathcal{F}_1, \mathcal{F}_3)$ is a simple deal.

It remains to show that one of the following conditions hold:

$$(a_{j^*i} = 1) \wedge (P_{j^*r} = 1) \tag{12}$$

or

$$(a_{ij^*} = 1) \wedge (\forall t,\, (a_{ti} = 1 \Rightarrow P_{tr} = 0)). \tag{13}$$

But since $(\mathcal{F}_1, \mathcal{F}_2)$ and $(\mathcal{F}_2, \mathcal{F}_3)$ are rational deals, we have

1. E_{ij} represents $(\mathcal{F}_1, \mathcal{F}_2)$ and one of the following conditions hold:
   (a) $(a_{ji} = 1) \wedge (P_{jr} = 1)$, or
   (b) $(a_{ij} = 1) \wedge (\forall t,\, (a_{ti} = 1 \Rightarrow P_{tr} = 0))$.
2. $E_{i^*j^*}$ represents $(\mathcal{F}_2, \mathcal{F}_3)$ and one of the following conditions hold:
   (a) $(a_{j^*i^*} = 1) \wedge (P_{j^*r} = 1)$, or
   (b) $(a_{i^*j^*} = 1) \wedge (\forall t,\, (a_{ti^*} = 1 \Rightarrow P_{tr} = 0))$.

If (1a) and (2a) hold, then $a_{ji} = 1$ and $a_{j^*i^*} = 1$ and since the relation $\succeq_h$ is transitive and $i^* = j$, then $a_{j^*i} = 1$. Moreover, from (2a), we have that $P_{j^*r} = 1$ and consequently (12) holds.
Now suppose that (1b) and (2b) hold. Again by transitivity of $\succeq_h$ we have that $a_{ij^*} = 1$. Moreover, condition (2b) implies (13).
If (1a) and (2b) hold, then $P_{jr} = 1$ and $P_{i^*r} = P_{jr} = 0$ which is a contradiction.
Similarly, if (1b) and (2a) hold, then $a_{j^*i} = 1$ and $P_{j^*r} = 1$ and taking $t = j^*$ in (2a), we have that $P_{j^*r} = 0$. Again a contradiction.
Hence, neither conditions (1a) and (2b) nor conditions (1b) and (2a) occur at the same time. This finishes the proof. ∎

The previous lemma can be generalized using mathematical induction to obtain the following result.

Corollary 1 *Suppose that the permutation matrices $E_{i_1j_1}, E_{i_2j_2}, \ldots, E_{i_mj_m}$ represent consecutive rational deals associated to the same resource r. Then $j_1 = i_2$, $j_2 = i_3, \cdots, j_{m-1} = i_m$, and $E_{i_1j_m} = E_{i_mj_m} \cdots E_{i_2j_2}E_{i_1j_1}$ is a permutation matrix that also represents a rational deal associated to the resource r.*

Good allocations

Our goal will be to construct an algorithm to find optimal resource allocations. Before, we will define what "good" resource allocations are.

Definition 10 *Let $\mathcal{F} = (f_{in})_{q\times k}$ be a resource allocation and fix $r \in \mathcal{R}$. Suppose $i \in \mathcal{A}$ is the only agent such that $f_{ir} = 1$. We will say that the column $c_r = (f_{ir})_{1\leq i\leq q}$, associated to the resource r, is in* good position *if*

$$(P_{ir} = 1) \wedge (\forall t(a_{ti} = 1 \wedge a_{it} = 0) \Rightarrow P_{tr} = 0)) \tag{14}$$

or

$$(\forall t(P_{tr} = 0)) \wedge (i \in \min(\mathcal{A})) \tag{15}$$

where $\min(\mathcal{A}) = \{t \in \mathcal{A} : a_{it} = 1, \forall i \in \mathcal{A}\}$.

We will say the $\mathcal{F}$ is a good resource allocation *if all its columns are in good position.*

Condition (14) can be interpreted as: Agent i requests the resource r and every other agent with higher hierarchy does not request it. Condition (15) says that agent i has the lowest possible hierarchy and no agents requests the resource r. A good resource allocation will then be an allocation in which each resource is either assigned to an agent that requests it and has the higher hierarchy among those that request the resource, or to an agent of the lowest hierarchy if no agent requests the resource.

The following lemma says that a column that is in good position cannot be further "improved" by using a rational deal.

Lemma 3 *Suppose that the r-th column c_r of a resource allocation $\mathcal{F}$ is in good position and that there exists a resource allocation $\mathcal{G}$ such that $(\mathcal{F}, \mathcal{G})$ is a rational deal associated to c_r, represented by the matrix E_{ij}. Then c_r and $E_{ij}c_r$ are equivalent.*

Proof: Suppose $\mathcal{F} = [c_1, \ldots, c_r, \cdots, c_k]$, we will prove that $a_{ij} = 1 = a_{ji}$ and $P_{ir} = P_{jr}$. Since $\succeq_h$ is total we consider the following two cases:

If $a_{ji} = 1$, then condition (10) holds and consequently we have that $P_{jr} = 1$ and condition (15) does not hold. But since c_r is in good position, then (14) holds and consequently $P_{ir} = 1$ and $a_{ij} = 1$. Hence c_r and $E_{ij}c_r$ are r-equivalent.

Condition $a_{ij} = 1$ and $a_{ji} = 0$ is impossible since if this were the case, then condition (11) holds and consequently $P_{ir} = 0$. Moreover $i \notin \min(A)$. This two facts contradict conditions (14) and (15), respectively. This finishes the proof. ∎

The following lemma says that if the column involved in a simple deal turns out to be in good position, then the deal must be rational.

Lemma 4 *Suppose $\mathcal{F}$ and $\mathcal{G}$ are two resource allocations and that $(\mathcal{F}, \mathcal{G})$ is a simple deal that converts column c_r to $E_{ij}c_r$. If $E_{ij}c_r$ is in good position and it is not equivalent to c_r, then $(\mathcal{F}, \mathcal{G})$ is a rational deal.*

Proof: We need to show that one of the conditions (10) or (11) hold. We will use again the totality of $\succeq_h$ and consider cases. Suppose that $a_{ji} = 1$, then we have that $j \notin \min(\mathcal{A})$ since otherwise $a_{ij} = 1$ and by condition (15) $P_{ir} = 0 = P_{jr}$, contradicting the hypothesis that $E_{ij}c_r$ and c_r are not equivalent. Hence, condition (14) holds and consequently $P_{jr} = 1$ and (10) holds.

Now suppose that $a_{ij} = 1$ and $a_{ji} = 0$, if condition (15) holds, then in particular we have that condition (11) holds.

If on the contrary, condition (14) holds, then $P_{jr} = 1$ and $P_{ir} = 0$ and moreover, for every $t \in \mathcal{A}$ such that $a_{ti} = 1$ we have by transitivity that $a_{tj} = 1$ and $a_{jt} = 0$ which, by condition (14) implies that $P_{tn} = 0$ proving that condition (11) holds. ∎

Remark 2 *The converse of the previous lemma is not generally true. That is, the resulting column involved in a rational deal, needs not to be in good position.*

Lemma 5 *If $\mathcal{F} = [c_1, \cdots, c_r, \cdots c_k]$ is resource allocation such that the column c_r is not good position, then there exists a rational deal which transforms it in a good column.*

Proof: Let c_r the column in not good position. We want to find a resource allocation $\mathcal{G}$ such that $(\mathcal{F}, \mathcal{G})$ is a rational deal associated to the resource in the position r and that leaves c_r in good position.
Let $i \in \mathcal{A}$ be the only agent such that $f_{ir} = 1$. Define

$$I_r = \{t \in \mathcal{A} : t \neq i \wedge P_{tr} = 1\}$$

If $I_r = \emptyset$, let j be any element in $\min(\mathcal{A})$. If $I_r \neq \emptyset$, let j be any element in $\max(I_r)$.
Now, if $\mathcal{F} = (f_{in})_{q\times k} = [c_1, \cdots, c_r \cdots, c_k]$, let's define the resource allocation $\mathcal{G} = [c_1, \cdots, E_{ij}c_r \cdots, c_k]$. Clearly $(\mathcal{F}, \mathcal{G})$ is a simple deal. To show that it is actually a rational deal, we will show that $E_{ij}c_r$ is in good position, and since it is not equivalent to c_r, and then use lemma 4.
We will consider each case, $I_r = \emptyset$ and $I_r \neq \emptyset$, separately:
Suppose first that $I_r = \emptyset$, then $\forall t \in \mathcal{A}$, $t \neq i$ we have that $P_{tr} = 0$. Thus $P_{ir} = 0$ since otherwise condition (14) would hold and c_r would be in good position, contradicting the hypothesis. Consequently, for every $t \in \mathcal{A}$, $P_{tr} = 0$ and since $j \in \min(A)$, then condition (15) holds. Hence, $E_{ij}c_r$ is in good position.
Now suppose that $I_r \neq \emptyset$, then $P_{jr} = 1$ and $j \in \max(I_r)$. Consequently, if $t \in \mathcal{A}$ is such that $t \neq i$, $a_{tj} = 1$ and $a_{jt} = 0$, we have that $j \notin I_r$ and therefore $P_{tr} = 0$. Moreover, it can not happen that $a_{ij} = 1$ and $a_{ji} = 0$ and $P_{ir} = 1$ since the maximality of j will then imply that $P_{tr} = 0$ for every $t \in \mathcal{A}$ such that $a_{ti} = 1$ but this implies that condition (14) holds for i reaching a contradiction. Thus, condition (14) holds for j and hence $E_{ij}c_r$ is in good position. ∎

The following theorem shows that any resource allocation can be converted into a good resource allocation by means of a finite sequence of rational deals. The proof of the theorem gives us an algorithm that will later result in an optimal resource allocation as we will see in Corollary 2.

Theorem 1 *Suppose $\mathcal{F}$ is not a good resource allocation and let l be the number of columns of $\mathcal{F}$ that are not in good position. Then there exists a sequence of rational deals $\mathcal{F}_1, \cdots, \mathcal{F}_{l+1}$ such that $\mathcal{F}_1 = \mathcal{F}$ and $\mathcal{F}_{l+1}$ is a good allocation.*

Proof: Let $\mathcal{C} = \{c_{n_1}, \cdots, c_{n_l}\}$, $1 \leq n_1 < \cdots < n_l \leq k$, the set of columns that are not in good position of $\mathcal{F}$. Obviously, $l \geq 1$. For c_{n_1}, define

$$\mathcal{F}_1 = \mathcal{F} = (c_1, \cdots, c_{n_1}, \cdots, c_k).$$

By Lemma 5, there exists a rational deal $(\mathcal{F}_1, \mathcal{F}_2)$ associated to c_{n_1} such that

$$\mathcal{F}_2 = (c_1, \cdots, E_{i_1 j_1}c_{n_1}, \cdots, c_k)$$

and $E_{i_1 j_1}c_{n_1}$ is in a good position. If $l = 1$, then $\mathcal{F}_2$ is a good allocation and this finishes the proof.
Now, suppose that $l > 1$. Using the same reasoning, construct the resource allocations $\mathcal{F}_3, \ldots, \mathcal{F}_{l+1}$ until a good resource allocation is reached. ∎

Optimal resource allocations

In this subsection we will show that good resource allocations are actually optimal under the Definition 6 as long as the relation $\sqsupseteq$ is positive. First we need two technical lemmas.

Lemma 6 *Let $\mathcal{F}$ and $\mathcal{G}$ be two resource allocations, then*

$$D_{\mathcal{FG}} = [\mathcal{F} \succ \mathcal{G}] \cup [\mathcal{G} \succ \mathcal{F}] \quad \textit{and} \quad [\mathcal{F} \succ \mathcal{G}] \cap [\mathcal{G} \succ \mathcal{F}] = \emptyset$$

Proof: If $D_{\mathcal{FG}} = \emptyset$, then $[\mathcal{F} \succ \mathcal{G}] = \emptyset = [\mathcal{G} \succ \mathcal{F}]$. Suppose that $D_{\mathcal{FG}} \neq \emptyset$. It is clear that $[\mathcal{F} \succ \mathcal{G}] \cup [\mathcal{G} \succ \mathcal{F}] \subseteq D_{\mathcal{FG}}$. It remains to show that $D_{\mathcal{FG}} \subseteq [\mathcal{F} \succ \mathcal{G}] \cup [\mathcal{G} \succ \mathcal{F}]$. Let $r \in D_{\mathcal{FG}}$ and let $i, j \in \mathcal{A}$ be the only agents such that $f_{ir} = 1 = g_{jr}$.

Suppose first that $a_{ij} \neq a_{ji}$, then from equation (5) we have that either

$$a^r_{ij} = P_{ir} \quad \text{and} \quad a^r_{ji} = 1 - P_{ir}$$

or

$$a^r_{ij} = 1 - P_{jr} \quad \text{and} \quad a^r_{ji} = P_{jr}.$$

In any of these cases, either $a^r_{ij} = 1$ or $a^r_{ji} = 1$. Thus, $r \in [\mathcal{F} \succ \mathcal{G}] \cup [\mathcal{G} \succ \mathcal{F}]$.
Suppose now that $a_{ij} = a_{ji}$. Then since $r \in D_{\mathcal{FG}}$ we have that $P_{ir} \neq P_{ir}$. Again by equation (5) we have that either

$$a^r_{ij} = P_{ir} \quad \text{and} \quad a^r_{ji} = P_{jr}$$

or

$$a^r_{ij} = 1 - P_{jr} \quad \text{and} \quad a^r_{ji} = 1 - P_{ir}.$$

And since $P_{ir} \neq P_{jr}$, the previous statement implies that $r \in [\mathcal{F} \succ \mathcal{G}]$ or $r \in [\mathcal{G} \succ \mathcal{F}]$. Hence, $D_{\mathcal{FG}} \subseteq [\mathcal{F} \succ \mathcal{G}] \cup [\mathcal{G} \succ \mathcal{F}]$.

On the other hand, suppose that $r \in [\mathcal{F} \succ \mathcal{G}] \cap [\mathcal{G} \succ \mathcal{F}]$. Then $r \in D_{\mathcal{FG}}$ and there exist agents $i, j \in \mathcal{A}$ such that $f_{ir} = 1 = g_{jr}$, and $a^r_{ij} = 1 = a^r_{ji}$.

From equation (5), exactly one of the following two possibilities happens:

$$a_{ij}P_{ir} = 1 = a_{ji}P_{jr}$$

or

$$(1 - a_{ij})(1 - P_{jr}) = 1 = (1 - a_{ji})(1 - P_{ir})$$

In both cases, $a_{ij} = a_{ji}$ and $P_{ir} = P_{jr}$ contradicting the hypothesis that $c^l_r \not\equiv_r c^p_r$. Hence, $[\mathcal{F} \succ \mathcal{G}] \cap [\mathcal{G} \succ \mathcal{F}] = \emptyset$.

∎

Proposition 1 *Let $\mathcal{F}_1$,$\mathcal{F}_1$,...,$\mathcal{F}_n$ be a sequence of rational deals. Then for every $1 < p \leq n$, we have that $D_{1,p} = [\mathcal{F}_p \succ \mathcal{F}_1]$, where $D_{1,p}$ is an abbreviation of for $D_{\mathcal{F}_1\mathcal{F}_p}$.*

Proof: Let $1 < p \leq n$ be fixed. Clearly,

$$[\mathcal{F}_p \succ \mathcal{F}_1] \subset D_{1,p}$$

It remains to show that $D_{1,p} \subset [\mathcal{F}_p \succ \mathcal{F}_1]$. Let $r \in D_{1,p}$ and let $i, j \in \mathcal{A}$ be the only agents such that $f^1_{ir} = 1 = f^p_{jr}$. We will show that

$$a^r_{ji} = 1 = a_{ji}P_{jr} + (1 - a_{ji})(1 - P_{ir})$$

Let $l \leq p$ be the number of rational deals from the sequence $\mathcal{F}_1$,$\mathcal{F}_2, \ldots$,$\mathcal{F}_p$ associated to the resource r. Let $\mathcal{G}_1, \mathcal{G}_2, \cdots \mathcal{G}_{l+1}$ be the subsequence of rational deals of $\mathcal{F}_1$,$\mathcal{F}_2, \ldots$,$\mathcal{F}_p$ that involve only the resource r. $\mathcal{G}_1 = \mathcal{F}_1$ and $\mathcal{G}_l = \mathcal{F}_p$. Let $E_{i_1 j_1}, E_{i_2 j_2}, \cdots, E_{i_l j_l}$ be the permutation matrices that represent the rational deals $(\mathcal{G}_1, \mathcal{G}_2), \cdots,$ $(\mathcal{G}_l, \mathcal{G}_{l+1})$, respectively. Notice that $i = i_1$, $j_1 = i_2$, $j_2 = i_3, \cdots, j_{l-1} = i_l$, and $j_l = j$. Using corollary 1, we get that

$$E_{ij} = E_{i_l j_l} \cdots E_{i_2 j_2} E_{i_1 j_1}$$

represents a rational deal associated to r. Thus, by Definition 8, it holds that either

$$a_{ji}P_{jr} = 1 \tag{16}$$

or

$$(a_{ij} = 1) \wedge (P_{ir} = 0) \wedge (\forall k(a_{ki} = 1 \Rightarrow P_{kr} = 0)). \tag{17}$$

If equation (16) holds, then $a^r_{ji} = a_{ji}P_{jr} = 1$ and consequently $r \in [\mathcal{F}_p \succ \mathcal{F}_1]$.

Let's suppose equation (17) holds. Since $c^1_r \not\equiv_r c^p_r$ then either $a_{ij} \neq a_{ji}$ or $a_{ij} = a_{ji}$ and $P_{ir} \neq P_{jr}$.

If $a_{ij} \neq a_{ji}$ then $a_{ji} = 0$ and since $P_{ir} = 0$, then $(1 - a_{ji})(1 - P_{ir}) = 1 = a^r_{ji}$.

On the other hand, if $a_{ij} = a_{ji}$ and $P_{ir} \neq P_{jr}$, then $a_{ji} = 1$ and $P_{ir} = 1$ and consequently, $a^r_{ji} = a_{ji}P_{jr} = 1$. In both cases we have that $r \in [\mathcal{F}_p \succ \mathcal{F}_1]$.

Thus $D_{1p} \subset [\mathcal{F}_p \succ \mathcal{F}_1]$ and this finishes the proof. ∎

Lemma 7 *Suppose $\mathcal{G}$ is a good resource allocation and $\mathcal{F}$ is any resource allocation. Then $[\mathcal{G} \succ \mathcal{F}] = D_{\mathcal{G}\mathcal{F}}$*

Proof: It is enough to show that $D_{\mathcal{G},\mathcal{F}} \subset [\mathcal{G} \succ \mathcal{F}]$. Let $r \in D_{\mathcal{G},\mathcal{F}}$ and suppose that $i, k \in \mathcal{A}$ are the only agents such that $g_{ir} = 1$ and $f_{kr} = 1$.

We want to show that $a^r_{ik} = 1$ where

$$a^r_{ik} = a_{ik}P_{ir} + (1 - a_{ik})(1 - P_{kr}) \tag{18}$$

Case 1: $P_{ir} = 1$. Then either $a_{ik} = 1$ or $a_{ki} = 0$.

If $a_{ik} = 1$, then pluging in equation (18), we have that $a^r_{ik} = 1$.

If $a_{ik} = 0$, then since $\succeq_h$ is total, then $a_{ki} = 1$. Now, since the any column of the allocation $\mathcal{G}$ is in good position, then condition and $P_{ir} = 1$, then (14) holds:

$$(P_{ir} = 1) \wedge (\forall j(a_{ji} = 1 \wedge a_{ij} = 0) \Rightarrow P_{jm} = 0).$$

Thus, $a_{ki} = 1$ and $a_{ik} = 0$ imply that $P_{kr} = 0$. Plugging in equation (18), we conclude that $a^r_{ik} = 1$.

Case 2: $P_{ir} = 0$. Again using the hypothesis that any column of $\mathcal{G}$ is in good position, condition (15) holds:

$$\forall j(P_{jr} = 0) \wedge (i \in \min(\mathcal{A})).$$

In consequence, $P_{kr} = 0$ and $a_{ki} = 1$. Hence $P_{kr} = P_{ir}$ and since the r columns in $\mathcal{F}$ and $\mathcal{G}$ are not equivalent, then we must have that $a_{ik} = 0$. Pluging in equation 18, we obtain $a^r_{ik} = 1$.

This finishes the proof. ∎

The next corollary follows from Lemmas 6 and 7.

Corollary 2 *Suppose $\mathcal{G}$ is a good resource allocation, then for every resource allocation $\mathcal{F}$ we have that $[\mathcal{F} \succ \mathcal{G}] = \emptyset$.*

We are finally ready to prove that good resource allocations and optimal resource allocations coincide when the binary relation $\sqsupseteq$ is positive in the following sense:

Definition 11 *Given a binary relation $\sqsupseteq$ defined on $\mathcal{P}(\mathcal{R})$, we say that $\sqsupseteq$ is* positive *if $\emptyset \sqsupseteq \emptyset$ and for every set $A \in \mathcal{P}(\mathcal{R})$, with $A \neq \emptyset$ we have that*

$$A \sqsupseteq \emptyset \wedge \emptyset \not\sqsupseteq A.$$

Theorem 2 *Suppose $\sqsupseteq$ is positive. Then any good resource allocation is also an optimal resource allocation and vice versa.*

Proof: Let $\mathcal{G}$ be a good resource allocation and $\mathcal{F}$ be any resource allocation, then by Lemma 7 and Corollary 2 we have that

$$[\mathcal{G} \succ \mathcal{F}] = D_{\mathcal{G}\mathcal{F}} \sqsupseteq \emptyset = [\mathcal{F} \succ \mathcal{G}]$$

and consequently $\mathcal{G} \trianglerighteq \mathcal{F}$. Thus $\mathcal{G}$ is optimal.

Conversely, suppose $\mathcal{G}$ not a good resource allocation, then by Theorem 1 and Proposition 1, there exists a good resource allocation $\mathcal{F}$ such that $[\mathcal{F} \succ \mathcal{G}] = D_{\mathcal{F}\mathcal{G}}$, consequently $[\mathcal{F} \succ \mathcal{G}] = \emptyset$ and by the positivity of $\sqsupseteq$, we have that $\mathcal{F} \triangleright \mathcal{G}$. Thus $\mathcal{G}$ is not optimal. ∎

Remark 3 *For any two optimal resource allocations $\mathcal{F}$ and $\mathcal{G}$ it holds that $\mathcal{F} \trianglerighteq \mathcal{G}$ and $\mathcal{G} \trianglerighteq \mathcal{F}$.*

Corollary 3 *Suppose $\sqsupseteq$ is positive, then for every resource allocation $\mathcal{F}$ there exists an optimal resource allocation $\mathcal{F}^*$ such that $\mathcal{F}^* \trianglerighteq \mathcal{F}$.*

The following corollary justifies the term "optimal allocation" in the sense of the relation $\trianglerighteq$.

Corollary 4 *Suppose $\sqsupseteq$ is positive. Suppose also that $\mathcal{G}$ is an optimal resource allocation and $\mathcal{F}$ is a non-optimal resource allocation. Then $\mathcal{G} \triangleright \mathcal{F}$.*

Discriminating good allocations

In this section we search to discriminate good allocations. We use a criterion of local satisfaction to compare the the good allocations the relation optimal with respect this criterion will be the the allocation locally fair.

In order to understand the problems of lack of discrimination let us see the following example:

Example 2 *Let* $A = \{1,2,3\}$ *and* $\mathcal{R} = \{r_1, r_2, r_3, r_4, r_5, r_6\}$. *Suppose that all the agents have equal hierarchy; moreover Agent 3 requests all the resources except excepto* r_6*; Agent 2 requests the resources* r_1*,* r_2*,* r_3 *and* r_4 *but he doesn't request* r_5 *nor* r_6*, whereas agente 1, request the first 3 resources. Then the request matrix* $\mathcal{P}$ *and the hierarchy matrix* $\mathcal{A}_h$ *are*

$$\mathcal{P} = \left(\begin{array}{c|cccccc} & r_1 & r_2 & r_3 & r_4 & r_5 & r_6 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 1 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 & 1 & 1 & 0 \end{array}\right)$$

$$A_h = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Notice that we can make a partition of $\mathcal{R}$ *in the subsets* $\{r_1, r_2\, r_3\}$, $\{r_4\}$, $\{r_5\}$ *and* $\{r_6\}$. *Any good allocation distributes the resources* $\{r_1,\ r_2,\ r_3\}$ *among the agents* $\{1,2,3\}$*;* $\{r_4\}$ *among the agents* $\{2,3\}$*; the resource* $\{r_5\}$ *to agent 5 and the resource* $\{r_6\}$ *among the agents* $\{1,2,3\}$. *Now consider the following allocations:*

$$\mathcal{E} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \mathcal{F} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

$$\mathcal{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}, \mathcal{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Although all four allocations are good, it seems that some are fairer *than others. Indeed,* $\mathcal{E}$ *looks very unfair because all the resources are allocated to the third agent. For example allocation* $\mathcal{F}$ *distributes better the resources than* $\mathcal{E}$. *If we now focus only on the subset* $\{r_1, r_2, r_3\}$*, then* $\mathcal{G}$ *and* $\mathcal{H}$ *have a better* local *distribution than* $\mathcal{F}$. *It is then natural to think that we need a criterion to better distinguish among good allocations. We will develop local criteria in order to distinguish good allocations. These local criteria will be aggregate in a social criterion.*

Let $\mathcal{B}$ the set of good allocations. For each r in $\mathcal{R}$, we define

$$A(r) = \{j \in \mathcal{A} : \exists \mathcal{F} = (f_{in}) \in \mathcal{B} \text{ such that } f_{jr} = 1\} \tag{19}$$

It is clear that $A(r) \neq \emptyset$. since otherwise, $\mathcal{B} = \emptyset$. Define $\mathcal{R}^* = \{s \in \mathcal{R} : \mathcal{P}_{is} = 0, \forall i \in \mathcal{A}\}$, that is, the set of resources that nobody requests. Notice that for every $r \in \mathcal{R}^*$, $A(r) = \min(\mathcal{A})$. If $r \notin \mathcal{R}^*$ and $i, j \in A(r)$ then i and j request r, and moreover both agents have the same hierarchy.

On $\mathcal{R} \setminus \mathcal{R}^*$ we define the equivalence relation $\sim$ as

$$r \sim s \Leftrightarrow A(r) = A(s).$$

We denote by $[r] = \{s \in \mathcal{R} \setminus \mathcal{R}^* : A(s) = A(r)\}$ the equivalence class of r on $\mathcal{R} \setminus \mathcal{R}^*$. Suppose that there are l different classes then

$$\mathcal{R} \setminus \mathcal{R}^* = [r_{m_1}] \cup [r_{m_2}] \cup \cdots \cup [r_{m_l}] \tag{20}$$

If we put $\mathcal{R}^* = [r_{m_{l+1}}]$, then we have the partition $\mathcal{R} = [r_{m_1}] \cup [r_{m_2}] \cdots [r_{m_l}] \cup [r_{m_{l+1}}]$.

Let $\mathcal{F} = (f_{in})$ be an allocation and $r \in \mathcal{R}$. We will denote as $\mathcal{F}_{[r]}$ the submatrix $A(r) \times [r]$ of $\mathcal{F}$ composed by the rows with the agents in $A(r)$ and the columns with the resources in $[r]$.

For every agent $i \in A(r)$, the number of resources of $[r]$ allocated to a i via $\mathcal{F}$ is $\sum_{s \in [r]} f_{is}$. Consider the average of these values:

$$\overline{X}(\mathcal{F}_{[r]}) = \sum_{i \in A(r)} \sum_{s \in [r]} \frac{f_{is}}{|A(r)|}. \tag{21}$$

It is easy to see that all good allocations have the same average on the submatrix $A(r) \times [r]$. Moreover,

$$\overline{X}(\mathcal{F}_{[r]}) = \frac{|[r]|}{|A(r)|}. \tag{22}$$

We consider now a dispersion measure with respect to this average denoted $var(\mathcal{F}_{[r]})$ defined as

$$var(\mathcal{F}_{[r]}) = \sum_{i \in A(r)} \sum_{s \in [r]} \frac{(f_{is} - \overline{X}_{\mathcal{F}_{[r]}})^2}{|A(r)|}, \tag{23}$$

and we define the dispersion vector of $\mathcal{F}$ by

$$\gamma(F) = (var(\mathcal{F}_{[r_1]}), var(\mathcal{F}_{[r_2]}), \cdots, var(\mathcal{F}_{[r_{l+1}]})). \tag{24}$$

Using this vector we define a new relation on the set of good allocations.

Definition 12 *Let* $\mathcal{F}$ *and* $\mathcal{G}$ *good allocations. We say that* $\mathcal{F}$ *gives more local satisfaction than* $\mathcal{G}$*, denoted* $\mathcal{F} \succeq_\gamma \mathcal{G}$*, if for all* $i = 1 \cdots l+1$ *we have* $(\gamma(\mathcal{G}))_i \geq (\gamma(\mathcal{F}))_i$. *If for every* $\mathcal{G} \in \mathcal{B}$*, we have* $\mathcal{F} \geq_\gamma \mathcal{G}$*, we say that* $\mathcal{F}$ *is locally fair.*

We say that $\mathcal{F} \succ_\gamma G$ iff $\mathcal{F} \succeq_\gamma \mathcal{G}$ and there exists k such that $(\gamma(G))_k > (\gamma(F))_k$. Whereas $\mathcal{F} \sim_\gamma G$ iff for every $i = 1 \cdots l+1$ we have $(\gamma(\mathcal{G}))_i = (\gamma(\mathcal{F}))_i$. It is clear that the relation $\succeq_\gamma$ is not a total preorder on $\mathcal{B}$. The following example illustrates the previous concepts.

Example 3 *Let's go back to example* (2) *Notice that,* $A(r_{m_1}) = A(r_1) = A(r_2) = A(r_3) = \{1,2,3\}$*;* $A(r_{m_2}) = A(r_4) = \{2,3\}$*,* $A(r_{m_3}) = A(r_5) = \{3\}$ *and* $A(r_{m_4}) = A(r_6) = \{1,2,3\}$. *It is also easy to check that* $[r_{m_1}] = \{r_1,\ r_2,\ r_3\}$, $[r_{m_2}] = [r_4] = \{r_4\}$, $[r_{m_3}] = [r_5] = \{r_5\}$ *and* $[r_{m_4}] = [r_6] = \{r_6\}$. *Using equation* (22)*, we have that in* $A(r_{m_1}) \times [r_{m_1}]$ *the average of any allocation in* $\mathcal{B}$ *is* 1*; in* $A(r_{m_2}) \times [r_{m_2}]$ *the average is* $1/2$*; in* $A(r_{m_3}) \times [r_{m_3}]$ *the average is* 1*; whereas in* $A(r_{m_4}) \times [r_{m_4}]$ *the average is* $1/3$.

Now by equations (23) *and* (24) *we have the dispersion vectors.*

$$\gamma(\mathcal{E}) = (2, 1/4, 0, 2/9) \qquad \gamma(\mathcal{F}) = (2/3, 1/4, 0, 2/9)$$

$$\gamma(\mathcal{G}) = (0, 1/4, 0, 2/9) \qquad \gamma(\mathcal{H}) = (0, 1/4, 0, 2/9)$$

Now, according to Definition 12, allocations $\mathcal{G}$ *and* $\mathcal{H}$ *give more local satisfaction than* $\mathcal{F}$ *which in turn, gives more local satisfaction than* $\mathcal{E}$. *That is,* $\mathcal{G} \sim_\gamma \mathcal{H}$ *and* $\mathcal{F} \succ_\gamma \mathcal{E}$. *Moreover, it can be shown that* $\mathcal{G}$ *and* $\mathcal{H}$ *are locally fair allocations.*

Although the relation $\geq_\gamma$ is not total on $\mathcal{B}$, we will show that there always exists a locally fair allocation. We will need the following technical Lemma.

Lemma 8 *Let $x_1, x_2, \cdots, x_n$ be positive integers. If $\sum_{i=1}^{n} x_i = L$, there exists a natural number $\beta \leq n-1$, such that*

$$\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \frac{L}{n}\right)^2 \geq \frac{(n-\beta)\beta}{n^2} \tag{25}$$

Proof: For every $1 \leq i \leq n$, $x_i \geq 1$, therefore $L \geq n$. Let α and β be the unique natural numbers such that $L = \sum_{i=1}^{n} x_i = n\alpha + \beta$ with $\beta \in \{0, \cdots, n-1\}$. Notice that, for each i, $x_i = \alpha$ or $x_i = \alpha + 1$. If $x_i = \alpha$ for every i, then $L = n\alpha$. consequently, $\beta = 0$ and (25) holds. Now suppose that there exist m natural numbers such that $x_i = \alpha + 1$. If $m = n$ then $\sum_{i=1}^{n} x_i = n\alpha + n \neq L$. Hence, $m < n$. On the other hand,

$$L = \sum_{i=1}^{m} x_i + \sum_{i=m+1}^{n} x_i = m(\alpha+1) + (n-m)\alpha = n\alpha + m$$

and therefore, $m = \beta$. Now,

$$\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \frac{L}{n}\right)^2 =$$
$$\frac{1}{n}\left[\sum_{i=1}^{m}\left(\alpha+1-\alpha-\frac{m}{n}\right)^2 + \sum_{i=m+1}^{n}\left(\alpha-\alpha-\frac{m}{n}\right)^2\right] =$$
$$\frac{1}{n}\left[\sum_{i=1}^{m}\left(\frac{n-m}{n}\right)^2 + \sum_{i=m+1}^{n}\left(\frac{m}{n}\right)^2\right] =$$
$$\frac{1}{n}\left[m\frac{(n-m)^2}{n^2} + (n-m)\frac{m^2}{n^2}\right] = \frac{m(n-m)}{n^2}$$

and, since $m = \beta$, the equation (25) holds. ∎

Theorem 3 *There exists a good allocation $\mathcal{F}^*$ that is locally fair.*

Proof: We want to find $\mathcal{F}^* \in \mathcal{B}$ such that $\gamma(\mathcal{F}^*) \leq \gamma(\mathcal{G})$ for all $\mathcal{G} \in \mathcal{B}$. Let's consider the partition of $\mathcal{R}$ as in equation (20).

For each $t = 1, \ldots, l+1$, there exists unique nonnegative integers α_t and β_t such that

$$|[r_{m_t}]| = |A(r_{m_t})|\alpha_t + \beta_t \tag{26}$$

with $\beta_t \in \{0, 1, \ldots, |A(r_{m_t})| - 1\}$.

For each $t \in \{1, \ldots l+1\}$, choose β_t agents in $A(r_{m_t})$. Let's say $\{j_1, \ldots j_{\beta_t}\} \subseteq A(r_{m_t})$.

Let's choose $\mathcal{F}^* = (f^*_{in}) \in \mathcal{B}$ such that for each $t \in \{1, \ldots l+1\}$,

$$\sum_{s\in[r_{m_t}]} f^*_{js} = \begin{cases} \alpha_{r_t} + 1, & \text{if } j \in \{j_1, \ldots j_{\beta_t}\} \\ \alpha_{r_t}, & \text{if } j \in A(r_{m_t}) \setminus \{j_1, \ldots j_{\beta_t}.\} \end{cases}$$

Now, on each submatrix $A(r_{m_t}) \times [r_{m_t}]$ we have that

$$\overline{X}(\mathcal{F}^*_{[r_t]}) = \alpha_{r_t} + \frac{\beta_{r_t}}{|A(r_t)|} \quad \text{and}$$

$$var(\mathcal{F}^*_{[r_t]}) = \beta_t \frac{|A(r_t)| - \beta_t}{|A(r_t)|^2}.$$

On the other hand, for every $\mathcal{G} \in \mathcal{B}$ and $t = 1, \ldots, l+1$, by equations (22) and (23) we have that

$$var(G_{[r_{m_t}]}) = \frac{1}{|A(r_{m_t})|}\sum_{i=1}^{|A(r)|}\left(\sum_{s\in[r_{m_t}]} g_{is} - \frac{|[r_{m_t}]|}{A(r_{m_t})}\right)^2$$

Finally by Lemma 8,

$$var(G_{[r_{m_t}]}) \geq var(F^*_{[r_{m_t}]})$$

and consequently, for all $\mathcal{G} \in \mathcal{B}$

$$\gamma(\mathcal{F}^*) \leq \gamma(\mathcal{G})$$

∎

The following result tells us that for the good allocations, the dispersion measure is invariant under any permutation in the submatrix $A(r) \times [r]$. For instance, in Example 3, the allocations $\mathcal{G}$ and $\mathcal{H}$ are different in the column 6. But,

$$\mathcal{G}_{[r_{m_4}]} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathcal{H}_{[r_{m_4}]}.$$

Consequently the allocation $\mathcal{F}^*$ in the previous theorem is not necessarily unique.

Proposition 2 *Supose that $\mathcal{R} = [r_{m_1}] \cup [r_{m_2}] \cup \cdots \cup [r_{m_l}] \cup [r_{m_{l+1}}]$, where l is the number of different equivalent classes of $\mathcal{R} \setminus \mathcal{R}^*$ and $[r_{m_{l+1}}] = \mathcal{R}^*$. Let $\mathcal{F}$ be a good allocation. If Q_1, Q_2 $\cdots Q_{l+1}$ are permutation matrices of the identity $I_{[r_{m_i}]}$ for every $i = 1, \cdots, l+1$, then*

$$\gamma(\mathcal{F}) = \gamma(\mathcal{G})$$

where $\mathcal{G}_{[r_{m_i}]} = Q_i \cdot \mathcal{F}_{[r_{m_i}]}$ for every $i = 1, \cdots, l+1$.

Proof:

$$\gamma(\mathcal{F}) = (var(\mathcal{F}_{[r_{m_1}]}), \cdots, var(\mathcal{F}_{[r_{l+1}]})) =$$
$$(var(Q_1 \cdot \mathcal{F}_{[r_{m_i}]}), \cdots, var(Q_{l+1} \cdot \mathcal{F}_{[r_{m_{l+1}}]})) =$$
$$\gamma(\mathcal{G})$$

∎

Final Remarks

In this work we have tackled the problem of finding a good resource allocation from a qualitative point of view. A central idea is to use a notion of qualitative social welfare based on the dominance plausible rule of Dubois et al. (Dubois et al. 2002; Dubois, Fargier, and Perny 2003). One important feature of this approach is to take into account the uncertainty of resources via the qualitative relation representing the likelihood of the resources. The use of matrix techniques of representation for the relations involved allows us a better organization of results and a simplification of proofs.

We have generalized the preliminary results presented by Pino Pérez et al. (Pino Peréz, Varela Montilva and Camacho 2016) in three ways:

1. The hierarchical relation between agents can be now a total preorder instead of a linear order.
2. The likelihood relation between the resources can be now a total preorder instead of linear order.
3. The plausibility relation between sets of resources can be now a positive relation extending the likelihood relation between the resources instead of only the posibilistic relation considered in (Pino Peréz, Varela Montilva and Camacho 2016).

Unlike the results in (Pino Peréz, Varela Montilva and Camacho 2016) in which an optimal allocation is unique, here the class of optimal allocations can contain more than one element. Examples where this situation occurs (given in Section *Discriminating good allocations*) motivated the study done in that Section where we propose the first clues for discriminating optimal allocations. In this direction, we have introduced some local techniques and give one technique to globally compare optimal allocations. A future work is to continue the quest of adequate techniques for discriminating these optimal allocations.

We have to note that in this work the attitude of an agent toward a resource is binary, that is, the agent requests or not the resource. Thus, another natural generalization of this work could be to consider more complex representations of the agents' preferences.

References

Barberà, S.; Bossert, W.; and Pattanaik, P. K. 2004. Ranking sets of objects. In Barberà, S. and Hammond, P.J. and Seidl, Eds. Volume 2 of *Handbook of Utility Theory*, 893–978. Springer.

Beviá, C. 1996. Identical preferences lower bound solution and consistency in economies with indivisible goods. *Social Choice and Welfare* 13:113–126.

Brams, S. J., and Taylor, A. D. 1995. An envy-free cake division protocol. *The American Mathematical Monthly* 102:9–18.

Brams, S. J., and Taylor, A. D. 1996. *Fair division - from cake-cutting to dispute resolution*. Cambridge University Press.

Brams, S. J., Taylor, A. D. and Zwicker, W. S. 1994. Old and new moving-knife schemes. *Economic Research Reports* # 94-30.

Brams, S. J., Taylor, A. D. and Zwicker, W. S. 1997. A moving-knife solution to the four-person envy-free cake division problem. *Proceedings of The American Mathematical Society* 125:547–554.

Brams, S. J. 1995. On envy-free cake division. *J. Comb. Theory, Ser. A* 70(1):170–173.

Chevaleyre, Y.; Endriss, U.; Estivie, S.; and Maudet, N. 2007. Reaching envy-free states in distributed negotiation settings. In Veloso, M. M., ed., *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 1239–1244.

Chevaleyre, Y.; Endriss, U.; and Maudet, N. 2017. Distributed fair allocation of indivisible goods. *Artif. Intell.* 242:1–22.

Dubois, D.; Fargier, H.; Prade, H.; and Perny, P. 2002. Qualitative decision theory: from savage's axioms to nonmonotonic reasoning. *J. ACM* 49(4):455–495.

Dubois, D.; Fargier, H.; and Perny, P. 2003. Qualitative decision theory with preference relations and comparative uncertainty: An axiomatic approach. *Artif. Intell.* 148(1-2):219–260.

Endriss, U.; Maudet, N.; Sadri, F.; and Toni, F. 2006. Negotiating socially optimal allocations of resources. *J. Artif. Intell. Res.* 25:315–348.

Moulin, H., and Thomson, W. 1997. Axiomatic analysis of resource allocation problems. In Arrow, K. J., Sen, A. and Suzumuram K., Eds., *Social Choice Re-examinated*, 101–120.

Moulin, H. 1988. *Axioms of Cooperative Decision Making*. Cambridge University Press.

Moulin, H. 2003. *Fair division and collective welfare*. MIT Press.

Pino Peréz, R., Varela Montilva J., and Camacho, F. 2016. *Resource allocation under uncertainty: Firt steps towars a qualitative approach*. In Rivas Echeverría, F. and Arciniegas Aguirre, S. Eds. *Avances y aplicaciones de sistemas inteligentes y nuevas tecnologías*. Pontifia Universidad Católica del Ecuador, sede Ibarra y Consejo de Publicaciones de la Universidad de Los Andes, 399–412.

Sandholm, T. W. 1998. *Contract types for satisficing task allocation: I Theoretical results*. 68–75.

Sandholm, T. W. 1999. *Distributed rational decision making*. 201–258.

Steinhaus, H. 1948. The problem of fair division. *Econometrica* 16:101–104.

Steinhaus, H. 1949. Sur la division pragmatique. *Econometrica* 17:315–319.

Thomson, W. 2011. Fair allocation rules. In Arrow, K. J., Sen, A. and Suzumura, K., Eds., *Handbook of Social Choice and Welfare, Volume 2*, 393–506.