The Donkey and the Monoid

Dynamic Semantics with Control Elements

Albert Visser (albert.visser@phil.uu.nl)
Department of Philosophy, Utrecht University

Abstract. Dynamic Predicate Logic (DPL) is a variant of Predicate Logic introduced by Groenendijk and Stokhof. One rationale behind the introduction of DPL is that it is closer to Natural Language than ordinary Predicate Logic in the way it treats scope.

In this paper I develop some variants of DPL that can more easily approximate Natural Language in some further aspects. Specifically I add flexibility in the treatment of polarity and and some further flexibility in the treatment of scope.

I develop a framework that is intended to encourage further experimentation with alternative variants of DPL. In this framework the new meanings are, roughly, indexed sets of old meanings. The indices can be viewed as 'files' or 'storage devices'. Each such file supports a separate 'information stream'. The interaction of the new meanings is 'programmed' with the help of certain monoids acting on the indices. The construction of the new meanings can be viewed as an application of the Grothendieck Construction to monoids.

Keywords: Dynamic Predicate Logic, Natural Language, Meaning, Information Streams, File Semantics, Polarity, Scope, Donkey, Monoid, Monoidal Processing

MSC2000 codes: 03B65, 68T50, 91F20

Table of Contents

1. Introduction

Natural Language is more flexible, truly and impressively more flexible, than any formal language cum semantics cooked up by man.[1]

Take for example the scoping mechanisms for variables. The scope of a Natural Language quantifier like *a woman* easily passes sentence boundaries, where the scope of the analogous quantifier $\exists x$ in Predicate Logic is pityfully confined to the formula containing its occurrence. Next take argument places. In Natural Language arguments seem to be arbitrarily extendable. Moreover they may travel around widely in sentences in which their predicate occurs. In Predicate Logic every predicate symbol is rigidly assigned a fixed number of arguments by the arity function. What is more, these arguments are jealously confined to occur in fixed order right behind the occurrence of their predicate symbol. Or take polarity. Natural Language seems to switch polarity effortlessly, often on such a thin basis as intonation or clues provided by the comparison the possible contents of the sentence. In Predicate Logic occurrences of subformulas have the same or different polarity in a given formula only on the basis of a count of the nesting of the antecedents of implications in which they appear.

Dynamic Predicate Logic or DPL is a variant of Predicate Logic. It was invented by Groenendijk and Stokhof. See their classic paper (Groenendijk and Stokhof, 1991). DPL was designed as a logic whose scoping mechanisms are more like the scoping mechanisms of Natural Language. What can we expect from such a formal language? The advantage of a language like DPL over Natural Language is quite simply that we know its syntax and semantics precisely and to the last detail. Translating a fragment of Natural Language to a language like DPL is, thus, a good way of specifying a semantics for this fragment. One constraint on this project is that the translations must be *compositional*. Since DPL is more like Natural Language than Predicate Logic, the translations can be more easily made compositional. Unfortunately all this doesn't mean that by specifying the DPL semantics, we have solved the riddle of Natural Language Anaphora Resolution. Some of the mechanisms of Natural Language are still quite different. Specifically Natural Language does not work with explicit variable names.[2]

[1] Perhaps this is a good point to remind the reader that many formal languages were developed precisely because this lack of flexibility was, rightly, deemed a virtue. Less syntactical flexibility means more syntactical control. The aim of Logic's founders was to develop languages that allowed us direct and easy syntactical control. A good example of a wonderful and strong control mechanism is the use of explicit variable names.

[2] To view logics like DPL as ways to specify Natural Language meanings is not the only way of looking at them. There are at least two other views. First,

In this paper I will study *three simple epicycles* to DPL. The idea of the first epicycle is as follows. We want to make the way DPL handles polarities more flexible, more like the way Natural Language handles them. To do this we will build a simple little machine: the polarity switcher, $\bowtie$ ('bowtie'). This little machine will enable us to switch polarity where and whenever we wish to do so. For example,

$$\bowtie \cdot \exists x \cdot \mathsf{farmer} \cdot \mathsf{hungry}(x) \cdot \bowtie \cdot \mathsf{eats}(x)$$

can serve as a paraphrase of *if a farmer is hungry, he eats.* Here the intuitive meaning is taken to be the success condition of the formula. We interpret the formula as follows. When we start reading, on the left hand side, we are in the default polarity: the positive one. First we meet $\bowtie$. This action switches the polarity to negative. So $\exists x \cdot \mathsf{farmer} \cdot \mathsf{hungry}(x)$ will be at a negative place. Then we have another switch, which brings us back to the positive polarity. So $\mathsf{eats}(x)$ will be positive. Finally, the definition of success will treat the negative information as the antecedent of a dynamic implication and the positive information as the consequent. Note that our switcher is radically different from negation. Here is a second example.

$$\bowtie \cdot \exists x \cdot \mathsf{dog}(x) \cdot \bowtie \cdot \mathsf{barks}(x) \cdot \bowtie \cdot \exists y \cdot \mathsf{cat}(y) \cdot \mathsf{sees}(x, y)$$

This formula can serve as a paraphrase of *a dog barks, if it sees a cat*, where we give *a dog* a universal reading.

Just as Natural Language does not have variable names, it does not have an explicit polarity switching device. Natural Language works with little words like *if*, *then*, *only*, with intonation, with higher level inference. We will not have much to say in this paper about Natural Language's true workings.

The basic idea behind our implementation is to treat negative information and positive information as two parallel information streams that are specified linearly on the syntactical level via interleaving. Every item of information is specified on the semantical level by giving its contribution to both streams. The switcher just changes the streams to which the local contributions are added. Apart from the switcher we will have a test-connective that 'takes stock of the information we have obtained in a given period of information receiving'.[3] The success of a

logics like DPL could turn out to be useful for computer applications. One could for example think of adding a variant of DPL to a programme for doing Geometrical Constructions. Introducing and labeling an arbitrary point corresponds to the DPL existential quantifier. Etcetera. Secondly, it seems to me that theories like DPL open up new avenues for Algebraic Logic. See e.g. (Hollenberg, 1997) and (Visser, 1997) for some preliminary explorations.

[3] In fact the only connectives we have in the primary system are composition and the test-connective. The switcher will be an atomic action.

formula will correspond, roughly, to the truth of the test of the formula. The test connective will cause the negative and positive streams to interact.

In the second epicycle we will add an atomic action $\triangle$ to the first epicycle. $\triangle$ will change the behaviour of the subsequent text w.r.t. scope. We will e.g. be able to implement backwards binding. For example,

$$\triangle \cdot \exists x \cdot \mathsf{dog}(x) \cdot \triangle \cdot \mathsf{sees}(x, y) \cdot \triangle \cdot \exists y \cdot \mathsf{cat}(y) \cdot \triangle$$

will be a paraphrase for *a dog sees a cat.* As we will see, the resulting theory has connections to Discourse Representation Theory (DRT).

The polarity switching machine of the first epicycle can only switch polarity in forward direction. Consider e.g. the following dialogue. (I'm not recommending this parental behaviour.)

child *Daddy, what will you give me?*

father *I will give you . . . I will give you . . . nothing.*

It seems to me that, in semantically processing the father's utterance, we shift the interpretation of *I will give you* to a negative place at the moment that we hear *nothing.* So we do not *first* determine the polarity and *then* semantically process *I will give you* immediately in the correct 'polarity file'. In short: we switch polarity retrospectively. In epicycle 3 we add a backwards acting polarity switcher to epicycle 2 that can better mimick the dramatic reversal of informational status effected by *no*, *nothing*, *nobody.*

The constructions studied in this paper can be viewed as *adding control elements* to the language. A control element is a symbol representing an instruction concerning storage of bits of meaning. Thus, control elements serve to steer semantic processing.

An example of a control element is the polarity switcher $\bowtie$ that regulates the polarity of the (interpretation of) the subsequent text. Polarity in classical Predicate Logic and DPL is regulated at the level of syntax. We can afford ourselves a more flexible (and thus *less strictly controled* syntax) by pushing the machinery that handles polarity into the semantics. Similar remarks hold for other control elements.

In some sense there is nothing new under the sun, since *variable names* can seen as control elements: they tell us where a value should be stored or where it can be found. Control elements also occur at the semantical level, in full analogy to variable names.

It would seem to me that many words in natural language, like *not* and *but* can be viewed as being (at least in part) concerned with control. If this is true, then the present work is a move in the right direction.

An important methodology of the present paper is the use of monoids to programme the interaction of meanings. In this paper we will not try to make the semantics fully monoidal. The reader is referred to (Visser and Vermeulen, 1996) for some discussion of the aims and claims of monoidal semantics.

2. What is a Dynamic Predicate Logic?

The aim of this section is to specify what we mean by a *variant of Dynamic Predicate Logic*. We will call such variants *dynamic predicate logics* or *dpl*'s. Thus, somewhat awkwardly, DPL itself will be a dpl.

All structures we will consider are *extended monoids* or briefly *e-monoids*. Specifically, languages will be *free* e-monoids. The central class of e-monoids of this paper is formed by the *dynamic relation algebras* or *dra*'s. These structures function as the analogues of Boolean algebras. Thus dra's provide 'the propositional[4] logic of dynamic logic'.[5]

In the dynamic case the relationship between propositional logic and predicate logic is tighter than classically. To obtain predicate logic from propositional logic in the dynamic world, we only need to specialize the language, we need not extend it with new operations. What this means in detail will be made clearer below. A pleasant bonus of the convenient relationship between propositional logic and predicate logic is that we can manufacture our variants of DPL by modifying the propositional part only. The methodology of this paper is, thus, to modify dra's in several ways to obtain 'variants of propositional logic'. We will not analyse what allowable variants of dynamic propositional logic are, we just state some convenient properties shared by our examples.

e-Monoids

An *extended monoid* or e-monoid of signature Λ is specified as follows. A signature Λ for an e-monoid, briefly *an e-signature*, is is a pair $\langle \mathsf{Op}, \mathsf{Ar}\rangle$. Here OP is a, possibly empty, set of function symbols and Ar is the arity function, a function from Op to the natural numbers (including 0). We will write e.g. $[F^2, G^3]$ for the e-signature $\langle\{F,G\},\{\langle F,2\rangle,\langle G,3\rangle\}\rangle$. An

[4] *Propositional* is, of course, a misleading designation here, since some of the meanings are intended to be actions and precisely not propositions. It is only used here to stress *the relative roles* of propositonal and predicate logics.

[5] Are dra's really the correct choice for this exalted role? That's is a serious worry. E.g. if one considers Dynamic Predicate Logic with partial assignments, one might wish to change the 'logic' to treat the partiality of the inputs in a plausible way. I propose to set aside this kind of problem for the moment and simply pretend that dra's are the right choice.

e-monoid of e-signature Λ is a structure $\langle \mathcal{M}, J\rangle$, where $\mathcal{M}$ is a monoid with domain M and J is a function on Λ, where $J(F) : M^{\mathsf{Ar}(F)} \to M$.

As usual we will write e.g. $\langle M, \bullet, \mathsf{id}, F, G\rangle$ instead of $\langle\langle M, \bullet, \mathsf{id}\rangle, J\rangle$ with J mapping $\{F, G\}$ to suitable functions. We will use $;, \circ, \cdot$ instead of $\bullet$ depending on the structures considered. A notational insolubile is the fact that *algebraically* speaking id or 1 is a suggestive choice of notation, but that from the standpoint of *logic* $\top$ is the better choice. $\top$ is especially unhappy since in relational algebra it stands for the universal relation, not for identity. We will always use id, *except when we are treating logical languages.*

A *language* $\mathcal{L}_P^\Lambda$ of signature Λ with 'variables' P is simply the free e-monoid of signature Λ on generators P. If we are talking about language we always use $\cdot, \top$. Note the curious 'dynamic features' of our language: brackets for $\cdot$ are automatically omitted, an expression like $\top \cdot p$ is automatically simplified to p.

Consider any e-monoid $\mathcal{E}$ of signature Λ. An *assignment* I for $\mathcal{L}_P^\Lambda$ is a function from P to M. $[\![.]\!]_I$ is the unique extension of I to a (structural) morphism of e-monoids. Thus, $[\![\phi]\!]_I$ is the interpretation of ϕ under I.

Dynamic Relation Algebras

A *full dynamic relation algebra*[6] or *full dra* on a non-empty set X is a structure $\mathcal{R}_X := \langle \mathsf{Rel}(X), ;, \mathsf{id}, \bot, \to\rangle$. Here:

1. $\mathsf{Rel}(X)$ is the set of binary relations on X, i.e., $\mathsf{Rel}(X) := \wp(X \times X)$.
2. The composition $R; S$ of R and S is defined by:

   $$x(R;S)y :\Leftrightarrow \exists z\ xRzSy.$$

   We distinguish ; from $\circ$ with $x(R \circ S)y :\Leftrightarrow \exists z\ xSzRy$.
3. id is the identity relation.
4. $\bot$ is the empty relation on X.
5. The dynamic implication $(R{\to}S)$ between R and S is defined by:

   $$x(R{\to}S)y :\Leftrightarrow x = y \text{ and } \forall z(xRz \Rightarrow \exists u\ zSu).$$

   Our use of $\to$ here overloads the symbol, since we also use it, sometimes, for implication in the object language. This notion of implication is originally due to Kamp ((Kamp, 1981)). In its present form it was introduced by Groenendijk and Stokhof ((Groenendijk and Stokhof, 1991)).

[6] See (Hollenberg, 1997) and (Visser, 1997). Our usage diverges a bit, but for our purposes inessentially, from the usage of (Hollenberg, 1997).

A dynamic relation algebra or *dra* is a substructure of a full dra. Note that the dynamic relations algebras are a class of e-monoids of e-signature $[\bot^0, \to^2]$.

There is a standard embedding diag of the Boolean Algebra

$$\mathcal{P}_X := \langle \wp X, X, \emptyset, \cap, \to \rangle$$

to $\mathcal{R}_X$. It is given by: $\mathsf{diag}(Y) := \{\langle y, y \rangle \mid y \in Y\}$. This embedding is a (structural) morphism of dra's.

We will write $\neg R$ for $(R \to \bot)$. We write $\mathsf{dom}(R)$ for the domain of R. It is easy to see that $\neg R = \mathsf{diag}(X \setminus \mathsf{dom}(R))$ and $\neg\neg R = (\mathsf{id} \to R) = \mathsf{diag}(\mathsf{dom}(R))$.

A relation R is a *test* or *condition* if $R \subseteq \mathsf{id}$. The range of diag consists precisely of the conditions.

Variants of Dynamic Relation Algebras

Our framework for variants of DPL provides a function from e-signatures and signatures for predicate logic to the corresponding language plus model theory. We introduce e-signatures here and signatures for predicate logic in the subsection below.

The dpl's or variants of DPL can be given in a way analogous to the presentation of DPL itself: the only difference is that we replace dra's as propositional logic analogue by a different class of structures based on dra's. Variants of dra's are always e-monoids. Their specification provides us with the following data.

1. An e-signature Λ. The variants are certain e-monoids of signature Λ. We assume that $\bot$ is in Λ.

2. A functor Φ from the category of dra's to the category of variants under consideration. The morphisms are here the obvious morphisms of algebras. Φ sends a dra to the variant constructed from it.

3. A term t of $\mathcal{L}^{\Lambda}_{p,q}$. The term t is intended to define $(p \to q)$ over the variant (in a restricted sense). Thus, t defines a binary function ι on a given variant $\mathcal{V} = \langle \mathcal{M}, J \rangle$. We map $\mathcal{V}$ via a mapping Ψ to an e-monoid, of signature $[\bot^0, \to^2]$, viz. $\langle \mathcal{M}, I \rangle$ with $I(\bot) = \bot$ and $I(\to) = \iota$.
(Warning: we do not demand that the resulting structure is a dra!)

4. We ask that $\mathcal{R}$, the original dra, can be naturally embedded into $\Psi(\Phi(\mathcal{R}))$ via, say, emb. So modulo definitional extension our original dra is supposed to be a substructure of its variant.

Signatures and Models for Predicate Logic

The treatment of the basics of predicate logic is completely classical. A signature Σ for predicate logic is a structure $\langle \mathsf{Pred}, \mathsf{Ar}, \mathsf{Con}, \mathsf{Var}, = \rangle$. We will call such signatures *ℓ-signatures.* Pred is a set of predicate symbols; Ar is a function from Pred to the natural numbers (including 0); Con is a set of constants; Var is a, possibly empty, set of variables. We put $\mathsf{Ref} {:=} \mathsf{Con} \cup \mathsf{Var}$. Ref is the set of *referents.* $=$ is in Pred and $\mathsf{Ar}(=) = 2$.

A model $\mathcal{M}$ of signature Σ is a tuple $\langle D, I \rangle$, where D is a non-empty domain. I is a function on $\mathsf{Pred} \cup \mathsf{Cons}$, where for $c \in \mathsf{Cons}$, $I(c) \in D$, and, for $P \in \mathsf{Pred}$, $I(P) \subseteq D^{\mathsf{Ar}(P)}$. (If $\mathsf{Ar}(P) = 0$, $I(P)$ will be either $\emptyset$ or $\{\Box\}$, where $\Box$ is the empty sequence. Here $\emptyset$ has the role of the truthvalue false and $\{\Box\}$ has the role of true.) We demand that $I(=) = \{\langle d, d \rangle \mid d \in D\}$. We define:

1. $\mathsf{Ass} := D^{\mathsf{Var}}$

2. For $f \in \mathsf{Ass}$ and $r \in \mathsf{Ref}$, $r[f] := f(r)$ if $r \in \mathsf{Var}$ and $r[f] := I(r)$ if $r \in \mathsf{Con}$.

The notion of model only depends on Σ. It is just the classic notion of model and is not specifically tied up with dynamics.

Language and Semantics

Fix an e-signature Λ and an ℓ-signature Σ. We define:

1. Atcond ($= \mathsf{Atcond}_\Sigma$) is the set of $P(r_1, \cdots, r_n)$, for $P \in \mathsf{Pred}$ with $\mathsf{Ar}(P) = n$ and $r_1, \ldots, r_n \in \mathsf{Ref}$.
   Atcond is the set of atomic conditions.

2. Reset ($= \mathsf{Reset}_\Sigma$) is the set of $\exists v$, for $v \in \mathsf{Var}$.

The DPL-language $\mathcal{L}$ of signature Λ, Σ is simply $\mathcal{L}^{\Lambda}_{\mathsf{Atcond}_\Sigma \cup \mathsf{Reset}_\Sigma}$.

A dpl is fully specified by giving a class of dra variants plus Φ, t and emb. Suppose we are given such a class of variants and a model $\mathcal{N}$. We map our generators, viz. $\mathsf{Atcond}_\Sigma \cup \mathsf{Reset}_\Sigma$, via $[\![.]\!]_0$ —to be defined below— to elements of $\Phi(\mathcal{R}_{\mathsf{Ass}})$. Here Ass is the class of assignments for $\mathcal{N}$ on the set Var provided by Σ.

– Let $\|P(r_1, \cdots, r_n)\| := \{f {\in} \mathsf{Ass} \mid \langle r_1[f], \cdots, r_n[f] \rangle \in I(P)\}$.
  Thus $\|P(r_1, \cdots, r_n)\|$ is the usual interpretation of $P(r_1, \cdots, r_n)$ in Predicate Logic. We take:

$$[\![P(r_1, \cdots, r_n)]\!]_0 := \mathsf{emb}(\mathsf{diag}(\|P(r_1, \cdots, r_n)\|)).$$

– $[\exists v]$ is the relation given by:

$$f[\exists v]g :\Leftrightarrow \forall w{\in}\mathsf{Var}\ (w \not\equiv v \Rightarrow f(w) = g(w)).$$

$[\exists v]$ is the relation *random-reset*. We take: $[\![\exists v]\!]_0 := \mathsf{emb}([\exists v])$.

We can extend $[\![.]\!]_0$ in a unique way to the full language $\mathcal{L}$. We call the resulting interpretation $[\![.]\!]$.

The set-up presented here will undoubtedly turn out both to broad and to narrow. Too broad, since we put only very light constraints on the relation between the e-monoids that our 'propositional logic variants' and the 'underlying' dra's. Too narrow since e.g. the choice of dra's is rather restrictive. We could, for example wish to add the possibility of things getting undefined to our logics.

Validity

In all our variants of dra's we can define a unary function val that sends a meaning to its set of truthmakers. E.g. in DPL, val will be dom, the domain function. We will define satisfaction in predicate logic as follows.

– $\mathcal{M}, f \models \phi :\Leftrightarrow f \in \mathsf{val}([\![\phi]\!])$

Original DPL

We obtain DPL, considered as a variant of itself, by taking Λ the signature of dra's and Φ, Ψ and emb the appropriate identity functions. We write [.] for the interpretation function of DPL. Here are some pleasant abbreviations:

– $\neg(\phi)$ for: $(\phi \to \bot)$

– $?(\phi)$ for: $(\top \to \phi)$ (or, alternatively, for $\neg(\neg(\phi))$

– $[x := c]$ for: $\exists x \cdot x = c$

– $\forall x\,(\phi)$ for: $(\exists x \to \phi)$

In subsection B.1 of appendix B, we will discuss how to translate ordinary Predicate Logic into DPL. In subsection B.2 of appendix B, we translate DPL into any dpl.

If we compare the way DPL treats scope with the way Predicate Logic treats scope, we see that DPL's way is more like the way Natural Language does it. We remind the reader of Geach's famous Donkey Sentence:

– *If a farmer owns a donkey, then he beats it.*
  This sentence can be paraphrased by:
  $(\exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \mathsf{owns}(x,y) \rightarrow \mathsf{beats}(x,y))$

Note that we employ the convention that $\cdot$ binds stronger than $\rightarrow$.

3. The Polarity Switcher

In this section we develop our first variant of DPL: we add a polarity switcher.

3.1. Definition of the Logic

We specify the construction Φ_{pol} of an e-monoid from a dynamic relation algebra. In this construction the small monoid Pol will play an important role.

Pol, the polarities monoid, has elements $+$ and $-$. We stipulate: $+ \cdot + = - \cdot - = +$ and $- \cdot + = + \cdot - = -$. Thus $+ = \mathsf{id}_{\mathsf{Pol}}$. We will often drop the "$\cdot$" and write $+-$, etc. $\alpha, \beta, \ldots$ will range over $+, -$. Pol is isomorphic with the monoid Add_2 of addition modulo 2, with $+$ in the role of 0 and $-$ in the role of 1. A second way of viewing it is as the monoid of the elements 1 and -1. under ordinary multiplication. A third way of looking at Pol is as the the monoid of relations with composition, on two elements a, b, consisting of two relations $R_+ := \mathsf{id}_{\{a,b\}}$, and $R_- := \{\langle a,b\rangle, \langle b,a\rangle\}$. This last representation is suggestive since the point of the polarities in our set-up is to switch state.

Given any dra, $\mathcal{Q} = \langle Q, \bullet, \mathsf{id}, \perp, \rightarrow\rangle$, we define an e-monoid as follows.

$$\mathcal{N} := \Phi_{\mathsf{pol}}(\mathcal{Q}) := \langle N, \bullet, \mathsf{id}, \perp, \bowtie, \mathsf{test}\rangle.$$

N consists of the triples $\langle q_-, q_+, \alpha\rangle$, where the q_β are in $\mathcal{Q}$. Define:

– $\mathsf{id} := \langle \mathsf{id}, \mathsf{id}, +\rangle$

– $\perp := \langle \mathsf{id}, \perp, +\rangle$

– $\bowtie := \langle \mathsf{id}, \mathsf{id}, -\rangle$

– $\langle q_-, q_+, \alpha\rangle \bullet \langle r_-, r_+, \beta\rangle := \langle q_- \bullet r_{-\alpha}, q_+ \bullet r_{+\alpha}, \alpha\beta\rangle$
  Equivalently,
  $\langle q_-, q_+, \alpha\rangle \bullet \langle r_-, r_+, \beta\rangle := \langle p_-, p_+, \alpha\beta\rangle$
  where $p_\gamma := q_\gamma \bullet r_{\gamma\alpha}$

– $\mathsf{test}(\langle q_-, q_+, \alpha\rangle) := \langle \mathsf{id}, (q_- \rightarrow q_+), +\rangle$

The intuition is that q_- represents the negative information, q_+ the positive information, and α represents the polarity of the *subsequent* information. α has the role of a 'dynamic context' that steers the way in which the stored information is merged with the incoming information. The test operation gives the negatively stored information scope priority over the positively stored information. This will happen even if the negative information is introduced later in the sentence, making kataphoricity possible. (This strategy —negative before positive— is undoubtedly too rigid.)

It is easily seen that $\langle N, \bullet, \mathsf{id}\rangle$ forms a monoid, as desired. (See also section 5.) Note that in dra's $\bot$ is an annihilator: $\bot \bullet q = q \bullet \bot = \bot$. However, in $\mathcal{N}$, we have $\bot \bullet \bowtie \neq \bot$. We may extend the mapping Φ_{pol} to morphisms in the obvious way.

Let u, v range over all elements of our e-monoid. We may regain implication by defining $(u \to v) := \mathsf{test}(\bowtie \bullet u \bullet \bowtie \bullet v)$. The functor Ψ_{pol} is given by $\to$ thus defined. We define emb by: $\mathsf{emb}(q) := \langle \mathsf{id}, q, +\rangle$. It is easily seen that emb has the required properties. Note that $\to$ only behaves like implication on the range of emb (or, more precisely, if its antecedent is in the range of emb). Outside that range it is not all that meaningful.

We can also embed Pol into $\mathcal{N}$ by: $\mathsf{emb}^*(\alpha) := \langle \mathsf{id}, \mathsf{id}, \alpha\rangle$. Such embeddings are, of course, fully analogous to the embedding of the natural numbers into the integers. We can see that the extension of semantical memory need not be dramatically wasteful. For whole stretches of discourse, we may accumulate positive information in the classic way. The existence of the embedding tells us that the classical way is simply there. The fact that we use triples instead of single relations is just a matter of metamathematical coding. Only when needed we activate the extra resource which is present *in potentia*. Among the integers the natural numbers also reappear as equivalence classes of pairs.

Finally we take $\mathsf{val}(\langle q_-, q_+, \alpha\rangle) := \mathsf{dom}(q_- \to q_+)$.

Here are some valid identities. Let x, y range over elements of the form $\langle \mathsf{id}, m, +\rangle$.

1. $\bowtie \bullet \bowtie = \mathsf{id}$
2. $\bowtie \bullet x \bullet \bowtie \bullet y = y \bullet \bowtie \bullet x \bullet \bowtie$
3. $\mathsf{test}(\mathsf{id}) = \mathsf{id}$, $\mathsf{test}(\bot) = \bot$
4. $\mathsf{test}(\bowtie \bullet \bot \bullet u) = \mathsf{id}$
5. $\mathsf{test}(\mathsf{test}(u)) = \mathsf{test}(u)$

The identity in 2 above means that first storing x in the negative file and then y in the positive file, has the same effect as first storing y in the positive file and then x in the negative file and then switching back to positive. The positive and negative file only interact in the tests. Thus, in the tests, backwards binding from the negative files to the positive files becomes possible. See the paraphrases below.

In section 5 we will provide a more abstract view of our construction. We call our new DPL-variant: $\mathsf{DPL}_{\mathsf{pol}}$.

REMARK 3.1. A pleasant alternative way of writing our meanings is obtained as follows. First we represent the triple $\langle q_-, q_+, \alpha\rangle$ as the function f on $\{-, +, \mathsf{pol}\}$ with $f(-) := q_-$, $f(+) := q_+$, $f(\mathsf{pol}) := \alpha$. The elements in the range are all elements of some monoid. We take the value id to be the default value. We will write e.g. $\{\!\![- : S \mid \mathsf{pol} : -]\!\!\}$ for the function g with $g(-) := S$, $g(+) := \mathsf{id}$, $g(\mathsf{pol}) := -$. Here are some specifications of meanings using the alternative notations.

- $[\![P(r_1, \cdots, r_n)]\!] := \{\!\![+ : [P(r_1, \cdots, r_n)]]\!\!\}$
- $[\![\exists v]\!] := \{\!\![+ : [\exists v]]\!\!\}$
- $[\![\top]\!] := \{\!\![\ ]\!\!\}$
- $[\![\bot]\!] = \{\!\![+ : \bot]\!\!\}$
- $[\![\bowtie]\!] = \{\!\![\mathsf{pol} : -]\!\!\}$

❑

The proper way to view $\bowtie$ is as a dynamic bracket, signalling a switch of polarity. It is somewhat analogous to the bracket \$ of LaTeX, compare $+ + \bowtie - - \bowtie + +$ with *plain \$ math \$ plain*. An important difference between $\bowtie$ and the brackets developed in (Visser and Vermeulen, 1996) is that nothing is thrown away when we switch polarity via $\bowtie$. What has been stored is remembered and can be further extended as soon as we return. In contrast a level that is popped by one of the dynamic brackets of (Visser and Vermeulen, 1996) cannot be returned to.

In appendix A, we give an alternative relational representation of the semantics of $\mathsf{DPL}_{\mathsf{pol}}$ In subsection B.3 of appendix B, we show how to 'translate' $\mathsf{DPL}_{\mathsf{pol}}$ into DPL.

3.2. A Donkey Owner's Manual

In this section we present some paraphrases of Natural Language sentences in $\mathsf{DPL}_{\mathsf{pol}}$. They are intended to illustrate both the flexibility and the rigidity of our language. To get the intuitive reading, we read our formulas ϕ as corresponding success condition i.e. as $\mathsf{val}([\![\phi]\!])$. One should, however, not forget that this fails to represent the dynamic effect of the sentence in a larger text.

1. *Only if a farmer OWNS a donkey, does he beat it.*
   $\bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \bowtie \cdot \mathsf{owns}(x,y) \cdot \bowtie \cdot \mathsf{beats}(x,y)$
   In section 6 we will introduce a method to follow the order of the original even more closely. The semantics makes our sentence equivalent to:
   If a farmer beats a donkey, he owns it.
   If we wish to get rid of the assumption of a donkey beating farmer in the subsequent discourse, we must change our paraphrase to:
   $?(\bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \bowtie \cdot \mathsf{owns}(x,y) \cdot \bowtie \cdot \mathsf{beats}(x,y))$
   We may wish to proceed with our discourse still having donkey and farmer with us, but without the assumption of a beating. E.g.:
   Only if a farmer OWNS a donkey, does he beat it.
   If he treats it well, he doesn't own it.
   We can do this as follows:
   $\bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \bowtie \cdot ?(\mathsf{owns}(x,y) \cdot \bowtie \cdot \mathsf{beats}(x,y)) \cdot$
   $?(\bowtie \cdot \mathsf{w\text{-}treats}(x,y) \cdot \bowtie \cdot \neg(\mathsf{own}(x,y)))$
   We do not really supply a semantics for *only*. It seems to me that *only* combines with the emphasis on *owns* to shift *owns* to a positive place. We describe the result but not the mechanism of this shift.

2. *Only if a FARMER owns a donkey, does he beat it.*
   $\bowtie \cdot \exists x \cdot \bowtie \cdot \mathsf{farmer}(x) \cdot \bowtie \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \mathsf{owns}(x,y) \cdot \mathsf{beats}(x,y)$
   This paraphrase gives as meaning:
   If something owns and beats a donkey, then it is a farmer

3. *A farmer beats a donkey, if he owns it.*
   $\bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \bowtie \cdot \mathsf{beats}(x,y) \cdot \bowtie \cdot \mathsf{owns}(x,y)$
   Alternatively:
   $\bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \bowtie \cdot ?(\mathsf{beats}(x,y) \cdot \bowtie \cdot \mathsf{owns}(x,y))$
   If we attempt to read the *a farmer* and the *a donkey* positively, we could get:
   $\exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \mathsf{beats}(x,y) \cdot \bowtie \cdot \mathsf{owns}(x,y)$
   Inspecting the semantics of this last example we see that the variables of $\mathsf{owns}(x,y)$ are not bound. An alternative paraphrase does work:
   $\exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot ?(\mathsf{beats}(x,y) \cdot \bowtie \cdot \mathsf{owns}(x,y))$
   Section 6 will provide a nicer alternative.

4. *He beats it, if a farmer owns a donkey.*
   $\mathsf{beats}(x,y) \cdot \bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \mathsf{owns}(x,y)$
   Deviant? We can do it anyway. However, this is not of much help, since we cannot do:
   He beats it, if a farmer owns a donkey.

He treats it well, if he doesn't own it.
We will show how to do this text in section 6.

5. We cannot do:
If he owns it, a farmer beats a donkey.
Here e.g. *a farmer* will not succeed in binding *he*, even if we put *a farmer* on a negative place. Section 6 will provide a solution.

6. Not English, but we *do* understand it:
If he owns it, he beats it. A farmer, a donkey.
$?(\bowtie \cdot \mathsf{owns}(x,y) \cdot \bowtie \cdot \mathsf{beats}(x,y)) \cdot \bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y)$
We cannot do it when we insist on the positive reading for *a farmer, a donkey*. The method of section 6 will provide also a reasonable paraphrase for the positive reading.

7. *He was quite angry. John. He was MAD.*
We can do this, if we are prepared to allow a little trick:
$\mathsf{q\text{-}angry}(x) \cdot \bowtie \cdot \exists x \cdot \mathsf{john} = x \cdot \bowtie \cdot \mathsf{MAD}(x)$
It is undeserved to shift *John* here to a negative place. Section 6 will allow us to do the trick positively.

8. *A man keeps his word. He is honest.*
$\bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \bowtie \cdot \mathsf{keepword}(x) \cdot \mathsf{honest}(x)$

9. *A dog barks. If it is beaten, it whines.*
$\bowtie \cdot \exists x \cdot \mathsf{dog}(x) \cdot \bowtie \cdot \mathsf{barks}(x) \cdot ?(\bowtie \cdot \mathsf{beaten}(x) \cdot \bowtie \cdot \mathsf{whines}(x))$
Note that the following paraphrase gets it wrong by also making the barking conditional on the beating:
$\bowtie \cdot \exists x \cdot \mathsf{dog}(x) \cdot \bowtie \cdot \mathsf{barks}(x) \cdot \bowtie \cdot \mathsf{beaten}(x) \cdot \bowtie \cdot \mathsf{whines}(x)$

10. *A dog barks. It whines, if beaten.*
$\bowtie \cdot \exists x \cdot \mathsf{dog}(x) \cdot \bowtie \cdot \mathsf{barks}(x) \cdot ?(\mathsf{whines}(x) \cdot \bowtie \cdot \mathsf{beaten}(x))$

11. *John sees nobody*
$\perp \cdot \bowtie \cdot \exists y \cdot \mathsf{person}(y) \cdot \mathsf{sees}(j,y)$
Note that we are forced to shift the *nobody* to the beginning of the sentence. In section 7 we will explore a way to be more faithful to the original order. The paraphrase of *nobody* as $\perp \cdot \bowtie \cdot \exists y \cdot \mathsf{person}(y)$ makes *nobody* into a mix of quantifier and control element. This interpretation diverges markedly from what is usual in the literature.

12. *No man sees a woman*
$\perp \cdot \bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \exists y \cdot \mathsf{woman}(y) \cdot \mathsf{sees}(x,y)$
Here *no* is paraphrased by $\perp \cdot \bowtie \cdot \exists x$. Thus, like *nobody*, *no* is partly 'about' control.

13. *A man comes in. He sees no woman*
$\exists x \cdot \mathsf{man}(x) \cdot \mathsf{comes\text{-}in}(x) \cdot ?(\bot \cdot \bowtie \cdot \exists y \cdot \mathsf{woman}(y) \cdot \mathsf{sees}(x,y))$

4. The Polarity Switcher meets Disjunction

In this section we extend $\mathsf{DPL_{pol}}$ with disjunction. We call the resulting theory cum semantics $\mathsf{DPL_{pol}}(\vee)$.

4.1. Definition of the Logic

We extend the e-signature of $\mathsf{DPL_{pol}}$ with a binary operation symbol $\vee$. We take as interpretation of $\vee$ the operation $\sqcup$, which is defined as follows:

– $\langle q_-, q_+, \alpha\rangle \sqcup \langle r_-, r_+, \beta\rangle := (\langle q_-; r_-, q_+ \cup r_+, +\rangle)$

Note that $\sqcup$ doesn't use the polarities α and β. Here is a heuristics for the definition. $\langle q_-, q_+, \alpha\rangle$ is a bit like $(q_0 \to q_1)$ in ordinary propositional logic. Similarly $\langle r_-, r_+, \alpha\rangle$ is like $(r_0 \to r_1)$. Now $\langle q_-, q_+, \alpha\rangle \sqcup \langle r_-, r_+, \beta\rangle$ will have to be like $((q_0 \to q_1) \vee (r_0 \to r_1))$, which is equivalent in ordinary propositional logic to $((q_0 \wedge r_0) \to (q_1 \vee r_1))$. Finally, $((q_0 \wedge r_0) \to (q_1 \vee r_1))$ is like $\langle q_-; r_-, q_+ \cup r_+, +\rangle$. The really important step is here the move from the static $\wedge$ in the antecedent to the dynamic ; in the negative file.

Modulo the addition of $\vee$, our logic as specified like $\mathsf{DPL_{pol}}$. We will abbreviate $?((\phi \vee \psi))$ by $?(\phi \vee \psi)$. We take $\cdot$ to bind stronger than $\vee$.

Simple union of relations as employed in the second component of $\sqcup$ is not necessarily the correct way to model disjunction or choice. E.g. one might think that too much of the identity of the relations that are thrown together is lost. This makes it hard, for example, to paraphrase examples in which each disjunct is picked up anaphorically in later discourse. I propose to ignore these worries in the present paper.

4.2. Paraphrases

Here are some paraphrases in $\mathsf{DPL_{pol}}(\vee)$. Remember that the intuitive readings are supposed to corespond to the success conditions.

1. *This house has no bathroom or it is very small.*
$\bot \cdot \bowtie \cdot \exists x \cdot \mathsf{bathroom}(x) \cdot \mathsf{has}(h,x) \vee \mathsf{verysmall}(x)$
This gives us the same meaning as:
If this house has a bathroom, then it is very small.

2. *A man drinks nothing or it is beer.*
   $\bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \bowtie \cdot \bot \cdot \bowtie \cdot \exists y \cdot \mathsf{drinks}(x,y) \vee \mathsf{beer}(y)$
   The translation of *nothing* is $\bot \cdot \bowtie \cdot \exists y$

3. *A man drinks a beer or he doesn't drink anything.*
   $\bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \bowtie \cdot \exists y \cdot \mathsf{beer}(y) \cdot \mathsf{drinks}(x,y) \vee \bot \cdot \bowtie \cdot \exists z \cdot \mathsf{drinks}(x,z)$
   This gives us the same meaning as:
   If a man drinks something, he drinks a beer.
   Note that the following paraphrase also works:
   $\bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \bowtie \cdot \exists y \cdot \mathsf{beer}(y) \cdot \mathsf{drinks}(x,y) \vee ?(\bot \cdot \bowtie \cdot \exists z \cdot \mathsf{drinks}(x,z))$

4. *A man drinks it or it is not a beer*
   $\bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \bowtie \cdot \mathsf{drinks}(x,y) \vee \bot \cdot \bowtie \cdot \exists y \cdot \mathsf{beer}(y)$
   Dear reader, judge the plausibility for yourself. We get as meaning, roughly, the meaning of:
   Every man drinks every beer.

5. *A dog barks or it whines. It is nuisance.*
   $(\bowtie \cdot \exists x \cdot \mathsf{dog}(x) \cdot \bowtie \cdot \mathsf{barks}(x) \vee \mathsf{whines}(x)) \cdot \mathsf{nuisance}(x)$
   Alternatively:
   $\bowtie \cdot \exists x \cdot \mathsf{dog}(x) \cdot \bowtie \cdot (\mathsf{barks}(x) \vee \mathsf{whines}(x)) \cdot \mathsf{nuisance}(x)$

6. *A man drinks a beer or a coke. He finds it refreshing.*
   $\bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \bowtie \cdot (\exists y \cdot \mathsf{beer}(y) \vee \exists y \cdot \mathsf{coke}(y)) \cdot \mathsf{drinks}(x,y) \cdot \mathsf{finref}(x,y)$
   Alternatively:
   $\bowtie \cdot \exists x \cdot \mathsf{man}(x) \cdot \bowtie \cdot \exists y \cdot (\mathsf{beer}(y) \vee \mathsf{coke}(y)) \cdot \mathsf{drinks}(x,y) \cdot \mathsf{finref}(x,y)$

5. Constructing certain e-Monoids

In this section we generalize the construction Φ_{pol}. As will pointed out in appendix C the construction is nothing but a special case of the well known Grothendieck Construction. In the present paper we will only use an embarrassingly small part of the construction. My justification for including it, is twofold. First I think it really becomes more clear what is going on by looking at the more general construction. Secondly, I hope to encourage the reader to vary the construction to produce her own variants of DPL. We restrict ourselves to the monoidal part. We first give a general construction, make it more specific and, then, make it more specific a second time to arrive at (the monoidal part of) Φ_{pol}.

Let I be any non empty set. Let $\mathcal{A} := \langle A, \cdot, \mathsf{id} \rangle$ be a monoid. $\mathcal{A}$ will be our generalization of Pol. A *right action* of $\mathcal{A}$ on I is a function $\mu : (I \times A) \to I$ with the following properties. Writing $i \cdot a$ for $\mu(i,a)$:

– $i \cdot \mathsf{id} = i$

– $i \cdot (a_1 \cdot a_2) = (i \cdot a_1) \cdot a_2$

A left action is similarly defined.

We extend the notion of an action of a monoid on a set to the notion of an action of a monoid on a monoid in te following way. Let $\mathcal{A} := \langle A, \cdot, \mathsf{id}\rangle$ and $\mathcal{B} := \langle B, \bullet, \mathsf{id}\rangle$ be monoids. $\mu : (B \times A) \to B$ is a right action of $\mathcal{A}$ on $\mathcal{B}$ if μ is a right action of $\mathcal{A}$ on B and the mapping $\mu_a : b \mapsto \mu(b, a)$ is a morphism of monoids, i.o.w.

– $\mathsf{id}_{\mathcal{B}} \cdot a = \mathsf{id}_{\mathcal{B}}$

– $(b_1 \bullet b_2) \cdot a = (b_1 \cdot a) \bullet (b_2 \cdot a)$

Left actions are defined in the obvious way. Let ν be a left action of $\mathcal{A}$ on $\mathcal{B}$. Define $\sum_{\mathcal{A}} \nu := \langle A \times B, \bullet, \mathsf{id}\rangle$. Here:

– $\langle a_1, b_1\rangle \bullet \langle a_2, b_2\rangle := \langle a_1 \cdot a_2, b_1 \bullet (a_1 \cdot b_2)\rangle$

– $\mathsf{id}_{\sum_{\mathcal{A}} \nu} := \langle \mathsf{id}_{\mathcal{A}}, \mathsf{id}_{\mathcal{B}}\rangle$

We can show that $\sum_{\mathcal{A}} \nu$ is a monoid. E.g.:

$$\begin{aligned}
(\langle a_1, b_1\rangle \bullet \langle a_2, b_2\rangle) \bullet \langle a_3, b_3\rangle &= \langle a_1 \cdot a_2, b_1 \bullet (a_1 \cdot b_2)\rangle \bullet \langle a_3, b_3\rangle \\
&= \langle a_1 \cdot a_2 \cdot a_3, b_1 \bullet (a_1 \cdot b_2) \bullet (a_1 \cdot a_2 \cdot b_3)\rangle \\
&= \langle a_1 \cdot a_2 \cdot a_3, b_1 \bullet (a_1 \cdot (b_2 \bullet (a_2 \cdot b_3)))\rangle \\
&= \langle a_1, b_1\rangle \bullet \langle a_2 \cdot a_3, b_2 \bullet (a_2 \cdot b_3)\rangle \\
&= \langle a_1, b_1\rangle \bullet (\langle a_2, b_2\rangle \bullet \langle a_3, b_3\rangle)
\end{aligned}$$

In appendix C we indicate how to view the present construction as a special case of the well known Grothendieck construction. We make our construction more specific. We replace $\mathcal{B}$ in our construction by a more specific monoid and we replace ν by a more specific left action. Let I be any non-empty set. I is the set of indices, files, streams, states Let ρ be a right action of $\mathcal{A}$ on I. Let $\mathcal{D}$ be any monoid. $\mathcal{D}$ is our generalization of the algebra of relations under composition. We consider D^I. Define:

– $\mathsf{id} : I \to D$ is given by: $\mathsf{id}(i) := \mathsf{id}_{\mathcal{D}}$

– For $f, g : I \to D$, $(f \bullet g)(i) := f(i) \bullet g(i)$

It is easy to see that $\mathcal{D}^I := \langle D^I, \bullet, \mathsf{id}\rangle$ is a monoid. We give $\mathcal{D}^I$ the role of $\mathcal{B}$ above. We define a left action $\lambda := \lambda_{\rho,\mathcal{D}}$ of $\mathcal{A}$ on $\mathcal{D}^I$, as follows.

– For $a \in A$ and $f : I \to D$, $(a \cdot f)(i) := f(i \cdot a)$

It is easy to check that λ is indeed a left action. We have e.g.:

$$(a_1 \cdot (a_2 \cdot f))(i) = (a_2 \cdot f)(i \cdot a_1) = f(i \cdot a_1 \cdot a_2) = (a_1 \cdot a_2) \cdot f(i),$$

so $a_1 \cdot (a_2 \cdot f) = (a_1 \cdot a_2) \cdot f$. Define $\Phi_\rho \mathcal{D} := \sum_{\mathcal{A}} \lambda_{\rho,\mathcal{D}}$.

Clearly a monoid $\mathcal{A}$ defines a right action $\rho_{\mathcal{A}}$ on A. We will notationally confuse $\mathcal{A}$ and $\rho_{\mathcal{A}}$. If we take $\mathcal{A} := \mathsf{Pol}$ and $\mathcal{D}$ the monoidal part of a given dra, the we find that our original $\Phi_{\mathsf{pol}}(\mathcal{D})$ is isomorphic to $\Phi_{\mathsf{Pol}}(\mathcal{D})$. An element $\langle r, s, \alpha\rangle$ of $\Phi_{\mathsf{pol}}(\mathcal{D})$ will correspond to an element $\langle \alpha, \{\langle -, r\rangle, \langle +, s\rangle\}\rangle$ of $\Phi_{\mathsf{Pol}}(\mathcal{D})$.

6. A Scoping Device

In this section we extend $\mathsf{DPL}_{\mathsf{pol}}$ with an extra scoping device. The logic so obtained will be called $\mathsf{DPL}_{\mathsf{pol,sco}}$. We will introduce two 'scope files' or 'scope streams'.

6.1. Definition of the Logic

Posco is defined as $\mathsf{Pol} \times \mathsf{Add}_2$. Here Add_2 is the monoid of addition modulo 2. So Posco is, modulo isomorphism, just the additive part of $\mathbb{Z}_2 \times \mathbb{Z}_2$.

The idea is that the things stored at the files labeled $\langle \alpha, 1\rangle$ will have scope priority over the things stored at $\langle \alpha, 0\rangle$. Since what is stored negatively takes, in test_1, precedence in scope over what is stored positively, the 'scope ordering' will be $\langle -, 1\rangle$, $\langle -, 0\rangle$, $\langle +, 1\rangle$, $\langle +, 0\rangle$. This is, of course, still a fairly rigid strategy of managing scope. We hope that the present example will encourage people to explore more flexible strategies.

Given any dra, $\mathcal{Q} = \langle Q, \bullet, \mathsf{id}, \bot, \rightarrow\rangle$, we define a new algebra as follows.

$$\mathcal{N} := \Phi_{\mathsf{pol,sco}}(\mathcal{M}) := \langle N, \bullet, \mathsf{id}, \bot, \bowtie, \triangle, \mathsf{test}_0, \mathsf{test}_1\rangle.$$

The monoidal part of $\mathcal{N}$ is simply $\langle N, \bullet, \mathsf{id}\rangle := \Phi_{\mathsf{Posco}}(\mathcal{M})$, as defined in section 5. The full definition is as follows.

- $N := Q^4 \times \{+, -\} \times \{0, 1\}$.
  We write an element of N as $\langle q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha, i\rangle$.
- $\mathsf{id} := \langle \mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{id}, +, 0\rangle$
- $\bot := \langle \mathsf{id}, \mathsf{id}, \mathsf{id}, \bot, +, 0\rangle$
- $\bowtie := \langle \mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{id}, -, 0\rangle$

- $\triangle := \langle \mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{id}, +, 1\rangle$

- $\langle q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha, i\rangle \bullet \langle r_{-,1}, r_{-,0}, r_{+,1}, r_{+,0}, \beta, j\rangle :=$
  $\langle q_{-,1} \bullet r_{-\alpha,1+i}, q_{-,0} \bullet r_{-\alpha,i}, q_{+,1} \bullet r_{\alpha,1+i}, q_{+,0} \bullet r_{\alpha,i}, \alpha.\beta, i+j\rangle$
  Written down in a more perspicuous way,
  $\langle q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha, i\rangle \bullet \langle r_{-,1}, r_{-,0}, r_{+,1}, r_{+,0}, \beta, j\rangle :=$
  $\langle p_{-,1}, p_{-,0}, p_{+,1}, p_{+,0}, \alpha.\beta, i+j\rangle$
  where $p_{\gamma,k} := q_{\gamma,k} \bullet r_{\gamma.\alpha,k+i}$

- $\mathsf{test}_0(\langle q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha, i\rangle) :=$
  $\langle q_{-,1}, \mathsf{id}, q_{+,1}, (q_{-,0} \to q_{+,0}), +, 0\rangle$

- $\mathsf{test}_1(\langle q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha, i\rangle) :=$
  $\langle \mathsf{id}, \mathsf{id}, \mathsf{id}, ((q_{-,1} \bullet q_{-,0}) \to (q_{+,1} \bullet q_{+,0})), +, 0\rangle$

The test_1 operation is our overall evaluation operation. In many cases we want to extend the scopes of actions with 'strong scope' beyond the sentence, but close the scope of the actions with 'weak scope' in the sentence. for this purpose the test_0 is useful. See the paraphrases below. We will use $?_i$ as the conective in the language corresponding to test_i.

We can easily extend $\Phi_{\mathsf{pol},\mathsf{sco}}$ to morphisms. To finish our description of the logic, define:

- $\mathsf{emb}_{\mathcal{Q}}(q) := \langle \mathsf{id}, \mathsf{id}, \mathsf{id}, q, +, 0\rangle$

- $\mathsf{val}_{\mathcal{Q}}(\langle q_{-,1}, q_{-,0}, q_{+,0}, q_{+,1}, \alpha, i\rangle) = \mathsf{dom}((q_{-,1}; q_{-,0}) \to (q_{+,1}; q_{+,0}))$

Thus, the satisfaction relation will be:

- $\mathcal{M}, f \models \phi :\Leftrightarrow \forall g\ (\, f(\llbracket\phi\rrbracket_{-,1}; \llbracket\phi\rrbracket_{-,0})g \Rightarrow \exists h\ \ g(\llbracket\phi\rrbracket_{+,1}; \llbracket\phi\rrbracket_{+,0})h\,)$

Note that $\mathcal{M}, f \models \phi \Leftrightarrow \mathcal{M}, f \models ?_1(\phi)$.

REMARK 6.1. As in remark 3.1 we may represent the elements of $\mathcal{N}$ as functions on $I := (\{+,-\} \times \{0,1\}) \cup \{\mathsf{pol}, \mathsf{sco}\}$. Under the conventions of remark 3.1, we can rewrite some of our definitions as follows.

- $\mathsf{id} := \llbracket\ \ \rrbracket$
- $\bot := \llbracket \langle +,0\rangle : \bot \rrbracket$
- $\bowtie := \llbracket \mathsf{pol} : - \rrbracket$
- $\triangle := \llbracket \mathsf{sco} : 1 \rrbracket$
- $\mathsf{test}_0(q) := \llbracket \langle -,1\rangle : q_{-,1} \mid \langle +,1\rangle : q_{+,1} \mid \langle +,0\rangle : (q_{-,0} \to q_{+,0}) \rrbracket$

– $\mathsf{test}_1(q) := [\![\langle +, 0\rangle : ((q_{-,1} \bullet q_{-,0}) \rightarrow (q_{+,1} \bullet q_{+,0}))]\!]$

❑

We have developed a logic with two scope files. Perhaps, it is more plausible to have rather an infinity of such files/streams, say structured as $\mathbb{Z}$ with sucessor and predecessor acting on them. It does not seem difficult to build a logic according to this alternative plan.

In subsection B.4 we show how to translate $\mathsf{DPL}_{\mathsf{pol,sco}}$ into DPL. In subsection B.5 we discuss the translation of Discourse Representation Theory (DRT) into $\mathsf{DPL}_{\mathsf{pol,sco}}$ and into DPL. That discussion indicates that $\mathsf{DPL}_{\mathsf{pol,sco}}$ can be viewed as a kind of generalization of both DPL and DRT.

6.2. Paraphrases

We read our formulas as the corresponding success conditions, i.e. as the truth of their test_1's.

1. *A farmer owns a Donkey. He beats it.*
   $?_0(\triangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \triangle \cdot \mathsf{owns}(x, y) \cdot \triangle \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \triangle \cdot \mathsf{beats}(x, y))$
   I added the test_0 in the paraphrase, just for reasons of systematicity of paraphrasing. Other examples suggest that it is natural to close every sentence with a 0-test.

   The example illustrates that we are able to keep closer to the natural order of discourse using our scoping device. However, the same effect can be obtained more convincingly by treating argument places like variables. A rough approximation of the paraphrase one obtains using this alternative strategy is as follows.
   $\exists \mathsf{sub} \cdot \exists \mathsf{ob} \cdot \exists \mathsf{val} \cdot \mathsf{sub} = \mathsf{val} \cdot \exists x \cdot x = \mathsf{val} \cdot \mathsf{farmer}(\mathsf{val}) \cdot \mathsf{val}\,\mathsf{E} \cdot \mathsf{owns}(\mathsf{sub}, \mathsf{ob}) \cdot$
   $\exists \mathsf{val} \cdot \mathsf{ob} = \mathsf{val} \cdot \exists y \cdot y = \mathsf{val} \cdot \mathsf{donkey}(\mathsf{val}) \cdot \mathsf{val}\,\mathsf{E} \cdot \mathsf{sub}\,\mathsf{E} \cdot \mathsf{ob}\,\mathsf{E} \cdot$
   $\exists \mathsf{sub} \cdot \exists \mathsf{ob} \cdot \exists \mathsf{val} \cdot \mathsf{sub} = \mathsf{val} \cdot x = \mathsf{val} \cdot \mathsf{val}\,\mathsf{E} \cdot \mathsf{beats}(\mathsf{sub}, \mathsf{val}) \cdot$
   $\exists \mathsf{val} \cdot \mathsf{ob} = \mathsf{val} \cdot y = \mathsf{val} \cdot \mathsf{val}\,\mathsf{E} \cdot \mathsf{sub}\,\mathsf{E} \cdot \mathsf{ob}\,\mathsf{E}$
   Here E is the exit operator, which throws away a declared variable. (Of course, more should be said about the semantics of the exit operator. See e.g. (Visser and Vermeulen, 1996).) We can make the paraphrase more readable by introducing the following abreviations: $[_v$ for $\exists v$; $\langle$ for $[_{\mathsf{sub}} \cdot [_{\mathsf{ob}}$; $($ for $[_{\mathsf{val}}$; a for $a = \mathsf{val}$; donkey for $\mathsf{donkey}(\mathsf{val})$, etc. Here is the improved paraphrase.
   $\langle \cdot (\cdot \mathsf{sub} \cdot [_x \cdot x \cdot \mathsf{farmer} \cdot) \cdot \mathsf{owns} \cdot (\cdot \mathsf{ob} \cdot [_y \cdot y \cdot \mathsf{donkey} \cdot) \cdot \rangle \cdot$
   $\langle \cdot (\cdot \mathsf{sub} \cdot x \cdot) \cdot \mathsf{beats} \cdot (\cdot \mathsf{ob} \cdot y \cdot) \cdot \rangle$
   It would be interesting to see how the dynamic treatment of argument places combines with the machinery of the present paper.

2. *Only if a farmer OWNS a donkey, does he beat it.*
   If he treats it well, he doesn't own it.
   $?_0(\bowtie \cdot \triangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \triangle \cdot \bowtie \cdot \mathsf{owns}(x,y) \cdot \bowtie \cdot \triangle \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \triangle \cdot \mathsf{beats}(x,y)) \cdot$
   $?_0(\bowtie \cdot \mathsf{w\text{-}treats}(x,y) \cdot \bowtie \cdot ?_0(\bot \cdot \bowtie \cdot \mathsf{own}(x,y)))$
   Note that the $\triangle$'s allow us to 'export' the farmer and his donkey out of the 0-tests.

3. *Only if a FARMER owns a donkey, does he beat it.*
   $?_0(\bowtie \cdot \triangle \cdot \exists x \cdot \bowtie \cdot \mathsf{farmer}(x) \cdot \bowtie \cdot \triangle \cdot \mathsf{owns}(x,y) \cdot \triangle \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \triangle \cdot \mathsf{beats}(x,y))$

4. *A farmer beats a donkey, if he owns it.*
   $?_0(\triangle \cdot \bowtie \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \bowtie \cdot \triangle \cdot \mathsf{beats}(x,y) \cdot \triangle \cdot \bowtie \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \bowtie \cdot \triangle \cdot \bowtie \cdot \mathsf{owns}(x,y))$
   A positive reading is also possible:
   $?_0(\triangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \triangle \cdot \mathsf{beats}(x,y) \cdot \triangle \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \triangle \cdot \bowtie \cdot \mathsf{owns}(x,y))$
   Note that the use of $?_1$ instead of $?_0$ would get it wrong.

5. *He beats it, if a farmer owns a donkey.*
   He treats it well, if he doesn't own it.
   $?_0(\mathsf{beats}(x,y) \cdot \bowtie \cdot \triangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \triangle \cdot \mathsf{owns}(x,y) \cdot \triangle \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \triangle) \cdot$
   $?_0(\mathsf{w\text{-}treats}(x,y) \cdot \bowtie \cdot ?_0(\bot \cdot \bowtie \cdot \mathsf{own}(x,y)))$

6. *If he owns it, a farmer beats a donkey.*
   We give the universal reading:
   $?_0(\bowtie \cdot \mathsf{owns}(x,y) \cdot \triangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \triangle \cdot \bowtie \cdot \mathsf{beats}(x,y) \cdot \bowtie \cdot \triangle \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \triangle)$

7. *If he owns it, he beats it. A farmer, a donkey.*
   We give the existential reading:
   $?_0(\bowtie \cdot \mathsf{owns}(x,y) \cdot \bowtie \cdot \mathsf{beats}(x,y)) \cdot ?_0(\triangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \triangle)$

8. *He was quite angry. John. He was MAD.*
   $?_0(\mathsf{q\text{-}angry}(x)) \cdot ?_0(\triangle \cdot \exists x \cdot \mathsf{john} = x \cdot \triangle) \cdot ?_0(\mathsf{MAD}(x))$
   The $?_0$'s do no real work. I only supplied them, to suggest a certain systematic way of translating.

9. *If he would have tried, a pilot would have hit a mig*
   that chased him. He was too hesitant.
   $?_0(\bowtie \cdot \mathsf{wht}(x) \cdot \bowtie \cdot \triangle \cdot \exists x \cdot \mathsf{pilot}(x) \cdot \triangle \cdot \mathsf{whh}(x,y) \cdot \triangle \cdot \exists y \cdot \mathsf{mig}(y) \cdot \mathsf{chased}(x,y)) \cdot$
   $?_0(\mathsf{toohes}(x))$

10. *A man saw no one on the stairs.*
    $?_0(\triangle \cdot \exists x \cdot \mathsf{man}(x) \cdot \triangle \cdot ?_1(\bowtie \cdot \mathsf{saw}(x,y) \cdot \bowtie \cdot \bot \cdot \bowtie \cdot \triangle \cdot \exists y \cdot \mathsf{person}(y) \cdot \mathsf{on\text{-}stairs}(y)))$

The necessary use of $?_1$ strikes me as somewhat ad hoc. Note that, in accordance with my intuitions, we cannot meaningfully paraphrase: *A man saw no one on the stairs. He was afraid of her.*

11. *If a woman is American, she loves Bill.*
If she is Dutch, she loves Wim.
$?_0(\bowtie \cdot \vartriangle \cdot \exists x \cdot \mathsf{woman}(x) \cdot \vartriangle \cdot \mathsf{American}(x) \cdot \bowtie \cdot \mathsf{loves}(x, b)) \cdot$
$?_0(\bowtie \cdot \mathsf{Dutch}(x) \cdot \bowtie \cdot \mathsf{loves}(x, w))$

12. *If a farmer owns a donkey, then he owns a horse.*
If he doesn't own it, then he owns a cow.
$?_0(\bowtie \cdot \vartriangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \vartriangle \cdot \mathsf{owns}(x, y) \cdot \vartriangle \cdot \exists y \cdot \mathsf{donkey}(y) \cdot \vartriangle \cdot \bowtie \cdot$
$?_1(\mathsf{owns}(x, z) \cdot \vartriangle \cdot \exists z \cdot \mathsf{horse}(z))) \cdot$
$?_0(\bowtie \cdot ?_0(\bot \cdot \bowtie \cdot \mathsf{own}(x, y)) \cdot \bowtie \cdot ?_1(\mathsf{owns}(x, u) \cdot \vartriangle \cdot \exists u \cdot \mathsf{cow}(u)))$
Note that we have to use $?_1$ to make the ownership of the horse dependent upon the ownership of the donkey and to make the ownership of the cow dependent on the non-ownership of the donkey.

13. *If a farmer is rich then he loves a donkey.*
If he is poor, he hates it.
$?_0(\bowtie \cdot \vartriangle \cdot \exists x \cdot \mathsf{farmer}(x) \cdot \vartriangle \cdot \mathsf{rich}(x) \cdot \bowtie \cdot \mathsf{loves}(x, y) \cdot \vartriangle \cdot \exists y \cdot$
$\mathsf{donkey}(y)) \cdot ?_0(\bowtie \cdot \mathsf{poor}(x) \cdot \bowtie \cdot \mathsf{hates}(x, y))$
Note that here we cannot have $?_1$ in the beginning of the consequent of the first implication. We need to have the donkey independent of the farmer's prosperity. Of course, we could also give *a donkey* a negative reading here.

7. Changing File Status Retrospectively

We introduce a logic $\mathsf{DPL}^{\mathsf{pol}}_{\mathsf{pol,sco}}$ in which we can change polarity in backwards direction. We only explore the idea of minimum interaction between the forwards and the backwards polarity changes. It should be stressed that the material below has only the status of a first experiment to see how such a construction could work.

7.1. Definition of the Logic

Given any dra, $\mathcal{Q} = \langle Q, \bullet, \mathsf{id}, \bot, \rightarrow \rangle$, we define a new algebra as follows.

$$\mathcal{N} := \Phi^{\mathsf{pol}}_{\mathsf{pol,sco}}(\mathcal{Q}) := \langle N, \bullet, \mathsf{id}, \bot, \triangleleft, \triangleright, \vartriangle, \mathsf{test}_0, \mathsf{test}_1 \rangle.$$

Here:

- $N := \{+,-\} \times Q^4 \times \{+,-\} \times \{0,1\}$.
  We write an element of N as $\langle \alpha_{\mathsf{b}}, q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha_{\mathsf{f}}, i\rangle$.
- $\mathsf{id} := \langle +, \mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{id}, +, 0\rangle$
- $\bot := \langle +, \mathsf{id}, \mathsf{id}, \mathsf{id}, \bot, +, 0\rangle$
- $\lhd := \langle -, \mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{id}, +, 0\rangle$
- $\rhd := \langle +, \mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{id}, -, 0\rangle$
- $\vartriangle := \langle +, \mathsf{id}, \mathsf{id}, \mathsf{id}, \mathsf{id}, +, 1\rangle$
- $\langle \alpha_{\mathsf{b}}, q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha_{\mathsf{f}}, i\rangle \bullet \langle \beta_{\mathsf{b}}, r_{-,1}, r_{-,0}, r_{+,1}, r_{+,0}, \beta_{\mathsf{f}}, j\rangle :=$
  $\langle \alpha_{\mathsf{b}}.\beta_{\mathsf{b}}, p_{-,1}, p_{-,0}, p_{+,1}, p_{+,0}, \alpha_{\mathsf{f}}.\beta_{\mathsf{f}}, i+j\rangle$.
  Here $p_{\gamma,k} := q_{\beta_{\mathsf{b}}.\gamma,k} \bullet r_{\gamma.\alpha_{\mathsf{f}},k+i}$
- $\mathsf{test}_0(\langle \alpha_{\mathsf{b}}, q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha_{\mathsf{f}}, i\rangle) :=$
  $\langle +, q_{-,1}, \mathsf{id}, q_{+,1}, (q_{-,0} \to q_{+,0}), +, 0\rangle$
- $\mathsf{test}_1(\alpha_{\mathsf{b}}, \langle q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha_{\mathsf{f}}, i\rangle) :=$
  $\langle +, \mathsf{id}, \mathsf{id}, \mathsf{id}, ((q_{-,1} \bullet q_{-,0}) \to (q_{+,1} \bullet q_{+,0})), +, 0\rangle$

In the evident way we can extend Φ to morphisms. Note that $\rhd$ is just our old friend $\bowtie$ in new clothes.

REMARK 7.1. As in remark 3.1 we may represent the elements of $\mathcal{N}$ as functions on $I := (\{+,-\} \times \{0,1\}) \cup \{\mathsf{polb}, \mathsf{polf}, \mathsf{sco}\}$. Under the conventions of remark 3.1, we can rewrite some of our definitions as follows.

- $\mathsf{id} := [\![\ ]\!]$
- $\bot := [\![\langle +, 0\rangle : \bot]\!]$
- $\lhd := [\![\mathsf{polb} : -]\!]$
- $\rhd := [\![\mathsf{polf} : -]\!]$
- $\vartriangle := [\![\mathsf{sco} : 1]\!]$
- $\mathsf{test}_0(q) := [\![\langle -, 1\rangle : q_{-,1} \mid \langle +, 1\rangle : q_{+,1} \mid \langle +, 0\rangle : (q_{-,0} \to q_{+,0})]\!]$
- $\mathsf{test}_1(q) := [\![\langle +, 0\rangle : ((q_{-,1} \bullet q_{-,0}) \to (q_{+,1} \bullet q_{+,0}))]\!]$

❏

Finally we specify emb and val.

– $\mathsf{emb}_{\mathcal{Q}}(q) = \langle +, \mathsf{id}, \mathsf{id}, \mathsf{id}, q, +, 0\rangle$

– $\mathsf{val}_{\mathcal{Q}}(\langle \alpha_{\mathsf{b}}, q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, \alpha_{\mathsf{f}}, i\rangle) =$
$\mathsf{dom}((q_{-,1}; q_{-,0}) \to (q_{+,1}; q_{+,0}))$

We will write $\mathfrak{n}$ for $\triangleleft \cdot \bot \cdot \triangleright$. The expression $\mathfrak{n}$ paraphrases the Natural Language *not.* We have e.g.:

– $[\![\mathfrak{n}]\!] = \langle -, \mathsf{id}, \mathsf{id}, \mathsf{id}, \bot, -, 0\rangle$

– $\langle -, \mathsf{id}, \mathsf{id}, \mathsf{id}, \bot, -, 0\rangle \bullet \langle +, q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, +, i\rangle =$
$\langle +, q_{-,1}, q_{-,0}, q_{+,1}, q_{+,0}, +, i\rangle \bullet \langle -, \mathsf{id}, \mathsf{id}, \mathsf{id}, \bot, -, 0\rangle =$
$\langle -, q_{+,1}, q_{+,0}, q_{-,1}, \bot, -, i\rangle$

– $\langle -, \mathsf{id}, \mathsf{id}, \mathsf{id}, \bot, -, 0\rangle \bullet \langle -, \mathsf{id}, \mathsf{id}, \mathsf{id}, \bot, -, 0\rangle$
$= \langle +, \mathsf{id}, \bot, \mathsf{id}, \mathsf{id}, +, 0\rangle$
This is, evidently, a somewhat strange meaning for a double negation. To make double negations work reasonably in the present set-up we have to interpose test operators.

7.2. Paraphrases

1. *Only if a farmer OWNS a donkey, does he beat it.*
   If he treats it well, he doesn't own it.
   $?_0(\triangle\cdot\exists x\cdot\mathsf{farmer}(x)\cdot\triangle\cdot\triangleleft\cdot\mathsf{owns}(x,y)\cdot\triangleright\cdot\triangle\cdot\exists y\cdot\mathsf{donkey}(y)\cdot\triangle\cdot\mathsf{beats}(x,y))\cdot$
   $?_0(\mathsf{w\text{-}treats}(x,y)\cdot\triangleleft\cdot ?_0(\bot\cdot\triangleright\cdot\mathsf{own}(x,y)))$

2. *Only if a FARMER owns a donkey, does he beat it.*
   $?_0(\triangle\cdot\exists x\cdot\triangleleft\cdot\mathsf{farmer}(x)\cdot\triangleright\cdot\triangle\cdot\mathsf{owns}(x,y)\cdot\triangle\cdot\exists y\cdot\mathsf{donkey}(y)\cdot\triangle\cdot\mathsf{beats}(x,y))$

3. *Every dog sees a cat. It chases it.*
   $?_0(\triangle\cdot\exists x\cdot\mathsf{dog}(x)\cdot\triangle\cdot\triangleleft\cdot\mathsf{sees}(x,y)\cdot\triangle\cdot\exists y\cdot\mathsf{cat}(y)\cdot\triangle)\cdot ?_0(\mathsf{chases}(x,y))$

4. *Mary did not see Karin*
   $?_0(\mathfrak{n}\cdot\mathsf{sees}(m,k))$
   Alternatively:
   $?_0(\triangle\cdot\exists x\cdot x = m\cdot\triangle\cdot ?_0(\mathfrak{n}\cdot\mathsf{sees}(x,y))\cdot\triangle\cdot\exists y\cdot k = y\cdot\triangle)$

5. *A man comes in. He sees nobody in the room*
   $?_0(\triangle\cdot\exists x\cdot\mathsf{man}(x)\cdot\triangle\cdot\mathsf{comes\text{-}in}(x))\cdot$
   $?_1(\mathsf{sees}(x,y)\cdot\mathfrak{n}\cdot\triangle\cdot\exists y\cdot\mathsf{person}(y)\cdot\triangle\cdot\mathsf{in\text{-}room}(y))$
   Replacing the $?_1$ by $?_0$ would make *a man* dependent on *a person.* The second sentence gets the same meaning as *he does not see anybody in the room.* Note that we may view $\mathfrak{n}\cdot\triangle\cdot\exists y\cdot\mathsf{person}(y)\cdot\triangle$ as giving the meaning of *nobody.*

6. *Nobody sees nobody*
   $?_0(\mathfrak{n} \cdot \triangle \cdot \exists x \cdot \mathsf{person}(x) \cdot \triangle \cdot ?_0(\mathsf{sees}(x, y) \cdot \mathfrak{n} \cdot \triangle \cdot \exists y \cdot \mathsf{person}(y) \cdot \triangle))$
   This gives us the meaning of *everybody sees someone*. Note that
   $?_0(\mathfrak{n} \cdot \triangle \cdot \exists x \cdot \mathsf{person}(x) \cdot \triangle \cdot \mathsf{sees}(x, y) \cdot \mathfrak{n} \cdot \triangle \cdot \exists y \cdot \mathsf{person}(y) \cdot \triangle)$
   gives the unexpected meaning specified by:
   $\triangle \cdot \exists x \cdot \mathsf{person}(x) \cdot \exists y \cdot \mathsf{person}(y) \cdot \triangle$
   This shows, that at least in the present approach, for two *not*'s or *no*'s, we have to choose which one governs which one.

7. *No dog sees no cat. It chases it.*
   $?_0(\mathfrak{n}\cdot\triangle\cdot\exists x\cdot\mathsf{dog}(x)\cdot\triangle\cdot?_0(\mathsf{sees}(x, y)\cdot\mathfrak{n}\cdot\triangle\cdot\exists y\cdot\mathsf{cat}(y)\cdot\triangle))\cdot?_0(\mathsf{chases}(x, y))$

7.3. On the Retrospective Construction

As in section 5 we will go from general to specific. Let three monoids $\mathcal{A} := \langle A, \cdot, \mathsf{id}\rangle$, $\mathcal{B} := \langle B, \bullet, \mathsf{id}\rangle$ and $\mathcal{C} := \langle C, \cdot, \mathsf{id}\rangle$ be given. Suppose μ is a left action of $\mathcal{A}$ on $\mathcal{B}$ and ν is a right action of $\mathcal{C}$ on $\mathcal{B}$. As usual we write $a \cdot b$ for $\mu(a, b)$ and $b \cdot c$ for $\nu(b, c)$. We demand the following further property:

– $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

Define a new algebra, say $\mathcal{D}$, as follows. $\mathcal{D} = \langle A \times B \times C, \bullet, \mathsf{id}\rangle$. Here:

– $\mathsf{id}_{\mathcal{D}} := \langle \mathsf{id}_{\mathcal{A}}, \mathsf{id}_{\mathcal{B}}, \mathsf{id}_{\mathcal{C}}\rangle$

– $\langle a_1, b_1, c_1\rangle \bullet \langle a_2, b_2, c_2\rangle := \langle a_1 \cdot a_2, (b_1 \cdot c_2) \bullet (a_1 \cdot b_2), c_1 \cdot c_2\rangle$

We claim that $\mathcal{D}$ is a monoid. We verify associativity.

Let $d_i := \langle a_i, b_i, c_i\rangle$, for $i = 1, 2, 3$. We have:

$$\begin{aligned}
(d_1 \bullet d_2) \bullet d_3 &= (\langle a_1, b_1, c_1\rangle \bullet \langle a_2, b_2, c_2\rangle) \bullet \langle a_3, b_3, c_3\rangle \\
&= \langle a_1 \cdot a_2, (b_1 \cdot c_2) \bullet (a_1 \cdot b_2), c_1 \cdot c_2\rangle \bullet \langle a_3, b_3, c_3\rangle \\
&= \langle a_1 \cdot a_2 \cdot a_3, (((b_1 \cdot c_2) \bullet (a_1 \cdot b_2)) \cdot c_3) \bullet (a_1 \cdot a_2 \cdot b_3), \\
&\quad c_1 \cdot c_2 \cdot c_3\rangle \\
&= \langle a_1 \cdot a_2 \cdot a_3, (b_1 \cdot c_2 \cdot c_3) \bullet (a_1 \cdot b_2 \cdot c_3) \bullet (a_1 \cdot a_2 \cdot b_3), \\
&\quad c_1 \cdot c_2 \cdot c_3\rangle \\
&= \langle a_1 \cdot a_2 \cdot a_3, (b_1 \cdot c_2 \cdot c_3) \bullet (a_1 \cdot ((b_2 \cdot c_3) \bullet (a_2 \cdot b_3))), \\
&\quad c_1 \cdot c_2 \cdot c_3\rangle \\
&= \langle a_1, b_1, c_1\rangle \bullet \langle a_2 \cdot a_3, (b_2 \cdot c_3) \bullet (a_2 \cdot b_3), c_2 \cdot c_3\rangle \\
&= \langle a_1, b_1, c_1\rangle \bullet (\langle a_2, b_2, c_2\rangle \bullet \langle a_3, b_3, c_3\rangle) \\
&= d_1 \bullet (d_2 \bullet d_3)
\end{aligned}$$

We will replace $\mathcal{B}$ by a more specific monoid and our actions by more specific actions. Let $\mathcal{E}$ be any monoid and let I be any non-empty set. The monoid $\mathcal{B}$ that we are interested in, will be of the form $\mathcal{E}^I$. Let ρ be a right action of $\mathcal{A}$ on I and let λ be a left action of $\mathcal{C}$ on I. We write $i \cdot a$ for $\rho(i, a)$ and $c \cdot i$ for $\lambda(c, i)$. We demand that $(c \cdot i) \cdot a = c \cdot (i \cdot a)$. We take $(a \cdot f)(i) := \mu(a, f)(i) := f(i \cdot a)$ and $(f \cdot c)(i) := \nu(f, c)(i) := f(c \cdot i)$.

We make things even more special by taking $I := A \times C$, $\langle a, c\rangle \cdot a' := \langle a \cdot a', c\rangle$ and $c' \cdot \langle a, c\rangle := \langle a, c' \cdot c\rangle$.

Finally, the monoidal part of $\Phi^{\mathsf{pol}}_{\mathsf{pol,sco}}(\mathcal{Q})$, can be obtained, modulo isomorphism, by taking $\mathcal{E}$ the monoidal part of $\mathcal{Q}$, by taking $\mathcal{A} := \mathsf{Posco} = \mathsf{Pol} \times \mathsf{Add}_2$ and by taking $\mathcal{C} := \mathsf{Pol}$.

In appendix D we sketch how to understand the construction of this subsection in terms of the construction of subsection 5.

8. Perspectives

In the present treatment we still have connectives like test that employ syntactical rather than semantical memory. Thus, while e.g. the bowtie is a meaningful bracket, reflected by an action in the semantics, the round brackets accompanying test are syncategorematical. I think we can avoid this and push all memory into the semantics. This would force us to construct substantially more complex meanings. Something like stacking cells, perhaps, where on each level we have a pair of relations stored.[7] For example we would like to represent the discourse *John sees nobody* somewhat as follows.

in between sentences ...

hearer Evidently, we have a break in the discourse here. I wil put a blocker in my database to remind myself of this.

speaker *John*

hearer OK, ..., John. Got it. What about him?

speaker *sees*

hearer Yes, he sees. But, what does he see?

speaker *nobody*

hearer ... Oh, fooled ... Good that I put down that blocker. I will add somebody and then convert all information from the blocker on to negative polarity.

[7] See (Visser, 1994) and (Visser and Vermeulen, 1996) for a discussion of stacking cells.

in between sentences . . .

Another issue is the issue of global versus local assignments. We have followed the classic Groenendijk & Stokhof approach by employing assignments that are defined on a given fixed set of variables.[8] This strategy forces us to give the existential quantifier the meaning *reset*, rather than *create*. Localizing the approach forces the issue of referential time travel in clearer perspective: how can we constrain a variable that has not yet been introduced?

A third issue is the treatment of argument places. The more flexible treatment of argument places proposed in (Visser and Vermeulen, 1996) could combine nicely with the present approach. E.g a more principled treatment of (*Only John was hungry*) could become available.

A fourth matter is extending the present framework to phenomena connected to focus. Rick Nouwen in his masters thesis (Nouwen, 1999a) made a preliminary exploration of this field. See also his (Nouwen, 1999b).

Acknowledgements

I thank Nuel Belnap for asking me the question about the semantics of

Only if a farmer owns a donkey, does he beat it.

I thank Kees Vermeulen, Freek Wiedijk, Rick Nouwen and Henriëtte de Swart for enlightening and stimulating discussions. I am grateful to Bart Jacobs for helpful comments. I thank the referee for his/her careful and critical report, which helped to improve the paper on several points.

Appendix

A. Relational Representation

The result of our construction Φ_{pol} was not a relational algebra where $\bullet$ ends up as relation composition. In the case where the input monoid is a monoid of relations between assignments, we can elegantly rephrase things in such a way that pol becomes an extra variable with certain

[8] Calling the opposition at issue here *total versus partial* is a less happy choice of words. A function is partial if it does not give values to some inputs present among a previously specified domain. We are talking here about growing domains. Our assignments are total on the referents at hand.

special values and in such a way that $\bullet$ becomes relation composition between relations on extended assignments.

Suppose $\mathcal{R}$ is a monoid of relations on D^{Var}. Define the mapping: $\mathcal{F} : \langle r_-, r_+, \alpha\rangle \mapsto R$, where R is a relation between functions on $\mathsf{Var} \cup \{\mathsf{pol}\}$ that send elements of Var to elements of D and that send pol to $+$ or $-$. We write f^* for $f \restriction \mathsf{Var}$. Our new relation R is given by:

$$fRg :\Leftrightarrow f^*(r_{f(\mathsf{pol})})g^* \text{ and } g(\mathsf{pol}) = f(\mathsf{pol}) \cdot \alpha$$

Let $R := \mathcal{F}(\langle r_-, r_+, \alpha\rangle)$, $S := \mathcal{F}(\langle s_-, s_+, \beta\rangle)$, $T := \mathcal{F}(\langle r_-, r_+, \alpha\rangle \bullet \langle s_-, s_+, \beta\rangle)$. We find:

$$\begin{aligned} f(R;S)g \;&\Leftrightarrow\; \exists h\; f^*(r_{f(\mathsf{pol})})h^*(s_{h(\mathsf{pol})})g^*,\;\; h(\mathsf{pol}) = f(\mathsf{pol}) \cdot \alpha \text{ and} \\ &\;\; g(\mathsf{pol}) = h(\mathsf{pol}) \cdot \beta \\ &\Leftrightarrow\; f^*(r_{f(\mathsf{pol})}; s_{f(\mathsf{pol})\cdot\alpha})g^* \text{ and } g(\mathsf{pol}) = f(\mathsf{pol}) \cdot \alpha \cdot \beta \\ &\Leftrightarrow\; fTg \end{aligned}$$

Moreover, clearly, $\mathcal{F}(\mathsf{id}) = \mathsf{id}$. Also it is easily seen that $\mathcal{F}$ is injective. So we succeeded in embedding the monoidal part of our e-monoid into relations with composition. Note that the $\perp$ and the $\rightarrow$ of the e-monoid do *not* correspond with the empty relation (of the new relations) and the relational $\rightarrow$ on the new relations.

It is quite easy to generalize the present idea to the context of the more abstract treatment of section 5.

B. Translations

We fix an ℓ-signature Σ.

B.1. Predicate Logic into DPL

We show how to translate the formulas of ordinary Predicate Logic into the formulas of DPL. Suppose e.g. we axiomatized Predicate Logic with logical constants $\top$, $\perp$, $\wedge$, $\rightarrow$, $\forall$, $\exists$. Then the translation, say ϵ, can be taken:

- ϵ commutes with $P(r_1, \cdots, r_n)$, $\top$, $\perp$, $\rightarrow$
- $(\phi \wedge \psi)^\epsilon := \phi^\epsilon; \psi^\epsilon$
- $(\forall x\; \phi)^\epsilon := (\exists x \rightarrow \phi^\epsilon)$
- $(\exists x\; \phi)^\epsilon := ?(\exists x \cdot \phi^\epsilon)$

In a given model, we find $[\phi^\epsilon] = \mathsf{diag}(\|\phi\|))$. Here $\|.\|$ is the interpretation function of Predicate Logic. Moreover we have: $\mathcal{M}, f \models_{\mathsf{pred}} \phi \Leftrightarrow \mathcal{M}, f \models_{\mathsf{DPL}} \phi^\epsilon$.

B.2. DPL into any Dynamic Predicate Logic

Consider any dpl $\mathsf{DPL}_{\mathcal{D}}$. Here $\mathcal{D}$ contains Φ, t, and emb. We translate the DPL-formulas of ℓ-signature Σ into the $\mathsf{DPL}_{\mathcal{D}}$-formulas of ℓ-signature Σ. ζ is defined, by recursion on the DPL-language[9], as follows. ζ commutes with all atomic formulas and connectives except $\rightarrow$. $(\phi \rightarrow \psi)^\zeta := t(\phi^\zeta, \psi^\zeta)$. We find: $[\![\phi^\zeta]\!] = \mathsf{emb}([\phi])$. Moreover, for a suitable model $\mathcal{M}$, we have $\mathcal{M}, f \models \phi \Leftrightarrow \mathcal{M}, f \models_{\mathcal{D}} \phi^\zeta$.

B.3. DPL with Polarity Switch into DPL

The translation backwards from $\mathsf{DPL}_{\mathsf{pol}}$ into DPL asks for a different approach. We translate a $\mathsf{DPL}_{\mathsf{pol}}$-formula ϕ, via the translation η, to a triple $\langle \phi_-, \phi_+, \alpha \rangle$, where the ϕ_β are DPL-formulas. We define two operations on triples:

- $\langle \phi_-, \phi_+, \alpha \rangle \bullet \langle \psi_-, \psi_+, \beta \rangle = \langle \phi_- \bullet \psi_{-\alpha}, \phi_+ \bullet \psi_{+\alpha}, \alpha \cdot \beta \rangle$
- $\mathsf{test}(\langle \phi_-, \phi_+, \alpha \rangle) := \langle \top, (\phi_- \rightarrow \phi_+), + \rangle$

η is given by the following clauses.

- $(P(r_1, \cdots, r_n))^\eta := \langle \top, P(r_1, \cdots, r_n), + \rangle$
- $(\exists v)^\eta := \langle \top, \exists v, + \rangle$
- $\top^\eta := \langle \top, \top, + \rangle$
- $\bot^\eta := \langle \top, \bot, + \rangle$
- $\bowtie^\eta := \langle \top, \top, - \rangle$
- $(\phi \cdot \psi)^\eta := \phi^\eta \bullet \psi^\eta$
- $(?(\phi))^\eta := \mathsf{test}(\phi^\eta)$

Suppose $\phi^\eta = \langle \phi_-, \phi_+, \alpha \rangle$. We find: $[\![\phi]\!] = \langle [\phi_-], [\phi_+], \alpha \rangle$. Moreover, for a suitable model $\mathcal{M}$, we have $\mathcal{M}, f \models_{\mathsf{pol}} \phi \Leftrightarrow \mathcal{M}, f \models (\phi_- \rightarrow \phi_+)$.

There is an analogy between the semi-syntactical triples and DRS's. We will elaborate on that analogy in subsection B.5.

[9] Remember that this recursion is really between free e-monoids of different signature. It is easy to see that it is well defined.

B.4. DPL with Switches for Polarity and Scope into DPL

We turn to 'translation' from $\mathsf{DPL}_{\mathsf{pol,sco}}$ to DPL. This translation is like the one $\mathsf{DPL}_{\mathsf{pol}}$ to DPL described in subsection B.3. We work again with the slightly modified DPL-formulas of subsection B.3. We translate a $\mathsf{DPL}_{\mathsf{pol,sco}}$-formula ϕ, via the translation κ, to a sextuple $\langle \phi_{-,1}, \phi_{-,0}, \phi_{+,1}, \phi_{+,0}, \alpha, i\rangle$. We define three operations on sextuples:

- $\langle \phi_{-,1}, \phi_{-,0}, \phi_{+,1}, \phi_{+,0}, \alpha, i\rangle \bullet \langle \psi_{-,1}, \psi_{-,0}, \psi_{+,1}, \psi_{+,0}, \beta, j\rangle =$
  $\langle \chi_{-,1}, \chi_{-,0}, \chi_{+,1}, \chi_{+,0}, \alpha \cdot \beta, i+j\rangle$
  where $\chi_{\gamma,k} := \phi_{\gamma,k} \bullet \psi_{\gamma\cdot\alpha,k\cdot i}$
- $\mathsf{test}_0(\langle \phi_{-,1}, \phi_{-,0}, \phi_{+,1}, \phi_{+,0}, \alpha, i\rangle) :=$
  $\langle \phi_{-,1}, \top, \phi_{+,1}, (\phi_{-,0} \rightarrow \phi_{+,0}), +, 0\rangle$
- $\mathsf{test}_1(\langle \phi_{-,1}, \phi_{-,0}, \phi_{+,1}, \phi_{+,0}, \alpha, i\rangle) :=$
  $\langle \top, \top, \top, ((\phi_{-,1} \bullet \phi_{-,0}) \rightarrow (\phi_{+,1} \bullet \phi_{+,0})), +, 0\rangle$

κ is given by the following clauses.

- $(P(r_1, \cdots, r_n))^\kappa := \langle \top, \top, \top, P(r_1, \cdots, r_n), +, 0\rangle$
- $(\exists v)^\kappa := \langle \top, \exists v, +\rangle$
- $\top^\kappa := \langle \top, \top, \top, \top, +, 0\rangle$
- $\bot^\kappa := \langle \top, \top, \top, \bot, +, 0\rangle$
- $\bowtie^\kappa := \langle \top, \top, \top, \top, -, 0\rangle$
- $\vartriangle^\kappa := \langle \top, \top, \top, \top, +, 1\rangle$
- $(\phi \cdot \psi)^\kappa := \phi^\kappa \bullet \psi^\kappa$
- $(?_0(\phi))^\kappa := \mathsf{test}_0(\phi^\kappa)$
- $(?_1(\phi))^\kappa := \mathsf{test}_1(\phi^\kappa)$

Suppose $\phi^\kappa = \langle \phi_{-,1}, \phi_{-,0}, \phi_{+,1}, \phi_{+,0}, \alpha, i\rangle$. We find:

$$[\![\phi]\!] = \langle [\phi_{-,1}], [\phi_{-,0}], [\phi_{+,1}], [\phi_{+,0}], \alpha, i\rangle.$$

Moreover, for a suitable model $\mathcal{M}$, we have

$$\mathcal{M}, f \models_{\mathsf{pol,sco}} \phi \Leftrightarrow \mathcal{M}, f \models ((\phi_{-,1} \bullet \phi_{-,0}) \rightarrow (\phi_{+,1} \bullet \phi_{+,0})).$$

B.5. DRT into DPL with and without Switches

We can translate the language of Discourse Representation Theory, DRT, with its semantics into the language cum semantics of $\mathsf{DPL}_{\mathsf{pol,sco}}$. For information about DRT the reader is referred to (Kamp, 1981), (Kamp and Reyle, 1993), (Zeevat, 1991). Let a signature Σ be given. The DRT-language of signature Σ is just the DPL-language of the same signature. A DRT-meaning (in a given model $\mathcal{M}$ of signature Σ is a pair $\langle V, F\rangle$, where V is a finite set of variables and where F is a set of assignments. We write $[V]$ for the relation between assignments with

$$f[V]g :\Leftrightarrow \forall w{\in}\mathsf{Var} \setminus V \;\; f(w) = g(w)$$

We assign to a DRT-meaning $\langle V, F\rangle$ a relation $[\langle V, F\rangle]$, abbreviated as $[V, F]$, as follows: $[V, F] := [V]; \mathsf{diag}(F)$.

We define two operations on DPL-meanings.

- $\langle V, F\rangle \bullet \langle W, G\rangle := \langle V \cup W, F \cap G\rangle$
- $(\langle V, F\rangle \to \langle W, G\rangle) := \langle \emptyset, \mathsf{dom}([V, F] \to [W, G])\rangle$
  Here dom is the function that gives the domain of the given relation.

Clearly $\bullet$ gives us a monoid with identity $\langle \emptyset, \mathsf{Ass}\rangle$. Here is the definition of the DRT-interpretation function:

- $[\![P(r_1, \cdots, r_n)]\!] := \langle \emptyset, \|P(r_1, \cdots, r_n)\|\rangle$.
  Remember that $\|P(r_1, \cdots, r_n)\|$ was defined as the meaning assigned to $P(r_1, \cdots, r_n)$ in Predicate Logic.
- $[\![\exists v]\!] := \langle \{v\}, \mathsf{Ass}\rangle$
- $[\![\top]\!] := \langle \emptyset, \mathsf{Ass}\rangle$
- $[\![\bot]\!] := \langle \emptyset, \emptyset\rangle$
- $[\![\phi \cdot \psi]\!] := [\![\phi]\!] \bullet [\![\psi]\!]$
- $[\![\phi \to \psi]\!] := ([\![\phi]\!] \to [\![\psi]\!])$

Satisfaction for DRT is defined as follows.

- $\mathcal{M}, f \models_{\mathsf{drt}} \phi :\Leftrightarrow \exists g \; f[[\![\phi]\!]]g$

We can translate DRT to $\mathsf{DPL}_{\mathsf{pol,sco}}$ as follows. Say the translation is δ.

- δ commutes with formulas of the form $P(r_1, \ldots, r_n)$, $\top$, $\bot$
- $(\exists v)^{\delta} := \triangle \cdot \exists v \cdot \triangle$

– δ commutes with $\cdot$

– $(\phi \to \psi)^\delta := ?_1(\bowtie \cdot \phi^\delta \cdot \bowtie \cdot \psi^\delta)$

We embed DRT-meanings into $\mathsf{DPL}_{\mathsf{pol,sco}}$-meanings via:

$$\mathsf{emb}^\circ : \langle V, F\rangle \mapsto \langle \mathsf{id}, \mathsf{id}, [V], \mathsf{diag}(F), +, 0\rangle.$$

We find: $[\![\phi^\delta]\!] = \mathsf{emb}^\circ([\![\phi]\!])$. Moreover:

$$\mathcal{M}, f \models_{\mathsf{drt}} \phi \Leftrightarrow \mathcal{M}, f \models_{\mathsf{dpl,pol,sco}} \phi^\delta.$$

It is interesting to translate first DRT via δ to $\mathsf{DPL}_{\mathsf{pol,sco}}$, and subsequently via κ to DPL. We get e.g.:

$$\exists x.P(x).\exists y.Q(y) \quad\mapsto\quad \langle \top, \top, \exists x\,\exists y, P(x)\,Q(y), +, 0\rangle$$

This last translation is strongly reminiscent of a Discourse Representation Structure or DRS, in our example: $\langle\{x, y\}, \{P(x), Q(y)\}\rangle$. The main differences are (i) a few empty locations, (ii) strings of existential quantifiers instead of sets of the corresponding variables, (iii) strings of conditions instead of sets of conditions. The empty locations are inessential. They are not activated. Except for the 0, they are due to the fact that we are carrying around a polarity mechanism that is absent from DRT. We can abstract from the strings in the case of DRT for a simple reason. Reset relations are commutative and idempotent. Conditions have the same properties. This fact makes it possible to abstract from order and number of occurrences in strings of resets and in strings of conditions, thus obtaining sets.

C. The Grothendieck Construction

The reader who knows a bit about category theory, will have noted that the construction of section 5 is just the Grothendieck Construction for monoids. See e.g. (Barr and Wells, 1989) or (Bénabou, 1985) or (Tarlecki et al., 1991) or (Jacobs, 1999). We will give a bit of detail here.

We will think of categories in the usual way. Thus, $f \circ g$ means: first g then f. We consider a monoid $\mathcal{M} = \langle M, \bullet, \mathsf{id}\rangle$ as a one object category in the following way:

– $\mathsf{Ob}_{\mathcal{M}} := \{*\}$, $\mathsf{Arr}_{\mathcal{M}} := M$

– $\mathsf{dom}(m) := \mathsf{cod}(m) := *$

– $\mathsf{id}_* := \mathsf{id}_{\mathcal{M}}$

– $m \circ n := n \bullet m$ (The reversal of order is because we intend '$n \bullet m$' to mean: first n, then m.)

Consider two monoids $\mathcal{A}$ and $\mathcal{B}$ and let ν be a left action of $\mathcal{A}$ on $\mathcal{B}$. Define the contravariant functor $\Theta = \Theta_\nu$ from $\mathcal{A}$ to Cat, by:

– $\Theta(*) := \mathcal{B}$

– $(\Theta(a))(*) := *$

– $(\Theta(a))(b) := a \cdot b$

It is easily seen that $\Theta(a)$ is a functor from $\mathcal{B}$ to $\mathcal{B}$. Moreover, $\Theta(\mathsf{id}_{\mathcal{A}}) = \mathsf{ID}_{\mathcal{B}}$ and:

$$\begin{aligned}(\Theta(a_1 \circ a_2))(b) &= (\Theta(a_2 \cdot a_1))(b)\\ &= (a_2 \cdot a_1) \cdot b\\ &= a_2 \cdot (a_1 \cdot b)\\ &= (\Theta(a_2))((\Theta(a_1))(b))\\ &= (\Theta(a_2) \circ \Theta(a_1))(b)\end{aligned}$$

So Θ is indeed a contravariant functor. We get: $\int_{\mathcal{A}} \Theta_\mu = \sum_{\mathcal{A}} \mu$. Here $\int_{\mathcal{A}} \Theta_\mu$ is the Grothendieck completion of the functor Θ_μ, considered as a split indexed category.

D. Opposites of Opposites

In this appendix we show how the construction of subsection 7.3 can be obtained from the construction of section 5.

For any monoid $\mathcal{M}$, we take $\mathcal{M}^{\mathsf{op}}$ to be the opposite monoid of $\mathcal{M}$, i.e. the monoid we obtain by taking the monoidal operation in reverse order. Of course, $\mathcal{M}^{\mathsf{op}}$ is just the opposite of $\mathcal{M}$ considered as a category. Note the following simple facts:

1. If ρ is a right action of $\mathcal{N}$ on $\mathcal{M}$, then $\rho^{\mathsf{op}} := \lambda$, where $\lambda(m, n) := \rho(n, m)$, is a left action of $\mathcal{N}^{\mathsf{op}}$ on $\mathcal{M}$. If we use $*$ is to denote both λ and the monoid operation of $\mathcal{N}^{\mathsf{op}}$, we have e.g.

$$(n_1 * n_2) * m = m \cdot (n_2 \cdot n_1) = (m \cdot n_2) \cdot n_1 = n_1 * (n_2 * m)$$

2. If λ is a left action of $\mathcal{N}$ on $\mathcal{M}$, then λ^* with where $\lambda^*(m, n) := \lambda(m, n)$, is a left action of $\mathcal{N}$ on $\mathcal{M}^{\mathsf{op}}$. (So the only difference between λ and λ^* consists in the structure acted upon.)

Let $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ be monoids and let ν be a left action of $\mathcal{A}$ on $\mathcal{B}$ and let μ be a right action of $\mathcal{C}$ on $\mathcal{B}$. Define a right action ρ_μ of $\mathcal{C}$ on $\mathcal{E} := \sum_{\mathcal{A}} \nu$ as follows:

– $\langle a, b\rangle \cdot c := \langle a, b \cdot c\rangle$

We have e.g.

$$\begin{aligned}(\langle a_1, b_1\rangle \bullet \langle a_2, b_2\rangle) \cdot c &= \langle a_1 \cdot a_2, b_1 \bullet (a_1 \cdot b_2)\rangle \cdot c \\ &= \langle a_1 \cdot a_2, (b_1 \bullet (a_1 \cdot b_2)) \cdot c\rangle \\ &= \langle a_1 \cdot a_2, (b_1 \cdot c) \bullet (a_1 \cdot b_2 \cdot c)\rangle \\ &= \langle a_1, b_1 \cdot c\rangle \bullet \langle a_2, b_2 \cdot c\rangle \\ &= (\langle a_1, b_1\rangle \cdot c) \bullet (\langle a_2, b_2\rangle \cdot c)\end{aligned}$$

$\mathcal{D}$ is isomorphic to $(\sum_{\mathcal{C}^{\mathsf{op}}} \rho_\mu^{\mathsf{op},*})^{\mathsf{op}}$ via the mapping $\langle a, b, c\rangle \mapsto \langle c, \langle a, b\rangle\rangle$. Let's use $\circledast$ for the opposite of $\bullet$ and $\odot$ for the opposite of $\cdot$. We have e.g.

$$\begin{aligned}\langle c_1, \langle a_1, b_1\rangle\rangle \bullet \langle c_2, \langle a_2, b_2\rangle\rangle &= \langle c_2, \langle a_2, b_2\rangle\rangle \circledast \langle c_1, \langle a_1, b_1\rangle\rangle \\ &= \langle c_2 \odot c_1, \langle a_2, b_2\rangle \circledast (c_2 \odot \langle a_1, b_1\rangle)\rangle \\ &= \langle c_2 \odot c_1, \langle a_2, b_2\rangle \circledast \langle a_1, c_2 \odot b_1\rangle\rangle \\ &= \langle c_1 \cdot c_2, \langle a_1, b_1 \cdot c_2\rangle \bullet \langle a_2, b_2\rangle\rangle \\ &= \langle c_1 \cdot c_2, \langle a_1 \cdot a_2, (b_1 \cdot c_2) \bullet (a_1 \cdot b_2)\rangle\rangle\end{aligned}$$

References

Barr, M. and C. Wells: 1989, *Category Theory for Computing Science*. New York: Prentice Hall.

Bénabou, J.: 1985, 'Fibered categories and the foundations of naive category theory'. *The Journal of Symbolic Logic* **50(1)**, 10–37.

Groenendijk, J. and M. Stokhof: 1991, 'Dynamic Predicate Logic'. *Linguistics and Philosophy* **14**, 39–100.

Hollenberg, M.: 1997, 'An equational axiomatisation of dynamic negation and relational composition'. *Journal of Language, Logic and Information* **6**(4), 381–401.

Jacobs, B.: 1999, *Categorical Logic and Type Theory*, No. 141 in Studies in Logic and the Foundations of Mathematics. Amsterdam: North Holland.

Kamp, H.: 1981, 'A Theory of Truth and Semantic Representation'. In: J. G. et al. (ed.): *Truth, Interpretation and Information*. Dordrecht: Foris, pp. 1–41.

Kamp, H. and U. Reyle: 1993, *From Discourse to Logic*, Vol. I, II. Dordrecht: Kluwer.

Nouwen, R.: 1999a, 'Action-Only Semantics: a study of the use of control elements in DPL'. Master's thesis, CKI, Dept. of Philosophy, Utrecht University. `http://www.phil.uu.nl/ckiscripties.html`.

Nouwen, R.: 1999b, 'DPL with control elements'. In: P. Dekker (ed.): *Proceedings of the twelfth Amsterdam Colloquium.*

Tarlecki, A., R. Burstall, and J. Goguen: 1991, 'Some fundamental algebraic tools for the semantics of computation: Part 3. Indexed categories'. **91**, 239–264.

Visser, A.: 1994, 'Actions under Presuppositions'. In: J. van Eijck and A. Visser (eds.): *Logic and Information Flow.* Cambridge, Mass.: MIT Press, pp. 196–233.

Visser, A.: 1997, 'Dynamic Relation Logic is the logic of DPL-relations'. *Journal of Language, Logic and Information* **6**(4), 441–452.

Visser, A. and C. Vermeulen: 1996, 'Dynamic Bracketing and Discourse Representation'. *Notre Dame Journal of Formal Logic* **37**, 321–365.

Zeevat, H.: 1991, 'A Compositional Approach to DRT'. *Linguistics and Philosophy* **12**, 95–131.

Address for Offprints:
Albert Visser
Department of Philosophy
Heidelberglaan 8, 3584 CS Utrecht
The Netherlands