Approximate Databases: A Support Tool for Approximate Reasoning

Patrick Doherty[*], Martin Magnusson[*], Andrzej Szałas[*,]**

[*] *Department of Computer and Information Science,*
University of Linköping, SE-581 83 Linköping, Sweden,
email: {patdo,marma,andsz}@ida.liu.se
[**] *The University of Economics and Computer Science*
Wyzwolenia 30, 10-106 Olsztyn, Poland

ABSTRACT. *This paper describes an experimental platform for approximate knowledge databases called the Approximate Knowledge Database (*AKDB*), based on a semantics inspired by rough sets. The implementation is based upon the use of a standard* SQL *database to store logical facts, augmented with several query interface layers implemented in* JAVA *through which extensional, intensional and local closed world nonmonotonic queries in the form of crisp or approximate logical formulas can be evaluated tractably. A graphical database design user interface is also provided which simplifies the design of databases, the entering of data and the construction of queries. The theory and semantics for* AKDBs *is presented in addition to application examples and details concerning the database implementation.*

KEYWORDS: *approximate reasoning, approximate databases, knowledge representation, second-order quantifier elimination*

1. Introduction

An essential component in many AI agent-based architectures is the agent's knowledge representation component which includes a variety of knowledge and data repositories with associated inference mechanisms. In the case of robotic systems, the knowledge representation component is often intended to provide models of aspects of the robot's embedding environment and its own and other agent capabilities. Designing, specifying and implementing KR components in a robotic system is particularly challenging due to soft and hard real-time constraints and the fact that knowledge structures are often derived from sensory input and fusion processes.

It is becoming increasingly important to take seriously the gap between access to low-level sensory data and its fusion and integration with more qualitative knowledge structures. These signal-to-symbol transformations should be viewed as an on-going process with a great deal of feedback between the levels of processing. In addition, because the embedding environments are often as complex and dynamic as those faced by humans, the knowledge representations which are used as models of the environment must necessarily be partial, elaboration tolerant and approximate in nature.

This type of requirement rules out the use of standard relational database technologies as commonly understood, where crisp relations are stored as tables and a standard closed-world assumption is implicit in the query mechanism associated with a relational database. The focus of our research is to relax many of these standard assumptions in order to meet the requirements associated with the use of knowledge representation components in robotic systems. For example, approximate relations will be used instead of crisp relations, an open-world assumption will be used instead of the closed-world assumption and our query language will support the specification of non-monotonic queries. In order to leverage the great amount of work already done with traditional relational database technologies, we will also show how it is possible to develop a theory of approximate databases and queries, but yet be able to compile such representations into standard SQL queries. In this manner, tractability is guaranteed through the use of smart encoding techniques.

A concrete domain, where such knowledge representation components are not only desirable, but necessary, involves the use of an unmanned aerial vehicle (UAV) operating over a road and traffic environment. In this case, the UAV system must be able to dynamically construct representations of objects it observes in the world and integrate these representations with other static representations such as concepts and taxonomic hierarchies about spatial or temporal relations, normative behavior of vehicles, traffic representations, etc.

In this case, the meaning of concepts such as *fast* or *slow*, *small* or *large vehicle*, *near*, *far*, or *between*, have a meaning different from that in applications with other temporal and spatial constraints. Assuming these primitive concepts as given and that they are continually re-grounded in changes in an operational environment via machine learning or sensor fusion, we would then like to use these primitive concepts in our knowledge representation structures. Since they are inherently approximate, any knowledge structure containing these concepts should also inherit or be influenced by these characteristics. For the purposes of this paper, we will assume that the approximate concepts have been generated in some manner. We will focus on their representation, storage and use, in what we call *approximate databases*.

The research described in this paper was initiated within the framework of the WITAS UAV Project.[1] This was a larger project involving the design and use of autonomous unmanned aerial vehicles and the development of an integrated software architecture for low- and high-end autonomous functionality [DOH 00a, DOH 04a].

1. WITAS is an acronym for the Wallenberg Information Technology and Autonomous Systems Laboratory at Linköping University, Sweden. UAV is an acronym for Unmanned Aerial Vehicles. The WITAS UAV Project ended officially in 2004. See `http://www.ida.liu.se/~patdo/auttek/` for continued activities in this area.

The research with approximate databases has been greatly influenced by the fact that the system had to be integrated and used in a complex UAV system which has been deployed in a number of complex scenarios involving photogrammetry, surveying, vehicle tracking and multi-platform scanning.

The UAV experimental platform offers an ideal environment for experimentation with the knowledge representation framework we propose, because the system architecture is rich with different types of knowledge representation structures, the operational environment is quite complex and dynamic, and signal-to-symbol transformations of data are an integral part of the architecture. In addition, much of the knowledge acquired by the UAV is necessarily approximate in nature. In several of the sections in this paper, we will use examples from this application domain to describe and motivate some of our techniques.

In this paper, we focus on a generalization of deductive database technology suitable for the characteristics of robotic domains mentioned previously. A standard deductive database (see, e.g., [ABI 95, PRZ 90]) can store ground atomic formulas to represent factual knowledge and use, for example, Horn-clause logic formulas as deductive rules to infer additional facts. Such databases often make assumptions, such as the assumption that the stored knowledge is precise or complete, that render them less suitable for the contexts we are interested in. *Approximate databases*, considered in a number of our publications (see, e.g., [DOH 99, DOH 03b, DOH 04b, DOH 04d, DOH 04e]) and summarized in book form in [DOH 06], relax many of these assumptions and become applicable in robotics contexts where standard techniques do not. An important part of our investigation involved the design, development and implementation of a database engine [MAG 05a], called the Approximate Knowledge Database (AKDB), that serves as an experimental platform for our research. Since the technology is built on top of traditional SQL relational database technology, its potential applicability goes well beyond the application domain described here although we have not yet pursued its use with other domains.

The paper is structured as follows. Section 2 introduces and motivates the concepts that are used in the remaining parts of the paper. Section 3 provides an overview of the architecture of approximate knowledge databases. In Section 4 we present syntax and semantics of approximate knowledge databases. Section 5 discusses the relation of our solutions to other work. In Section 6 the concepts introduced earlier are illustrated on a surveillance mission case study. Finally, Section 7 concludes the paper and Appendix A mentions some additional functionality.

An online interface for experimenting with AKDBs is available via
http://www.ida.liu.se/~marma/akdb/

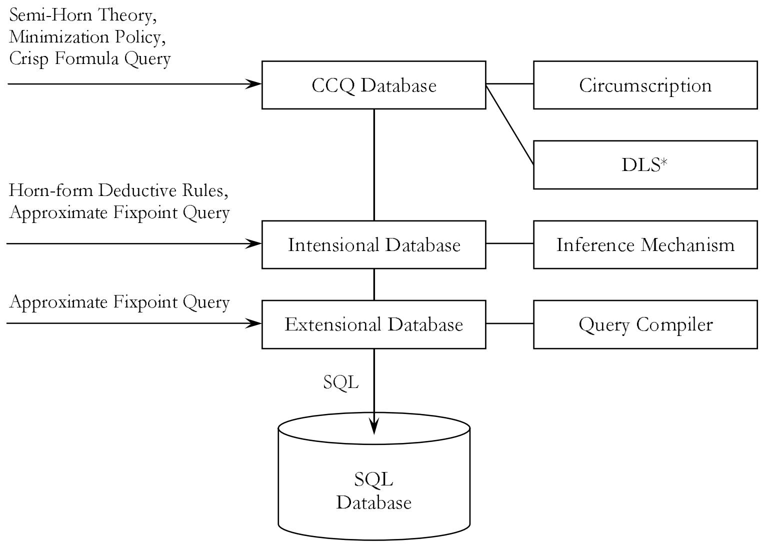


Figure 1. *An overview of the* AKDB *architecture.*

2. Overview and Preliminaries

2.1. *Approximate Knowledge Databases*

The approximate knowledge database augments a standard SQL database with several extension layers, each providing an extended query language building upon the lower layers. As shown in the overview in Figure 1, there are three main layers. The extensional database implements the concept of approximate relations based on the three-valued logic of Kleene under an open world assumption by using a compilation mechanism to regular database relations. The intensional database uses rules and an inference mechanism to deduce both new positive and negative facts. Finally, the CCQ database provides mechanisms to apply local closed world assumptions while evaluating a query through the use of second-order circumscription that is reduced to first-order logic using the DLS* second-order quantifier elimination algorithm. A more detailed description of the AKDB system is found in Section 3.

2.2. *Approximate Relations*

Approximate relations are introduced as basic elements to be stored in approximate databases. These are intended to naturally express knowledge with imprecise concepts and should be contrasted with the *crisp* (precise) relations and formulas used in standard deductive databases. Approximate relations are based on ideas from rough set theory.

In rough set theory, a rough set is characterized by two classical sets, one representing a lower approximation to the set and one representing an upper approximation to the set. Both lower and upper approximations to the set are defined in terms of equivalence classes of individuals in the domain of discourse where the partitions are generated relative to a set of attributes associated with individuals. The net result is a partitioning of individuals relative to the lower and upper approximations and the difference between the two, the boundary region of the rough set.

For approximate sets (relations), the basic semantics is generalized in a number of ways to create a semantics for approximate databases. Most importantly, rather than building up lower and upper approximations in terms of equivalence classes on individuals, one defines different types of neighborhood functions on individuals, where each function may be specified using different sets of constraints. Rough sets then become a special case of approximate sets with a particular set of constraints.

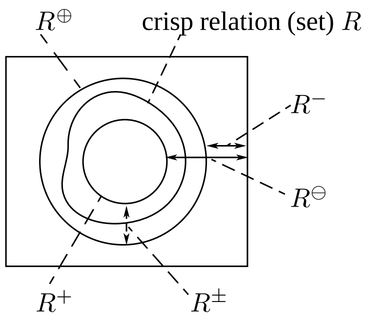


Figure 2. *Approximations of a relation (set).*

Any approximate relation R has (see Figure 2), a *positive part*, denoted by R^+, containing objects known to satisfy R, a *negative part*, denoted by R^-, containing objects known not to satisfy R, a *boundary part*, denoted by $R^{\pm}$, containing objects that are neither known to satisfy R nor known not to, a *positive-boundary part*, denoted by $R^{\oplus}$, containing objects in the positive or boundary part, a *negative-boundary part*, denoted by $R^{\ominus}$, containing objects in the negative or boundary part.

Approximations of a set (relation) are usually given by means of a family of neighborhoods of domain elements, as defined below. Note that such approximations not only provide us with techniques for dealing with vague concepts, but also are important in verifying whether in a given application domain one actually deals with rough concepts, tolerance-based concepts or maybe concepts reflecting yet other similarity constraints. Certain properties of similarities are also used to ensure the validity of certain assumptions as, e.g., those discussed in Section 2.5.5.

DEFINITION 1. — *Let* DOM *be a given domain of elements. Then any*

$$n : \textsc{Dom} \longrightarrow 2^{\textsc{Dom}}, \textit{ such that for any } u \in \textsc{Dom}, n(u) \neq \emptyset$$

is called a neighborhood *function.*

Sets $n(u)$, for $u \in \textsc{Dom}$ are interpreted as direct neighborhoods of elements. The neighborhoods are usually generated by some similarity relation σ, by assuming that

$$n(x) \stackrel{\mathrm{def}}{=} \{y \mid \sigma(x, y)\}$$

(see, e.g., [DOH 04f, DOH 03a]). For example, when σ is an equivalence relation, one has rough set approximations (cf. [PAW 91, PAL 04]), when σ is reflexive and symmetric, one has tolerance-based approximations (cf. [DOH 03a]).

In some applications it is also reasonable to consider even weaker similarity relations (cf. [DOH 04f, DOH 05]).

The *lower* and *upper approximation of* R *wrt* n is defined as:

$$\begin{array}{rcl} R_{n^+} & \stackrel{\mathrm{def}}{=} & \{u \in \textsc{Dom} : n(u) \subseteq R\} \\ R_{n^\oplus} & \stackrel{\mathrm{def}}{=} & \{u \in \textsc{Dom} : n(u) \cap R \neq \emptyset\}. \end{array}$$

Observe that the requirement that $n(u) \neq \emptyset$ in Definition 1 ensures that $R_{n^+} \subseteq R_{n^\oplus}$.

In the presence of the neighborhood function n, $R^+ \stackrel{\mathrm{def}}{=} R_{n^+}$ and $R^\oplus \stackrel{\mathrm{def}}{=} R_{n^\oplus}$.

As shown in [DOH 04f], one can obtain correspondences between similarities and approximations using second-order quantifier elimination techniques. The techniques described are similar to those used in modal correspondence theory. Such correspondences are crucial when one wants to make sure that the underlying reasoning mechanism used with approximate relations reflects the properties of, say, rough sets or tolerance spaces.

For example, in order to satisfy constraints associated with tolerance spaces, one requires that $R^+ \subseteq R$ (reflexivity of σ) and $R \subseteq \left(R^\oplus\right)^+$ (symmetry of σ). In order to satisfy constraints associated with rough sets, one has to additionally make sure that $R^+ \subseteq \left(R^+\right)^+$ (transitivity of σ).

One of the weakest requirements on approximations is that $R^+ \subseteq R^\oplus$. This requirement corresponds to the *seriality* of the underlying similarity relation, i.e., to the requirement that $\forall x \exists y \sigma(x, y)$. This observation is crucial for the soundness and completeness of the Feferman-Gilmore translation discussed in Section 2.5.5, since $R^+ \subseteq R^\oplus$ is equivalent to $R^+ \cap R^- = \emptyset$, required there.

REMARK 2. — As indicated in [DOH 04f], such correspondences can also be obtained based on modal correspondence theory.[2] This is done by translating R^+ into $\Box R$ and $R^\oplus$ into $\Diamond R$. Then the required properties of the similarity relation are exactly those of the accessibility relation in Kripke structures for modal logics. Some-

2. For a survey of modal correspondence theory see [BEN 84]. For automated techniques for correspondence theory see [GAB 92, SZA 93, NON 98, NON 99].

times one can then apply known facts from the area of modal logics, e.g. the fact that axiom 5:

$$\Diamond R \rightarrow \Box\Diamond R \text{ or, in the terminology of approximations, } R^{\oplus} \subseteq \left(R^{\oplus}\right)^{+}$$

can replace both properties of reflexivity and symmetry, since logic $KT5$ is $S5$. □

Approximate relations can be constructed in a number of ways (see, e.g., [DOH 06, DOH 04c, DOH 00b]), which include supervised machine learning, employing expert knowledge, and approximating logical theories.

2.3. *Open World Assumption*

Traditionally, classical reasoning and planning techniques have been developed for environments in which the reasoning agent is assumed to have complete information about the world in which it is embedded and the only changes to the world are the effects which result from the agent's invocation of actions. Under this assumption, an efficient means of representing negative information about the world in each planning or reasoning state is to apply the *Closed World Assumption* (CWA) [ABI 95, REI 78]. In this case, information about the world, absent in a state, is assumed to be false.

In many realistic application domains, such as robotics, the assumption of complete information is not feasible. Therefore, the CWA can not be used. The embedding environment is simply too dynamic and information about it too sparse. For example, a UAV flying over a region cannot have a complete model of that region. New objects are continually sensed or encountered and agents other than the UAV agent cause change in the region. In applications such as this, an *Open World Assumption* (OWA), where information not known by the agent is assumed to be unknown, is the ontologically right choice to make, but complicates both the representational and implementational aspects associated with inference mechanisms and the use of negative information.

The CWA and the OWA represent two extremes. Quite often, a reasoning agent has information which permits the application of the CWA *locally* (see [ETZ 97, DOH 00c, DOH 03b]). For example, if an agent has a camera sensor, the agent can assume complete information about objects in the focus of attention (FOA) of the camera; e.g., the only objects in the FOA are those identified by the image processing module. This method of reasoning is called the Local Closed World Assumption (abbreviated as LCWA).

The OWA is used with approximate databases. In this case, both positive and negative information must be stored in the database. Positive and negative information about relations is given via extensional and intensional database layers. Positive and negative facts that do not follow from these layers are assumed to be unknown. We also provide a machinery of *contextually closed queries* (CCQ, introduced in [DOH 03b]) for reasoning based on LCWA, which will be described more precisely in Section 3.4.

2.4. *Second-order Quantifier Elimination*

Many concepts associated with reasoning in the context of incomplete information, such as the local closed-world assumption, approximate representations of logical theories, or circumscription axioms, can often be specified in a succinct and efficient manner using second-order logic. It is also sometimes the case that such second-order formulas can be reduced to logically equivalent first-order or fixpoint formulas using second-order quantifier elimination techniques, such as the DLS [DOH 97] and DLS* [DOH 96a] algorithms. The DLS algorithm is an extension of the algorithm in [SZA 93] and is based on Ackermann's lemma [ACK 35]. The DLS* algorithm makes use of a fixpoint elimination theorem in [NON 98] (Theorem 4 below) and strengthens the DLS algorithm. For an early implementation of the DLS algorithm see [GUS 96] and for newer generation implementations of the DLS and DLS*algorithms, see [MAG 05b].

Although it is clear that there is no algorithm for reducing any arbitrary second-order formula to a logically equivalent first-order formula, when the application of one of the algorithms described above is successful, the result may be a formula of first-order logic, validity of which (over finite databases) is in PTIME and LOGSPACE (here we apply the DLS algorithm) or a formula of fixpoint logic, validity of which (over finite databases) is in PTIME.[3] In the latter case we apply the DLS* algorithm.

In designing our solutions we always restrict the expressiveness of our knowledge representation fragments to those where second-order quantifier elimination is guaranteed to be in PTIME. Consequently, the methods we propose in this context are guaranteed to be tractable, but still the resulting techniques cover all PTIME computable queries on ordered databases due to the well-known expressiveness of fixpoint queries (see, e.g., [ABI 95]).

Relations generated using fixpoint computations are one important aspect of approximate database use. In order to formulate the fixpoint theorem of [NON 98] (Theorem 4 below), which is used in our semantic theory for approximate databases, we require the following definitions.

DEFINITION 3. — *A relation symbol R is said to* occur positively *(respectively* negatively*) in a formula A if it appears under an even (respectively odd) number of negations.*[4] *A formula A is* positive w.r.t. relation symbol R *iff all occurrences of R in A are positive. A formula A is* negative w.r.t. relation symbol R *iff all occurrences of R in A are negative. If $B(X)$ is a second-order formula, where X is a k-argument relational variable and $C(\bar{x})$ is a first-order formula with free variables $\bar{x} = \langle x_1, \ldots, x_k \rangle$. Then by $B[X(\bar{t}) := C(\bar{x})]$ we mean the formula obtained from $B(X)$ by substituting*

3. Recall that fixpoint logic captures all problems solvable in deterministic polynomial time, provided that the underlying domain is linearly ordered — see, e.g., [ABI 95, IMM 98, EBB 95].

4. It is assumed here that all implications of the form $p \rightarrow q$ are substituted by $\neg p \vee q$ and all equivalences of the form $p \equiv q$ are substituted by $(\neg p \vee q) \wedge (\neg q \vee p)$.

each occurrence of X of the form $X(\bar{t})$ in $B(X)$ by $C(\bar{t})$, renaming any newly bound variables in $C(\bar{x})$ with fresh variables.

The fixpoint theorem of [NON 98] is formulated below.

THEOREM 4. — Assume that formula A is a first-order formula positive w.r.t. X.

– If B is a first-order formula negative w.r.t. X then

$$\exists X \left\{\forall \bar{y}\Big[A(X) \rightarrow X(\bar{y})\Big] \wedge B(X)\right\} \equiv B\Big[X(\bar{t}) := \mathtt{lfp} X(\bar{y})\,[A(X)[\bar{t}]]\Big]. \quad (1)$$

– If B is a first-order formula positive w.r.t. X then

$$\exists X \left\{\forall \bar{y}\Big[X(\bar{y}) \rightarrow A(X)\Big] \wedge B(X)\right\} \equiv B\Big[X(\bar{t}) := \mathtt{gfp} X(\bar{y})\,[A(X)[\bar{t}]]\Big]. \quad (2)$$

A survey of various approaches to second-order quantifier elimination is given in [NON 99].

2.5. *Three-valued Logic of Kleene*

The semantics of approximate databases is based on the *strong* three-valued logic of Kleene, denoted by K_3.[5] In K_3 we use three logical values

$$\{\text{TRUE}, \text{FALSE}, \text{UNKNOWN}\},$$

with the following logical truth ordering: $\text{FALSE} \leq \text{UNKNOWN} \leq \text{TRUE}$.

We define the set $\{\text{UNKNOWN}, \text{TRUE}\}$ to be the set of designated truth values.[6] Below we shall briefly discuss propositional, first-order, fixpoint and second-order three-valued logic, reflecting the languages used in AKDBs.

2.5.1. *The Propositional Case*

In the propositional version of K_3 we build formulas based on a set P_3 of three-valued propositional variables using the standard recursive applications of the propositional connectives $\vee, \wedge, \rightarrow$.

Any valuation $v : P_3 \longrightarrow \{\text{TRUE}, \text{FALSE}, \text{UNKNOWN}\}$ is extended to the set of all formulas as follows:

– $$v(\neg A) \overset{\text{def}}{=} \begin{cases} \text{TRUE} & \text{when } v(A) = \text{FALSE} \\ \text{FALSE} & \text{when } v(A) = \text{TRUE} \\ \text{UNKNOWN} & \text{when } v(A) = \text{UNKNOWN} \end{cases}$$

5. This makes our solution, among others, compatible with standard SQL, where K_3 is used to define the semantics of queries when null values can appear in database relations.
6. Recall that in the area of many-valued logics, members of the set of designated truth values act as being true and are used in the definition of entailment.

– $v(A \vee B) \overset{\text{def}}{=} \max\{v(A), v(B)\}$

– $v(A \wedge B) \overset{\text{def}}{=} \min\{v(A), v(B)\}$

– $v(A \rightarrow B) \overset{\text{def}}{=} \begin{cases} \text{TRUE} & \text{when } v(A) = \text{FALSE or } v(B) = \text{TRUE} \\ \text{FALSE} & \text{when } v(A) = \text{TRUE and } v(B) = \text{FALSE} \\ \text{UNKNOWN} & \text{otherwise.} \end{cases}$

In K_3, as in classical propositional logic, it is the case that:

$$v\big(A \rightarrow (B \vee C)\big) \quad = \quad v\big((A \wedge \neg B) \rightarrow C\big) \tag{3}$$

$$v\big((A \wedge B) \rightarrow C\big) \quad = \quad v\big(A \rightarrow (\neg B \vee C)\big). \tag{4}$$

This is crucial in the definition of expansion (see Definition 13).

The choice of K_3 is also crucial in guaranteeing that the Feferman-Gilmore translation (see Section 2.5.5) is sound and complete wrt the semantics accepted for AKDBS.

REMARK 5. — In the literature, the semantics of deductive databases is sometimes specified using the logic proposed in [FIT 85] (see also [PRZ 90]) as a basis, where $\neg, \vee, \wedge$ are defined as in K_3 and $\rightarrow$ is defined by

$$v(A \rightarrow B) \overset{\text{def}}{=} v(A) \leq v(B)). \tag{5}$$

From our perspective, definition (5) is doubtful. First of all, equations (3) and (4), crucial to our understanding of IDB rules (defined later) are no longer valid within this semantics. Secondly, UNKNOWN intuitively denotes a value which is not yet determined by a robotic agent, but may be determined at a future time. Thus UNKNOWN $\rightarrow$ UNKNOWN, which is TRUE according to (5), should be UNKNOWN, since such an implication, after determining truth values of assumption and conclusion, may equally well appear FALSE or TRUE. Moreover, one sometimes obtains results which we find counterintuitive. For example, consider the formula:

$$\Big[\big(fourWheels(a) \wedge onRoad(a)\big) \rightarrow car(a)\Big] \wedge \neg car(a). \tag{6}$$

Assume that an agent has been able to determine that $v\left(car(a)\right) = $ FALSE and has been unable to determine whether a has four wheels, i.e., $v\left(fourWheels(x)\right) =$ UNKNOWN, or whether a is on the road, i.e., $v\left(onRoad(x)\right) =$ UNKNOWN. The definition provided by formula (5) would evaluate formula (6) to be FALSE. On the other hand, intuitively (6) should not be inconsistent. Note that in K_3, formula (6) is evaluated in this situation to be UNKNOWN. □

2.5.2. *The First-Order Case*

For the first-order language with quantifiers $\exists, \forall$, we define v on atomic formulas and then extend it to deal with quantifiers:

– $v\big(\exists x\, A(x)\big) \overset{\text{def}}{=} \max\limits_{a \in \text{DOM}} \{v\big(A(x := a)\big)\}$

– $v\big(\forall x\, A(x)\big) \overset{\text{def}}{=} \min\limits_{a \in \text{DOM}} \{v\big(A(x := a)\big)\}$.

2.5.3. *The Second-Order Case*

For the second-order language with second-order quantifiers $\exists, \forall$ ranging over relations, we first define v on first-order formulas and then extend it to deal with second-order quantifiers (below R is a three-valued relation over DOM with the same arity as relational variable X):

– $v\big(\exists X\, A(X)\big) \overset{\text{def}}{=} \max\limits_{R}\{v\big(A(X := R)\big)\}$

– $v\big(\forall X\, A(X)\big) \overset{\text{def}}{=} \min\limits_{R}\{v\big(A(X := R)\big)\}.$

2.5.4. *Fixpoint Formulas*

For formulas $A(X(\bar{x}))$ positive wrt X one can define operators

$$\texttt{lfp}X(\bar{x})\,[A(X(\bar{x}))] \text{ and } \texttt{gfp}X(\bar{x})\,[A(X(\bar{x}))]$$

denoting the least and the greatest fixpoint of $A(X(\bar{x}))$ (see, e.g., [ABI 95, EBB 95, IMM 98]). Of course, the terms "least" and "greatest" are to be understood wrt a particular ordering on relations. In AKDB we assume the *truth ordering* $\sqsubseteq$ defined as follows (where $\bar{t}$ can contain constants only):[7]

$$R \sqsubseteq S \overset{\text{def}}{\equiv} \begin{array}{l} \{\bar{t} \mid R(\bar{t}) \equiv \text{TRUE}\} \subseteq \{\bar{t} \mid S(\bar{t}) \equiv \text{TRUE}\} \text{ and} \\ \{\bar{t} \mid R(\bar{t}) \equiv \text{FALSE}\} \supseteq \{\bar{t} \mid S(\bar{t}) \equiv \text{FALSE}\}. \end{array} \tag{7}$$

In the classical two-valued case $\sqsubseteq$ coincides with the standard inclusion $\subseteq$.

Assume the (three-valued) valuation v on first-order formulas is given. It is extended to deal with fixpoint operators as follows:

– $v\big(\texttt{lfp}X(\bar{x})\,[A(X(\bar{x}))]\big) \overset{\text{def}}{=} v\big(A(X := R)\big)$, where R is the least (wrt $\sqsubseteq$) relation such that $A(X := R) = R$

– $v\big(\texttt{gfp}X(\bar{x})\,[A(X(\bar{x}))]\big) \overset{\text{def}}{=} v\big(A(X := R)\big)$, where R is the greatest (wrt $\sqsubseteq$) relation such that $A(X := R) = R$.

2.5.5. *The Feferman-Gilmore Translation*

In order to provide a semantics for approximate databases, we shall require the definition of the Feferman-Gilmore translation (see [FEF 84, GIL 74]), used in translating three-valued logic formulas into the classical two-valued logic.

DEFINITION 6. — *By a* Feferman-Gilmore translation of a three-valued formula A, *denoted by* FG(A), *we shall mean the formula obtained from A by replacing all positive literals of the form $R(\bar{y})$ by $R^+(\bar{y})$ and all negative literals of the form $\neg R(\bar{y})$ by $R^-(\bar{y})$.*

7. In the literature one can also find *knowledge ordering*, defined by replacing $\supseteq$ in the second line of (7) by $\subseteq$. Observe that this ordering appears naturally in AKDBs due to the application of the translation defined in Section 2.5.5.

In AKDBs, instead of dealing with a single relation symbol representing a classical relation, say R, we introduce two relation symbols R^+ and R^-, the first one for representing positive facts known about R and the second one for representing negative facts known about R.

The Feferman-Gilmore translation is sound and complete for K_3 in the sense that formula α entails formula β in K_3 iff in the classical logic the following formula is valid:

$$\left[\neg(R_1^+ \wedge R_1^-) \wedge \ldots \wedge \neg(R_k^+ \wedge R_k^-)\right] \to (\alpha \to \beta),$$

where $R_1, \ldots, R_k$ are all atoms in α and β (see, e.g., [BUS 96]).

Observe that whenever the underlying similarity relation is at least serial then, as indicated in Section 2.2, for any relation R approximated by means of similarity neighborhoods we have that $\neg(R^+ \wedge R^-)$ holds.

2.6. *Inconsistencies*

Since we envision using approximate databases in a distributed manner across agent systems and often assume several approximate databases being used in a single agent system, fusing information from various sources will be common place. Consequently, local inconsistencies may arise in AKDBs. Many approaches have been proposed for dealing with such inconsistencies (see, e.g., [BEL 77]). For the purposes of this paper, we do not deal with techniques for resolving inconsistencies, but instead assume this is left to the user of the AKDB, who can provide IDB rules for resolving inconsistencies. This should be considered a feature since there are many strategies for resolving inconsistencies and their choice is often dependent on a particular context. For example one can assume that whenever a tuple, say $\bar{t}$, satisfies both a relation R and its negation $\neg R$, then the result of query $R(\bar{t})$ as well as $\neg R(\bar{t})$ is UNKNOWN. Another approach might depend on prioritizing data sources, prefer answers given by more trustable sources, and rely on less trustable sources only when more trustable provide answer UNKNOWN. Yet another solution might be based on providing weights for data sources and consider the "weighted sum" of answers with thresholds allowing to determine what logical value is actually represented by the computed answer or to apply the majority voting principle. For a discussion of possible strategies see, e.g., [DOH 02, DOH 06].

2.7. *Circumscription*

Circumscription is a powerful non-monotonic formalism introduced in [MCC 80] (for a survey see [LIF 91]). Although circumscription is a second-order formalism, for a rich class of formulas it can be reduced to first-order logic (see [DOH 97]) or to the fixpoint logic using the DLS* algorithm.

If U and V are relation expressions of the same arity, then $U \leq V$ stands for $\forall \bar{x}\ (U(\bar{x}) \rightarrow V(\bar{x}))$.[8] Similarly, if $\bar{U} = \langle U_1, \ldots, U_n \rangle$ and $\bar{V} = \langle V_1, \ldots, V_n \rangle$ are similar tuples of relation expressions, i.e., for $1 \leq i \leq n$, U_i and V_i are of the same arity, then $\bar{U} \leq \bar{V}$ is an abbreviation for $\bigwedge_{i=1}^{n} [U_i \leq V_i]$.

We write $\bar{U} = \bar{V}$ for $(\bar{U} \leq \bar{V}) \wedge (\bar{V} \leq \bar{U})$, and $\bar{U} < \bar{V}$ for $(\bar{U} \leq \bar{V}) \wedge \neg(\bar{V} \leq \bar{U})$.

DEFINITION 7. — *Let $\bar{P} = \langle P_1 \ldots, P_n \rangle$ be a tuple of distinct relation symbols, $\bar{S} = \langle S_1, \ldots, S_m \rangle$ be a tuple of distinct relation symbols disjoint with $\bar{P}$, and let $T(\bar{P}, \bar{S})$ be a theory. The* circumscription *of $\bar{P}$ in $T(\bar{P}, \bar{S})$ with varied $\bar{S}$, written* CIRC$(T; \bar{P}; \bar{S})$*, is the sentence*

$$T(\bar{P}, \bar{S}) \wedge \forall \bar{X} \forall \bar{Y} \left\{ \left[T(\bar{X}, \bar{Y}) \wedge [\bar{X} \leq \bar{P}] \right] \rightarrow [\bar{P} \leq \bar{X}] \right\} \tag{8}$$

where $\bar{X} = \langle X_1 \ldots, X_n \rangle$ and $\bar{Y} = \langle Y_1, \ldots, Y_m \rangle$ are tuples of relation variables similar to $\bar{P}$ and $\bar{S}$, respectively.

3. An Overview of the Architecture of Approximate Knowledge Databases

It is important to facilitate experimentation to provide a better feel for the utility and applicability of the ideas and techniques behind approximate databases, and such experience will likely only come from a real system. We have consequently spent some considerable effort on an actual implementation called the Approximate Knowledge Database (AKDB). Grounded in a standard SQL database, several extension layers each provide an extended query language building upon its lower layers. Figure 1 displays an architectural overview of the system that is described in more detail below, starting from the bottom abstraction layer, moving upwards.

It is important to emphasize that the intended user language is the language of the classical crisp first-order or fixpoint logic. All facts, intensional rules and queries are then translated using the Feferman-Gilmore translation and the results are then translated back according to the semantics provided in Section 4. However, the user can also access the database using the database syntax directly via a graphical user interface (see Appendix A.1) or text files.

3.1. *SQL Database*

Even though the SQL database forms the fundamental layer of the system, it can easily be replaced, choosing from a long list of SQL databases such as POSTGRESQL, MYSQL, ORACLE, SYBASE, INFORMIX, DB2, MS SQL SERVER, etc. Whatever database is chosen, it will use regular, crisp, database relations to store approximate

8. Note that $U \leq V$ means that the extension of U is a subset of the extension of V.

relations, but need not know the details of the representation. Deciding the exact format is the concern of the next abstraction layer, which still leaves the SQL database the task of optimizing and executing SQL queries passed down from higher layers.

Note that for some applications a standard SQL database might not be the most efficient means of storing and retrieving data. In a robotic system a secondary memory footprint may need to be minimized or a high frequency of low complexity queries might call for an implementation where data is stored in primary memory, in which case such an implementation can be plugged into the system, bypassing the SQL interface.

3.2. *Extensional Database*

The Extensional Database layer (EDB) provides a mapping from approximate relations to a specific representation scheme based on regular database relations. In particular, the positive (R^+) and negative (R^-) parts of relations are stored explicitly in tables while the boundary, positive boundary, and negative boundary are only stored implicitly but can still be generated through more complex queries to the database.

When all relation arguments are assumed to have finite domains, all parts of relations consist of a finite number of tuples, and the division between explicit and implicit storage becomes a potential implementation choice point to which the best answer will depend on the final application. We believe that, in the general case, it is profitable to save space by storing known information explicitly and unknown information implicitly since the amount of unknown information about world state is often vastly larger than that which is known.

As mentioned previously, logic is consistently used as the query language in all parts of the AKDB. Both the fact that the logical query language may refer explicitly to the boundary parts that are not explicitly stored and the fact that any, arbitrarily complex, logical formula may be used as a query contribute to the necessity of some kind of evaluation mechanism. To make this feasible, we have developed a query compilation mechanism that will recursively transform any logical formula query into a, sometimes very elaborate, SQL query, the only exception being fixpoint formulas, which need to be iteratively evaluated until a fixpoint is reached before the result can be returned. By delaying the actual evaluation of any part of the query until it reaches the SQL database we can benefit from the potential performance increase resulting from SQL query optimization techniques employed by a particular database of choice.

In order to illustrate the use of the extensional database, let us assume that we have a qualitative approximate relation specifying the colors of cars:

– $Color(x, y)$ (the color of car x is y)

The EDB is populated with some facts, expressed using the $Color$ relation, about three cars, C_1, C_2, and C_3, and two colors, $Black$ and Red, as in (9).

$$Color(C_1, Black) \wedge \neg Color(C_1, Red) \wedge Color(C_2, Red) \tag{9}$$

A simple query asking for the colors of cars, $Color(x, y)$, results in the tuples $\langle C_1, Black\rangle$ and $\langle C_2, Red\rangle$, while $\neg Color(x, y)$ would return $\langle C_1, Red\rangle$. The color of C3 is not known and an explicit query about it, e.g., $Color(C_3, Black)$, results in the value UNKNOWN. This is so because the OWA is being used and no additional constraints restricting a car to have only one color are part of the theory.

More subtle queries can be formed by using the language of approximate relations directly. Asking for cars that *might* not be red, $Color^{\ominus}(x, Red)$, returns the cars $\langle C_1\rangle$ and $\langle C_3\rangle$, since C_1 is known not to be red and C_3 is possibly not. Any complex formulas can be evaluated, e.g., a query asking if there is a color that all cars might have and that some car is known to have can be expressed as $\forall x[Color^{\oplus}(x, y)] \wedge \exists x[Color^{+}(x, y)]$, and returns the tuple $\langle Black\rangle$.

3.3. ***Intensional Database***

The Intensional Database layer (IDB) uses stored rules to infer additional information from the facts in the EDB. The intensional rules are approximate implications with a single literal head, but differ from Horn-clause type rules in that the head literals can contain negations. To deduce new facts, the rules are translated into approximate formula fixpoint queries that must then be checked for consistency since both new positive and new negative information might be produced.

Performing the necessary inferences for a specific query is the task of the inference mechanism, which currently uses the naive method of repeatedly applying all IDB rules until no new facts can be inferred, evaluating the query in this new context, and finally withdrawing the generated facts to restore the original database state. This method can clearly be improved upon using the rich set of techniques developed for efficiently evaluating intensional database queries (see, e.g., Chapter 13 of [ABI 95]).

Continuing the previous example, we might use the IDB to add a rule that enables us, whenever we detect the color of a car, to conclude that it has only that color and not any other color, expressed in (10).

$$\forall x, y_1, y_2[Color(x, y_1) \wedge y_1 \neq y_2 \rightarrow \neg Color(x, y_2)] \tag{10}$$

If we then evaluate the query, $\neg Color(x, y)$, the result will be both $\langle C_1, Red\rangle$ and $\langle C_2, Black\rangle$, which includes the new consequence that C_2 can not be $Black$ since it was known to be Red.

3.4. ***Contextually Closed Query Database***

The Contextually Closed Query database layer (CCQ) provides the functionality that renders locally closed world reasoning feasible. This is accomplished through the use of *circumscription* in the context of a *closure policy* provided by the user.

A contextually closed query, as introduced in [DOH 03b], consists of a logical formula query together with a crisp logical theory, describing the relations in the query, and a closure policy defining what relations are affected by the closure in one of the following ways:

– A relation may be *fixed*, in which case it will not be modified by the closure.

– A relation may be *minimized*, in which case tuples may be moved from the boundary part into the negative part of the relation in order to minimize its extension.

– A relation may be *varied*, in which case tuples may be moved from the boundary part into either the positive or negative part as an effect of the minimization of other relations.

Closure policies reflect the ideas behind circumscription but with local theories or contexts being attached directly to queries to contextualize them (see, e.g., [MCC 80, LIF 91]). Of course, circumscription-based reasoning is not tractable. However, the DLS* algorithm can reduce any second-order formula from the class of semi-Horn formulas into an equivalent first-order formula in polynomial time [DOH 98], where by a *semi-Horn* formula we understand any formula of the form appearing inside of brackets $\{\ldots\}$ in (1) and (2). Consequently, second-order circumscription theories can often be reduced to logically equivalent first-order theories and standard querying techniques can then be used.

Another, important class of closure policies, the universal closure policies, computable in polynomial time, is defined in [DOH 03b]).

The major difference between circumscription and closure policies lays in the underlying methodology. Namely, contextually closed queries focus on preserving the indicated integrity constraints, expressed by means of a first-order theory, that are to be preserved while a given query is being evaluated. The set of chosen integrity constraints can be made dependent on a particular context of a query.

Contextually closed queries are frequently used to model the LCWA (For the connection between LCWAs and circumscription see also [DOH 00c]).

Adding another, undoubtedly approximate, relation to the previous example will help illustrate a CCQ query:

– $Sporty(x)$ (car x is kind of sporty)

Assuming that we want to express the opinion that red cars are sporty, we formulate the simple integrity constraint in (11).

$$\forall x[Color(x, Red) \rightarrow Sporty(x)] \tag{11}$$

If we apply the closure policy of minimizing the new $Sporty$ relation while varying the $Color$ relation, a query such as, $Sporty(x)$, will return the tuple $\langle C_2\rangle$, since C_2 was known to be Red and therefore has to be sporty, while the query $\neg Sporty(x)$ returns the tuples $\langle C_1\rangle$ and $\langle C_3\rangle$ since these cars can be assumed not to be Red in order to minimize $Sporty$.

Running the DLS* algorithm on a circumscribed logical theory can have undesirable effects on its complexity. Although encouraging results in [DOH 99] show that, at least in the semi-Horn case, the size of the syntactic characterizations of varied and minimized relations are linear in relation to the size of the CCQ query, in practice there is often both a need and opportunity for simplifications. A number of equivalence-preserving simplifications are applied to each contextually closed query, e.g. taking advantage of the unique names assumption in the AKDB.

3.5. *Logic Parser*

A parser for database input was generated using the JAVACC parser generator. Our syntax supports the representation of approximate formulas in addition to regular first-order and fixpoint formulas, but also knowledge base-specific constructs such as relation definitions, extensional facts, intensional rules, and logical theories together with closure policies, making it possible to specify entire use scenarios in a single text file. This latter functionality is complementary to the user of the AKDB as a service in a larger system, in which case one would make direct use of the AKDB interface to gain access to methods for, among others, adding and retracting facts or rules.

4. Syntax and Semantics of Approximate Knowledge Databases

4.1. *Syntax of Formulas*

The syntax for crisp first-order formulas is defined in the standard way, assuming that “`->`", “`|`", “`&`", “`-`", “`forall`", “`exists`" denote implication, disjunction, conjunction, negation, universal quantification and existential quantification, respectively.

EXAMPLE 8. — Examples of crisp first-order formulas:

```
Edge(A,B)
forall x,y [exists z [Edge(x,z) & Path(z,y)] -> Path(x,y)]
```
□

Approximate first-order formulas use the same syntax except that atomic formulas have approximation symbols. We will refer to such formulas, assuming that “`+`", “`-`", “`+-`", “`++`" and “`--`" denote the parts of approximate relations $^+, ^-, ^\pm, ^\oplus$ and $^\ominus$, respectively.

Quantifier-free (crisp and approximate) formulas without variables are called *ground formulas*.

EXAMPLE 9. — Examples of approximate first-order formulas:

```
Edge+(A,B)
forall x,y [exists z [Edge+(x,z) & Path+(z,y)] -> Path+(x,y)]
```
□

Fixpoint formulas are like crisp first-order formulas except that there are two additional fixpoint operators, “`lfp`" and “`gfp`", denoting the least and the greatest fixpoint respectively. Combining the approximate first-order formula and fixpoint formula grammar extensions yields the class of approximate fixpoint formulas.

EXAMPLE 10. — Examples of fixpoint and approximate fixpoint formulas:

```
gfp Safe(x) [Bad(x) & forall y [-Edge(x,y) | Safe(y)]]
gfp Safe(x) [Bad--(x) & forall y [Edge--(x,y) | Safe+(y)]]
lfp Tc(x,y) [Edge+(x,y) | exists z [Edge+(x,z) | Tc+(z,y)]]
```
□

Second-order formulas are used to directly access the DLS and DLS*algorithms. They have the same syntax as crisp first-order formulas except for two additional second-order quantifiers, “`forallrelations`" and “`existsrelation`", denoting the universal and existential second-order quantifier respectively.

EXAMPLE 11. — Example of a second-order formula:

```
forallrelations P [
  forallrelations Q [
    forall x [
      -forall y [R(x,y) -> (P(y) | Q(y))] |
      forall z [R(x,z) -> P(z)] |
      forall v [R(x,v) -> Q(v)]
    ]
  ]
]
```
□

4.2. ***The Extensional Database Layer***

4.2.1. *The Language of the Extensional Database Layer*

The extensional database consists of sets of positive and negative facts. We thus assume that the language of the extensional database is a set of ground literals, i.e., formulas of the form $R(\bar{c})$ or $\neg R(\bar{c})$, where R is a relation symbol and $\bar{c}$ is a tuple of constant symbols. In order to store such facts in the EDB we apply the Feferman-Gilmore translation (see Definition 2.5.5). Thus, in the database syntax, these facts are expressed by $\texttt{R+}(\bar{\texttt{c}})$ and $\texttt{R-}(\bar{\texttt{c}})$, respectively.

4.2.2. *Semantics of the Extensional Database Layer*

We give the semantics of relations of the EDB layer. The semantics of formulas can then be extended according to the three-valued logic of Kleene, as defined in Sections 2.5.1-2.5.4.

Consider an atomic ground formula $\texttt{R}(\bar{\texttt{t}})$. Let EDB_c denote the set of facts representing (positive and negative) database facts of a given database instance. Then, in that instance, if $\texttt{R}(\bar{\texttt{t}})$ is an atomic ground formula, then

– $\texttt{R+}(\bar{\texttt{t}}) \overset{\text{def}}{\equiv} \text{TRUE}$ iff $\texttt{R+}(\bar{\texttt{t}}) \in \text{EDB}_c$

– $\texttt{R-}(\bar{\texttt{t}}) \overset{\text{def}}{\equiv} \text{TRUE}$ iff $\texttt{R-}(\bar{\texttt{t}}) \in \text{EDB}_c$.

Based on the above definitions one can define other approximation operators as well as the semantics of the crisp relation:

– `R++(t̄)` $\overset{\text{def}}{\equiv}$ TRUE iff `R-(t̄)` $\not\equiv$ TRUE

– `R--(t̄)` $\overset{\text{def}}{\equiv}$ TRUE iff `R+(t̄)` $\not\equiv$ TRUE

– `R+-(t̄)` $\overset{\text{def}}{\equiv}$ TRUE iff `R+(t̄)` $\not\equiv$ TRUE and `R-(t̄)` $\not\equiv$ TRUE

– `R(t̄)` $\overset{\text{def}}{\equiv} \begin{cases} \text{TRUE} & \text{when } \texttt{R+}(\bar{t}) \equiv \text{TRUE} \\ \text{FALSE} & \text{when } \texttt{R-}(\bar{t}) \equiv \text{TRUE} \\ \text{UNKNOWN} & \text{otherwise.} \end{cases}$

Observe that a literal of EDB can be both TRUE and FALSE, resulting in inconsistencies to be resolved as discussed in Section 2.6.

4.3. *The Intensional Database Layer*

4.3.1. *The Language of the Intensional Database Layer*

The intensional database is intended to infer new facts, both positive and negative via application of intensional rules to the EDB. The rules are of the form

$$\pm P(\bar{x}) \leftarrow \pm P_1(\bar{x}_1), \ldots \pm P_k(\bar{x}_k)$$

where $\pm$ is either the empty string or the negation symbol $\neg$.

The above rule, is translated into the database syntax by applying the Feferman-Gilmore translation. For example, $\neg P(x) \leftarrow Q(x) \wedge \neg R(x,y)$ is translated into `Q+(x) & R-(x,y) -> P-(x).`

REMARK 12. — Note that contrary to DATALOG, we do not require the *safety condition*, i.e., that a variable occurring in the rule's head has to occur in the rule's body. In cases violating the safety condition, we assume that the total relation referring to variables that are in the rule's head and not in its body are added to the rule's body. More precisely, consider the rule

$$R(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_k(\bar{x}_k). \tag{12}$$

Let $\bar{z} \overset{\text{def}}{=} \bar{x} - (\bar{x}_1 \cup \ldots \cup \bar{x}_k)$ be nonempty.[9] Let $T(\bar{z})$ be a new relation symbol, representing the total relation, i.e., for all $\bar{z}$, $T(\bar{z})$ holds. Rule

$$R(\bar{x}) \leftarrow R_1(\bar{x}_1), \ldots, R_k(\bar{x}_k), T(\bar{z}),$$

equivalent to (12), is safe. Note that $T(\bar{z})$ is finite, since the database domain is finite.

The semantics for unsafe rules in AKDBs, is then provided by assuming that the total relation binding the "unsafe" variables is implicitly added to each unsafe rule. □

9. By $\bar{x} \cup \bar{y}$ we shall mean all variables that are in $\bar{x}$ or $\bar{y}$ (after removing duplicates), and by $\bar{x} - \bar{y}$ we shall always mean all variables that are in $\bar{x}$ and not in $\bar{y}$.

4.3.2. *Semantics of the Intensional Database Layer*

Observe that, after the translation, all rules in IDB become Horn rules, as in the case of standard DATALOG. We then use the standard semantics (see, e.g., [ABI 95]), according to which minimal relations satisfying the rules are computed.

The semantics of IDB is now defined similarly as in the case of EDB. Namely, let IDB$_c$ denote the set of facts representing (positive and negative) minimal relations computed according to DATALOG semantics over a given database instance and let $\texttt{R}(\bar{\texttt{t}})$ be an atomic ground formula. Then, in that instance,

– $\texttt{R+}(\bar{\texttt{t}}) \overset{\text{def}}{\equiv}$ TRUE iff $\texttt{R+}(\bar{\texttt{t}}) \in$ IDB$_c$

– $\texttt{R-}(\bar{\texttt{t}}) \overset{\text{def}}{\equiv}$ TRUE iff $\texttt{R-}(\bar{\texttt{t}}) \in$ IDB$_c$.

The semantics of approximation operators $\texttt{R++}(\bar{\texttt{t}}), \texttt{R--}(\bar{\texttt{t}}), \texttt{R+-}(\bar{\texttt{t}})$ as well as the semantics of the crisp relation $\texttt{R}(\bar{\texttt{t}})$ and arbitrary formulas can now be defined by analogy with Section 4.2.2.

Observe that a literal of IDB can be both TRUE and FALSE, resulting in inconsistencies to be resolved as discussed in Section 2.6.

4.4. *Contextually Closed Queries*

4.4.1. *The Language of Contextually Closed Queries*

Contextually closed queries use *local contextual closure policies*, LCC s, which are expressions of the form

$$\text{LCC}\,[L_1, \ldots, L_p; K_1, \ldots, K_r] : I, \tag{13}$$

where $L_1, \ldots, L_p$ are (positive or negative) literals, $K_1, \ldots, K_r$ are relation symbols not appearing in L_i's and I is a set of integrity constraints. Literals $L_1, \ldots, L_p$ are minimized assuming that integrity constraints I are preserved and relations $K_1, \ldots, K_r$ can vary. Thus LCC corresponds to circumscription of $I \cup EDB \cup IDB$ with relations $L_1, \ldots, L_p$ minimized (positive literals) or maximized (negative literals) and $K_1, \ldots, K_r$ allowed to vary. By an LCC *assumption* we mean a minimization or maximization of a single literal from $L_1, \ldots, L_p$ in (13).

A *contextually closed query* consists of the query itself, which can be any fixpoint or first-order formula together with the local closure policy representing the closure context.

4.4.2. *Semantics of Contextually Closed Queries*

Let the EDB and IDB be defined as before, let I denote a finite set of integrity constraints, and let LCC $[\bar{L}; \bar{K}]$:I be a specific LCC policy. Then, if $\texttt{R}(\bar{\texttt{t}})$ is an atomic ground formula, then under the given policy,

– `R+(t̄)` $\stackrel{\text{def}}{\equiv}$ TRUE iff $\text{CIRC}(I \cup \text{IDB} \cup \text{EDB}; \bar{L}; \bar{K}) \models R(\bar{c})$

– `R-(t̄)` $\stackrel{\text{def}}{\equiv}$ TRUE iff $\text{CIRC}(I \cup \text{IDB} \cup \text{EDB}; \bar{L}; \bar{K}) \models \neg R(\bar{c})$.

The semantics of approximation operators `R++(t̄)`, `R--(t̄)`, `R+-(t̄)` as well as the semantics of the crisp relation `R(t̄)` and arbitrary formulas can again be defined by analogy with Section 4.2.2.

4.4.3. *Implementation Issues*

In general, the problem of querying an approximate database containing LCC policies defined using unrestricted logical theories is CO-NPTIME-COMPLETE. When one restricts theories to semi-Horn, the situation becomes tractable, since for such theories DLS* succeeds (see, [DOH 96b]).

However, for the case of *universal* theories a more efficient computation mechanism, not involving DLS* can be provided (see, [DOH 03b]). By a *universal* theory we understand any set of formulas of the form,

$$\forall \overline{y}[(\pm R_1(\overline{x}_1) \wedge \cdots \wedge \pm R_k(\overline{x}_k)) \rightarrow \pm R(\overline{x})] \tag{14}$$

where $\pm$ denotes an optional negation sign, $R_1, \ldots, R_k, R$ are relation symbols, $\overline{y}$ is the vector of all variables occurring in $\overline{x}_1, \ldots, \overline{x}_k, \overline{x}$ and $\overline{x} \subseteq \overline{x}_1 \cup \ldots \cup \overline{x}_k$.

To compute syntactic definitions of the minimized and varied relations we first use the equivalences reflecting laws (3) and (4):

$$\begin{aligned}\forall \overline{x}[A(\overline{R}) \rightarrow (B(\overline{R}) \vee M(\overline{y}))] &\equiv \forall \overline{x}[(A(\overline{R}) \wedge \neg M(\overline{y})) \rightarrow B(\overline{R})]\\ \forall \overline{x}[(A(\overline{R}) \wedge M(\overline{y})) \rightarrow B(\overline{R})] &\equiv \forall \overline{x}[A(\overline{R}) \rightarrow (B(\overline{R}) \vee \neg M(\overline{y}))]\end{aligned} \tag{15}$$

to generate the *expansion* of each formula in the logical theory, where the notion of expansion is defined below.

DEFINITION 13. — *The* expansion *of a universal formula F, wrt a closure policy, is defined as the least set of formulas, obtained by applying tautologies (15) to F, such that any occurrence of a minimized or varied relation is a consequent of one implication.*

To minimize a relation we collect, from the expanded theory, all the antecedents for which the relation must be true and make the relation false in all other cases. This is accomplished by first applying the Feferman-Gilmore translation to the expanded theory and then forming a disjunction of the implications' antecedents where the consequent is the positive part of the relation to be minimized. This serves directly as a definition of the positive part of the new minimized relation, while the negative part is defined by the rough negation of the positive part, where the rough negation of R^+ is $R^{\ominus}$ if R belongs to the set of relations to be minimized or varied given the closure policy being applied and R^- otherwise (and similarly, the rough negation of R^- is $R^{\oplus}$ or R^+). Definitions of the positive and negative parts of varied relations are constructed by collecting antecedents from the implications with corresponding consequents, combining them in disjunctions and finally substituting occurrences of minimized relations by their new definitions obtained in the previous step (see [DOH 03b]).

5. Relation to other Work

In comparison to variants of DATALOG (see [ABI 95, PRZ 90]) we are able to naturally represent approximated concepts and relations. Instead of a single relation, say R, we consider two relations, one responsible for positive information (R^+) and the other (R^-) for negative information; then negative literals are substituted by R^- using the Feferman-Gilmore translation. We obtain negation-free IDB rules and apply the standard DATALOG semantics for computing R^+ and R^- (with the CWA). However, the relation R itself is represented by its positive and negative parts, which reflects the OWA and can result both in inconsistencies and the lack of full knowledge about R.

Moreover, we permit the use of much more complicated IDB rules than those offered by DATALOG. As mentioned before, we do not require the *safety condition* that a variable occurring in the rule's head has to occur in the rule's body. Secondly, users may supply IDB rules that contain negations in the head literal as well as the body, which are then translated as noted above. Finally, the use of quantifiers is more relaxed as existential quantifiers may be used freely in the rule body.

An approach to rough databases is provided in [VIT 03a, VIT 03b], where a logic programming based implementation of rough databases is given. The strength of these solutions is in applying numerical measures and quantitative reasoning not directly present in AKDBs. On the other hand our approach offers a much greater flexibility with our ability to deal not only with rough set-based reasoning but also with the combination of rough and crisp logic.

Unlike both DATALOG and the rough database approach above, the AKDB introduces the use of restricted circumscriptive policies in the form of contextually closed queries, which allows for highly expressive nonmonotonic queries. We also provide an algorithm for second-order quantifier elimination (DLS*) that has many important applications, e.g. the calculation of weakest sufficient and strongest necessary conditions discussed in Appendix A.3, and that is guaranteed to succeed for the well-defined class of semi-Horn formulas.

6. A Surveillance Mission Case Study

Consider a scenario involving a UAV that makes use of contextually closed queries during a surveillance mission. A black car has been reported stolen and the task of the UAV is to locate the car by investigating areas in which the car is suspected to be located. To represent this scenario we make use of the relations:

– $In(x, y)$ (car x is in region y)
– $Color(x, z)$ (the color of car x is z)
– $SuspectIn(y)$ (the stolen car is suspected to be in region y)
– $Investigate(x, y)$ (the UAV should search for car x in region y).

Using these relations we construct a crisp logical theory (16) expressing the behavior we wish the UAV to exhibit. All black cars that are in a suspect region should be investigated. If a car is known to have some color other than black it is not necessary to look for it in any region. Finally, when we know that the searched car is not in a region, there is no point going there looking for it.

$$\begin{aligned}&\forall x, y\Big[\big(In(x,y) \wedge SuspectIn(y) \wedge Color(x, Black)\big)\\&\qquad\qquad\qquad\qquad\qquad\qquad \rightarrow Investigate(x,y)\Big] \wedge\\&\forall x, y, z\Big[\big(Color(x,z) \wedge z \neq Black\big) \rightarrow \neg Investigate(x,y)\Big] \wedge\\&\forall x, y\Big[\neg In(x,y) \rightarrow \neg Investigate(x,y)\Big]\end{aligned} \tag{16}$$

In AKDB (16) is translated and represented as

```
forall x,y [In+(x,y) & SuspectIn+(y) & Color+(x,Black)
                -> Investigate+(x,y)] &
forall x,y,z [Color+(x,z) & z!=Black -> Investigate-(x,y)] &
forall x,y [In-(x,y) -> Investigate-(x,y)]
```

Additionally an intensional rule (17) is added to the IDB expressing the fact that if we know the region a car is in, it can not simultaneously be in some other region.

$$\forall x, y_1, y_2[\neg In(x, y_2) \leftarrow In(x, y_1) \wedge y_1 \neq y_2] \tag{17}$$

In the database syntax (17) is expressed as

```
forall x,y1,y2 [In+(x,y1) & y1!=y2 -> In-(x,y2)]
```

Continuing the example, we construct a specific scenario by adding facts to the approximate knowledge base. Given three cars, C_1, C_2 and C_3, three regions, R_1, R_2 and R_3, and two colors, $Black$ and Red, we add the facts expressed in (18). A black car C_1 is known to be in region R_1, the car C_2 is red but we do not know in which region it is, and nothing is known about the third car C_3. Furthermore, the stolen car is believed to be located somewhere in region R_1 or R_2.

```
In+(C1,R1) &
Color+(C1,Black) & Color+(C2,Red) &                 (18)
SuspectIn+(R1) & SuspectIn+(R2)
```

The current knowledge base does not contain any information about which cars and what regions are interesting for the UAV, but this is information that would be invaluable when determining appropriate strategies to search regions for target vehicles. To acquire such information, a contextually closed query can be formulated which takes account of current context. In this case, new information specific to regions of interest can be generated nonmonotonically.

To do this, the closure policy associated with the contextually closed query will minimize the number of suspected regions in order to avoid searching regions that we have no specific reason to believe the stolen car to be in, while varying what cars and regions the UAV should investigate to obtain information about possible actions to take. Consequently we construct the policy of minimizing $SuspectIn$ while varying $Investigate$ and fixing the remaining relations In and $Color$.

In order to ask contextually closed queries we first compute the expansion of (16) (recall that expansion is defined in Definition 13), where the only formula containing an occurrence of a minimized or varied relation that is not already in the consequent is the first one. Thus, in this case, the expansion of (16) is obtained by replacing the first conjunct by

$$\begin{aligned}&\forall x, y[In(x, y) \wedge SuspectIn(y) \wedge Color(x, Black)\\ &\qquad\qquad\qquad\rightarrow Investigate(x, y)]\\ &\forall x, y[In(x, y) \wedge \neg Investigate(x, y) \wedge Color(x, Black)\\ &\qquad\qquad\qquad\rightarrow \neg SuspectIn(y)]\end{aligned} \tag{19}$$

We now obtain syntactic definitions for $SuspectIn$ and $Investigate$ according to the description provided in Section 4.4.3 and obtain the results shown in (20). The positive part of the minimized $SuspectIn$ relation is simply those tuples explicitly stored as positive in the extensional database, while the negative part contains the rest of the tuples, while the definition of the varied relation $Investigate$ is more complex.

$SuspectIn$:	`SuspectIn+(y)`
$\neg SuspectIn$:	`SuspectIn--(y)`
$Investigate$:	`Investigate+(x,y) |`
	`In+(x,y) & SuspectIn+(y) & Color+(x,Black)`
$\neg Investigate$:	`Investigate-(x,y) | In-(x,y) |`
	`exists z [Color+(x,z) & z!=Black]`

(20)

Observe that the definitions contain unbound variables, and presenting them to the knowledge base as queries will produce exactly the tuples satisfying the new relation definitions. To evaluate a complex query containing the minimized or varied relations it suffices to replace those occurrences with their syntactic definitions and pass the modified query to the intensional database layer. Evaluating the definitions in our examples produces the tuples in (21), including new tuples produced by the IDB rule.

$$\begin{aligned} In(x,y) :&\quad \langle \mathtt{C1}, \mathtt{R1}\rangle\\ \neg In(x,y) :&\quad \langle \mathtt{C1}, \mathtt{R2}\rangle, \langle \mathtt{C1}, \mathtt{R3}\rangle\\ SuspectIn(y) :&\quad \langle \mathtt{R1}\rangle, \langle \mathtt{R2}\rangle\\ \neg SuspectIn(y) :&\quad \langle \mathtt{R3}\rangle\\ Investigate(x,y) :&\quad \langle \mathtt{C1}, \mathtt{R1}\rangle\\ \neg Investigate(x,y) :&\quad \langle \mathtt{C1}, \mathtt{R2}\rangle, \langle \mathtt{C1}, \mathtt{R3}\rangle, \langle \mathtt{C2}, \mathtt{R1}\rangle,\\ &\quad \langle \mathtt{C2}, \mathtt{R2}\rangle, \langle \mathtt{C2}, \mathtt{R3}\rangle \end{aligned} \tag{21}$$

Although the IDB rule excluded the possibility of C_1 being anywhere else than in R_1, it remains unknown which regions the other cars are in. Minimizing $SuspectIn$

removes R_3 from the set of suspected regions since there is no reason to believe otherwise, while varying $Investigate$ prompts the UAV to search for C_1 in region R_1 since we know it is a black car located in a region which we suspect the stolen car to be in. In addition, the UAV concludes that it is not necessary to look for C_1 anywhere else, using the IDB rule and the part of the theory stating that it should not investigate a region, looking for a car it knows is not there. Car C_2 can be in any of the regions but there is no point looking for it as it has the color Red, different from $Black$. Finally, it remains unknown, even after applying the closure policy, if searching for the car C_3 in any of the regions is necessary.

Now, assume the UAV takes action, flying over region R_1 looking for C_1, and that it finds the car but it is not the stolen car we are looking for. It updates the knowledge base by removing R_1 from the list of suspected regions and adding the fact that, while searching R_1 for C_1, the car C_3 was not encountered, expressed by `In-(C3,R1)`. Using the same syntactic definitions of relations, we reevaluate the queries in light of these new facts.

$$\begin{aligned} In(x,y):&\quad \langle \texttt{C1},\texttt{R1}\rangle \\ \neg In(x,y):&\quad \langle \texttt{C1},\texttt{R2}\rangle, \langle \texttt{C1},\texttt{R3}\rangle, \langle \texttt{C3},\texttt{R1}\rangle \\ SuspectIn(y):&\quad \langle \texttt{R2}\rangle \\ \neg SuspectIn(y):&\quad \langle \texttt{R1}\rangle, \langle \texttt{R3}\rangle \\ Investigate(x,y):&\quad \\ \neg Investigate(x,y):&\quad \langle \texttt{C1},\texttt{R2}\rangle, \langle \texttt{C1},\texttt{R3}\rangle, \langle \texttt{C2},\texttt{R1}\rangle, \\ &\quad \langle \texttt{C2},\texttt{R2}\rangle, \langle \texttt{C2},\texttt{R3}\rangle, \langle \texttt{C3},\texttt{R1}\rangle \end{aligned} \tag{22}$$

The In tuples in (22) changed to incorporate the fact that C_3 has not yet been found, and the R_1 tuple in the $SuspectIn$ relation has moved to reflect the fact that no stolen car was found there, but the varied $Investigate$ relation has changed too. The UAV has already searched region R_1 for C_1, and it concludes that it is no longer necessary to investigate whether C_3 is in R_1, but it is still unknown if the UAV should look for C_3 in one of the other regions.

Notice that without changing the definitions, the query results have changed to reflect the new knowledge situation. This will stay true until we modify the closure policy or the logical theory describing the mission, in which case the definitions must be recalculated. As long as the policy and theory stay the same the AKDB caches the calculated definitions, improving efficiency.

In its current state of uncertainty, the UAV might either explain the two remaining possibilities to a mission operator, asking for new information or advice on which action to take, or continue on itself, e.g. by systematically searching for C_3, first in region R_2 and then in R_3. Assuming the latter alternative, and that the stolen car is in fact located in one of the regions, the UAV will find it and successfully complete the mission.

7. Conclusions

We have developed an experimental environment in which the ideas and techniques summarized in [DOH 06] can be investigated and explored. The system, called the Approximate Knowledge Database, is implemented in JAVA and consists of a layered architecture based on a plug-in SQL database. The AKDB may be used either as a service through an interface, or stand-alone through file input or a graphical user interface, the Graphical Database Design Tool. This environment is intended to offer robust support to users wishing to explore the use of approximate databases for knowledge representation. A number of functionalities, not normally associated with databases are also included such as quantifier elimination algorithms and generators for WSCs and SNCs. The techniques used are tractable, but still allow for advanced types of non-monotonic reasoning, including variants of default logics, limited circumscriptive reasoning as well as reasoning with the local closed world assumption.

The system is highly portable and may be embedded in robotic systems such as the UAV platforms we use in our research. We are currently investigating the use of AKDBs with planning algorithms which deal with incomplete information.

Acknowledgements

This work is partially funded by the Wallenberg Foundation under the WITAS UAV Project, the NFFP03-539 COMPAS Project and the grant 3 T11C 023 29 of the Polish Ministry of Science and Information Society Technologies.

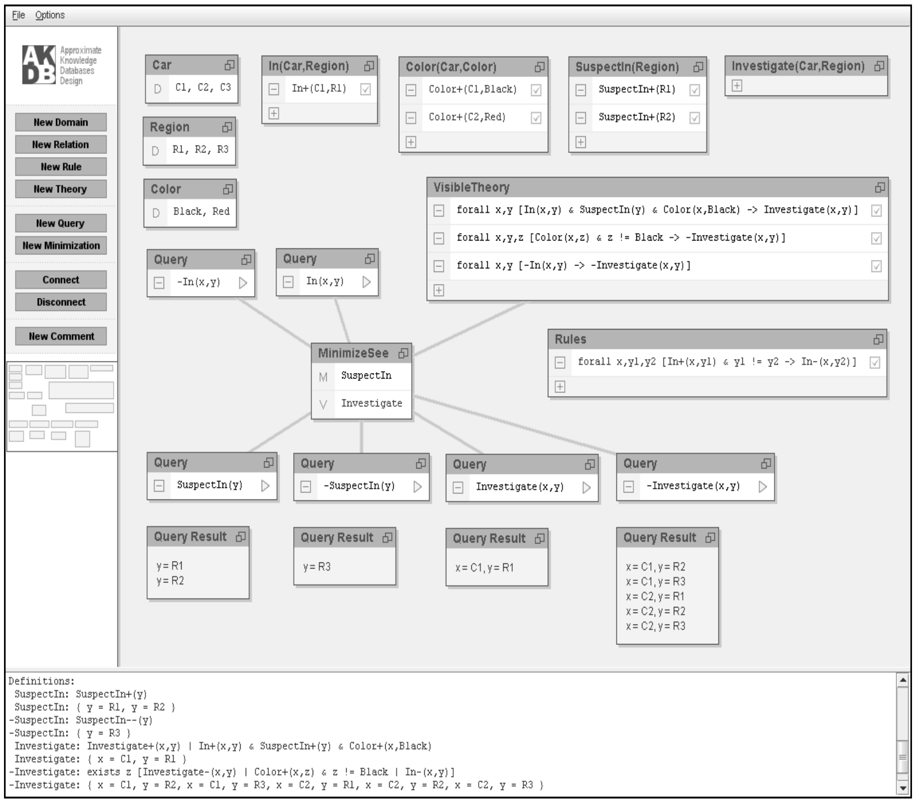
Appendix

A. Additional Functionality

This appendix describes some additional functionality available in the current version of our AKDBpackage.

A.1. *Graphical Database Design*

Even if not a necessary functional part of a system, a graphical user interface can often substantially simplify experimentation. Since we built an experimental platform for a collection of new deductive database techniques not yet extensively explored, such functionality is beneficial. This is the idea behind the Graphical Database Design Tool, shown in Figure 3, for the AKDB that provides an environment where knowledge bases, complete with relation definitions, facts, rules, and theories, can be created, changed, or removed. The interface is built upon a window system, where each relation, theory, policy, or query, has its own window. The windows can then be connected to link a query with a policy and a theory.



VisibleTheory

```
forall x,y [In(x,y) & SuspectIn(y) & Color(x,Black) -> Investigate(x,y)]
forall x,y,z [Color(x,z) & z != Black -> -Investigate(x,y)]
forall x,y [-In(x,y) -> -Investigate(x,y)]
```

Rules

```
forall x,y1,y2 [In+(x,y1) & y1 != y2 -> In-(x,y2)]
```

```
Definitions:
 SuspectIn: SuspectIn+(y)
 SuspectIn: { y = R1, y = R2 }
-SuspectIn: SuspectIn--(y)
-SuspectIn: { y = R3 }
 Investigate: Investigate+(x,y) | In+(x,y) & SuspectIn+(y) & Color+(x,Black)
 Investigate: { x = C1, y = R1 }
-Investigate: exists z [Investigate-(x,y) | Color+(x,z) & z != Black | In-(x,y)]
-Investigate: { x = C1, y = R2, x = C1, y = R3, x = C2, y = R1, x = C2, y = R2, x = C2, y = R3 }
```

Figure 3. *A screenshot of the Graphical Database Design user interface.*

A.2. *DLS**

DLS* is substantial for many solutions offered by AKDBs. It is also possible to call the DLS* algorithm using a www interface (see [MAG 05b]).

A.3. *Weakest Sufficient and Strongest Necessary Conditions*

Weakest sufficient (WSC) and strongest necessary (SNC) conditions have been introduced in [LIN 00] in the framework of the classical propositional logic and then generalized to the first-order case in [DOH 00b]. WSCs and SNCs have many important applications, including[10] building communication interfaces between agents, modularization and information hiding, knowledge compilation and theory approximation, abduction and hypotheses generation, reasoning with reduced data sets.

10. For a deeper discussion see, e.g., [LIN 00, DOH 00b, DOH 06, DOH 04b, DOH 04e].

WSCs and SNCs can be calculated by right-clicking the title bar of a theory window and selecting [`Calculate WSC`] or [`Calculate SNC`], respectively. The background theory of the WSC or SNC will consist of the conjunction of the formulas in the theory window, while the formula to be explained ($\langle$*formula*$\rangle$) is entered into a dialog box, as is a list of relations ($\langle$*rel-sym-list*$\rangle$) that should be projected out of the resulting explanation. The computation method used here has been proposed in [DOH 00b], where WSCs and SNCs are first characterized by means of the second-order formulas and then reduced by applying the DLS*. For a large class of formulas (semi-Horn) reduction of WSCs and SNCs to the first-order or fixpoint logic is guaranteed.

A.4. *Default Rules*

An important functionality depends on providing default rules in the spirit of default logic of Reiter [REI 80] and followers. Unfortunately, the original logic is very complex. To make it tractable, we use the idea of approximations, as initiated in [DOH 02].[11] The idea is to formulate Reiter's default rules $\frac{A:B}{C}$ as intensional rules of the form $\left(A^+ \wedge B^\oplus\right) \to C^+$.

The AKDB engine allows one to deal with such rules. However, in order to facilitate the use of default rules, a part of the graphical interface specialized to deal with defaults is under development.

8. References

[ABI 95] ABITEBOUL S., HULL R., VIANU V., *Foundations of Databases*, Addison-Wesley, 1995.

[ACK 35] ACKERMANN W., "Untersuchungen über das Eliminationsproblem der mathematischen Logik", *Mathematische Annalen*, vol. 110, 1935, p. 390–413.

[BEL 77] BELNAP N., "A useful four-valued logic", EPTEIN G., DUNN J., Eds., *Modern Uses of Many Valued Logic*, Reidel, 1977, p. 8–37.

[BEN 84] VAN BENTHEM J., "Correspondence theory", GABBAY D., GUENTHNER F., Eds., *Handbook of Philosophical Logic*, vol. 2, D. Reidel Pub. Co., 1984, p. 167–247.

[BUS 96] BUSCH D., "Sequent Formalizations of Three-Valued Logics", DOHERTY P., Ed., *Partiality, Modality and Nonmonotonicity*, p. 45–75, CSLI Publications, 1996.

[DOH 96a] DOHERTY P., LUKASZEWICZ W., SZALAS A., "General Domain Circumscription and its First-Order Reduction", *FAPR*, vol. 1085 of *Lecture Notes in Computer Science*, Springer, 1996, p. 93-109.

11. The idea is discussed in depth in [DOH 06] for various variants of default logics, including defaults with strong prerequisites and the prioritized default logic.

[DOH 96b] DOHERTY P., LUKASZEWICZ W., SZALAS A., “A Reduction Result for Circumscribed Semi-Horn Formulas”, *Fundamenta Informaticae*, vol. 28, num. 3-4, 1996, p. 261-271.

[DOH 97] DOHERTY P., LUKASZEWICZ W., SZALAS A., “Computing Circumscription Revisited: A Reduction Algorithm”, *Journal of Automated Reasoning*, vol. 18, num. 3, 1997, p. 297-336.

[DOH 98] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “General Domain Circumscription and its Effective Reductions”, *Fundamenta Informaticae*, vol. 36, num. 1, 1998, p. 23-55.

[DOH 99] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “Declarative PTIME Queries for Relational Databases using Quantifier Elimination”, *Journal of Logic and Computation*, vol. 9, num. 5, 1999, p. 739-761.

[DOH 00a] DOHERTY P., GRANLUND G., KUCHCINSKI K., NORDBERG K., SANDEWALL E., SKARMAN E., WIKLUND J., “The WITAS Unmanned Aerial Vehicle Project”, *Proceedings of the 14th European Conference on Artificial Intelligence*, 2000, p. 747–755.

[DOH 00b] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “Computing Strongest Necessary and Weakest Sufficient Conditions of First-Order Formulas”, *Proceedings of the International Joint Conference on AI (IJCAI’2001)*, 2000, p. 145 – 151.

[DOH 00c] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “Efficient Reasoning using the Local Closed-World Assumption”, CERRI A., DOCHEV D., Eds., *Proc. 9th Int. Conference AIMSA 2000*, vol. 1904 of *LNAI*, Springer-Verlag, 2000, p. 49–58.

[DOH 02] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “CAKE: A Computer-Aided Knowledge Engineering Technique”, VAN HARMELEN F., Ed., *Proc. 15th European Conference on Artificial Intelligence, ECAI’2002*, Amsterdam, 2002, IOS Press, p. 220–224.

[DOH 03a] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “Tolerance spaces and approximative representational structures”, GÜNTER A., KRUSE R., NEUMANN B., Eds., *Proc. 26th German Conf. on AI*, vol. 2821 of *LNAI*, Springer-Verlag, 2003, p. 475–489.

[DOH 03b] DOHERTY P., KACHNIARZ J., SZAŁAS A., “Using Contextually Closed Queries for Local Closed-World Reasoning in Rough Knowledge Databases”, *[PAL 04]*, 2003, p. 219–250.

[DOH 04a] DOHERTY P., “Advanced Research with Autonomous Unmanned Aerial Vehicles”, *Proc. of International Conference on Principles of Knowledge Representation and Reasoning*, 2004, p. 731–732.

[DOH 04b] DOHERTY P., KERTES S., MAGNUSSON M., SZAŁAS A., “Towards a Logical Analysis of Biochemical Pathways”, ALFERES J., LEITE J., Eds., *Proceedings of the 9th Conference on Artificial Intelligence JELIA’2004*, vol. 3229 of *LNAI*, Springer Verlag, 2004, p. 667–679.

[DOH 04c] DOHERTY P., ŁUKASZEWICZ W., SKOWRON A., SZAŁAS A., “Approximation Transducers and Trees: A Technique for Combining Rough and Crisp Knowledge”, *[PAL 04]*, 2004, p. 189–218.

[DOH 04d] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “Approximate Databases and Query Techniques for Agents with Heterogenous Perceptual Capabilities”, *Proc. of the 7th Int. Conf. on Information Fusion, FUSION’2004*, 2004, p. 175–182.

[DOH 04e] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “Approximative Query Techniques for Agents with Heterogeneous Ontologies and Perceptive Capabilities”, DUBOIS D., WELTY C., WILLIAMS M.-A., Eds., *Proc. of the 9th Int. Conf. KR'2004*, AAAI Press, 2004, p. 459–468.

[DOH 04f] DOHERTY P., SZAŁAS A., “On the Correspondence between Approximations and Similarity”, TSUMOTO S., SLOWINSKI R., KOMOROWSKI J., GRZYMALA-BUSSE J., Eds., *Proc. of 4th Int. Conf. RSCTC'2004*, vol. 3066 of *LNAI*, Springer-Verlag, 2004, p. 143–152.

[DOH 05] DOHERTY P., ŁUKASZEWICZ W., SZAŁAS A., “Similarity, Approximations and Vagueness”, SLEZAK D., YAO J., PETERS J., ZIARKO W., HU X., Eds., *Proceedings of the Conference RSFDGrC 2005*, vol. 3641 of *LNAI*, Springer-Verlag, 2005, p. 541–550.

[DOH 06] DOHERTY P., ŁUKASZEWICZ W., SKOWRON A., SZAŁAS A., *Knowledge Engineering Techniques: A Rough Set Approach*, Studies in Fuziness and Soft Computing, Springer Physica Verlag, 2006, To appear.

[EBB 95] EBBINGHAUS H.-D., FLUM J., *Finite Model Theory*, Springer-Verlag, Heidelberg, 1995.

[ETZ 97] ETZIONI O., GOLDEN K., WELD D., “Sound and Efficient Closed-World Reasoning for Planning”, *Artificial Intelligence*, vol. 89, 1997, p. 113-148.

[FEF 84] FEFERMAN S., “Towards useful type-free theories”, *Journal of Symbolic Logic*, vol. 49, 1984, p. 75–111.

[FIT 85] FITTING M., “A Kripke-Kleene Semantics for Logic Programs”, *Journal Logic Programming*, vol. 2, num. 4, 1985, p. 295–312.

[GAB 92] GABBAY D. M., OHLBACH H. J., “Quantifier Elimination in Second-Order Predicate Logic”, NEBEL B., RICH C., SWARTOUT W., Eds., *Principles of Knowledge representation and reasoning, KR 92*, p. 425–435, Morgan Kauffman, 1992.

[GIL 74] GILMORE P., “The consistency of partial set theory without extensionality”, *Axiomatic Set Theory, Proceedings of Symposia in Pure Mathematics 13*, 1974, p. 147–153.

[GUS 96] GUSTAFSSON J., “The DLS Algorithm”, 1996, `http://www.ida.liu.se/labs/kplab/projects/dls/`.

[IMM 98] IMMERMAN N., *Descriptive Complexity*, Springer-Verlag, New York, Berlin, 1998.

[LIF 91] LIFSCHITZ V., “Circumscription”, GABBAY D. M., HOGGER C. J., ROBINSON J. A., Eds., *Handbook of Artificial Intelligence and Logic Programming*, vol. 3, p. 297–352, Oxford University Press, 1991.

[LIN 00] LIN F., “On strongest Necessary and Weakest sufficient Conditions”, COHN A., GIUNCHIGLIA F., SELMAN B., Eds., *Proc. 7th International Conf. on Principles of Knowledge Representation and Reasoning, KR2000*, San Francisco, Ca., 2000, Morgan Kaufmann Pub., Inc., p. 167–175.

[MAG 05a] MAGNUSSON M., “Approximate Knowledge Database Users Manual”, 2005.

[MAG 05b] MAGNUSSON M., “DLS*”, 2005, `http://www.ida.liu.se/labs/kplab/projects/dlsstar/`.

[MCC 80] McCarthy J., “Circumscription: A Form of Non-Monotonic Reasoning”, *Artificial Intelligence Journal*, vol. 13, 1980, p. 27–39.

[NON 98] Nonnengart A., Szalas A., “A Fixpoint Approach to Second-Order Quantifier Elimination with Applications to Correspondence Theory”, Orłowska E., Ed., *Logic at Work: Essays Dedicated to the Memory of Helena Rasiowa*, vol. 24 of *Studies in Fuzziness and Soft Computing*, Springer Physica-Verlag, 1998, p. 307-328.

[NON 99] Nonnengart A., Ohlbach H. J., Szalas A., “Elimination of Predicate Quantifiers”, *Logic, Language and Reasoning. Essays in Honor of Dov Gabbay, Part I*, Kluwer, 1999, p. 159-181.

[PAL 04] Pal S., Polkowski L., Skowron A., Eds., *Rough-Neural Computing: Techniques for Computing with Words*, Cognitive Technologies, Springer–Verlag, Heidelberg, 2004.

[PAW 91] Pawlak Z., *Rough Sets. Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht, 1991.

[PRZ 90] Przymusinski T., Przymusinska H., “Semantic issues in deductive databases and logic programs”, Banerji R., Ed., *Formal Techniques in Articial Intelligence*, North Holland, 1990, p. 321–367.

[REI 78] Reiter R., “On Closed World Databases”, Gallaire H., Minker J., Eds., *Logic and Databases*, 1978, p. 55–76.

[REI 80] Reiter R., “A Logic for Default Reasoning”, *Artificial Intelligence Journal*, vol. 13, 1980, p. 81–132.

[SZA 93] Szalas A., “On the Correspondence Between Modal and Classical Logic: an Automated Approach”, *Journal of Logic and Computation*, vol. 3, 1993, p. 605-620.

[VIT 03a] Vitória A., Damásio C., Małuszyński J., “From rough sets to rough knowledge bases”, *Fundamenta Informaticae*, vol. 57, num. 2-4, 2003, p. 215–246.

[VIT 03b] Vitória A., Damásio C., Małuszyński J., “Query answering for rough knowledge bases”, Wang G., Liu Q., Yao Y., Skowron A., Eds., *Proceedings of 9th Internatinal Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, vol. 2639 of *LNCS*, Springer-Verlag, 2003, p. 197–204.