Towards a Logical Reconstruction of a Theory for Locally Closed Databases

MARC DENECKER, ÁLVARO CORTÉS-CALABUIG,
and MAURICE BRUYNOOGHE
Katholieke Universiteit Leuven
and
OFER ARIELI
The Academic College of Tel-Aviv

The *Closed World Assumption* (CWA) on databases expresses the assumption that an atom not in the database is false. This assumption is applicable only in cases where the database has complete knowledge about the domain of discourse. In this article, we investigate *locally closed* databases, that is: databases that are sound but partially incomplete about their domain. Such databases consist of a standard database instance, augmented with a collection of *Local Closed World Assumptions* (LCWAs). A LCWA is a "local" form of the CWA, expressing that a database relation is complete in a certain area, called a *window of expertise*. In this work, we study locally closed databases both from a knowledge representation and from a computational perspective. At the representation level, the approach taken in this article distinguishes between the data that is conveyed by a database and the metaknowledge about the area in which the data is complete. We study the semantics of the LCWA's and relate it to several knowledge representation formalisms. At the reasoning level, we study the complexity of, and algorithms for two basic reasoning tasks: computing *certain* and *possible* answers to queries and determining whether a database has complete knowledge on a query. As the complexity of these tasks is unacceptably high, we develop efficient *approximate* methods for query answering. We also prove that for useful classes of queries and locally closed databases, these methods are *optimal*, and thus they solve the original query in a tractable way. As a result, we obtain classes of queries and locally closed databases for which query answering is tractable.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Systems—*Query processing*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods; F.2.2

Á. Cortés-Calabuig is currently with the Department of Mathematics and Computer Science, University of Antwerp, Belgium.
The research was supported by FWO Vlaanderen. Á. Cortés-Calabuig was additionally funded by a doctoral scholarship awarded by the International Office of K.U. Leuven.
Authors' addresses: M. Denecker, Á. Cortés-Calabuig (corresponding author), M. Bruynooghe, Department of Computer Science, Katholieke Universiteit Leuven, Belgium; email: alvaro.cortes@cs.kuleuven.be; O. Arieli, Department of Computer Science, The Academic College of Tel-Aviv, Israel; email: oarieli@mta.ac.il.

DOI 10.1145/1806907.1806914 http://doi.acm.org/10.1145/1806907.1806914

[**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic

General Terms: Theory

Additional Key Words and Phrases: Databases, closed world assumption, locally closed databases

ACM Reference Format:

Denecker, M., Cortés-Calabuig, Á., Bruynooghe, M., and Arieli, O. 2010. Towards a logical reconstruction of a theory for locally closed databases. ACM Trans. Datab. Syst., 35, 3, Article 22 (July 2010), 60 pages. DOI = 10.1145/1806907.1806914 http://doi.acm.org/10.1145/1806907.1806914

1. INTRODUCTION

In database theory, it is common to assume that any atomic fact that does not appear in the database instance is false. This approach follows Reiter's Closed World Assumption (CWA) [Reiter 1982], that presupposes a complete knowledge about the database's domain of discourse.

Databases, however, are not always complete.[1] Incompleteness in relational databases has been investigated almost since their inception in the seventies. The general problem of representing incompleteness in a relational database is already discussed by Imielinsky and Lipski [1981] and in Abiteboul and Grahne [1985] and Grahne [1984, 1989]. Reiter [1986] provides an early semantic characterization of databases containing null values and a sound algorithm for querying such databases. At the representation level, Motro [1989] is perhaps the first to study databases that are partially complete. Incompleteness is important in the context of stand-alone databases but it is truly inherent in settings where data is stored in a distributed fashion since there, each separate data source stores, per definition, only part of the data. Not surprisingly, dealing with incomplete information has a recognized central role in important database fields such as data exchange and integration. Grahne and Mendelzon [1999] provide a framework for dealing with incompleteness in mediator-based systems, based on tableaux techniques for query answering. In Grahne [2002], the general problem of incompleteness of databases from a data integration perspective is further explored. Recently, incompleteness issues in data exchange are studied by Libkin [2006].[2]

There are many reasons for the presence of incomplete knowledge in a database, including ignorance about the domain, lack of proper maintenance, incomplete migration, accidental deletion of tuples, the intrinsic nature of database mediator-based systems (see Lenzerini [2002]), and so forth. Unless properly handled, partial information in database systems might lead to erroneous conclusions, as illustrated in the following examples.

Example 1. Consider a database of a computer science (CS) department that stores information about the telephone numbers of the department's members and collaborators (Figure 1).

[1]Nor they are always correct, but we do not address this problem here.

[2]A more detailed discussion on different methods and their relations to our method is given in Section 6.

Telephone	
Name :	*Telephone*
Lien Desmet	6531421
Lien Desmet	0923314
Bart Delvaux	5985625
Tom Demans	5845213

Department	
Name :	*Department*
Bart Delvaux	Computer Science
Lien Desmet	Philosophy
Tom Demans	Computer Science
David Finner	Biology

Fig. 1. A database of contact phone numbers for a CS department.

Car Owners		
Name :	*Model*	*CarID*
Peter Steward	Mercedes 320	Qn-5452
John Smith	Volvo 230	Bx-5242
Mary Clark	BMW 550	Bx-5462

Location	
Name :	*Residence*
Peter Steward	Queens
Mary Clark	Bronx
John Smith	Bronx

Fig. 2. A database of the traffic tax administration system.

Suppose that this database is complete with respect to all CS department members, but possibly incomplete regarding their external collaborators. Thus, appropriate answers for queries such as *Telephone*(Bart Delvaux, 3962836) and *Telephone*(Lien Desmet, 3212445) are "no" and "unknown", respectively. If completeness of the database is taken for granted, however, the answer for both of these queries is "no". Similarly, under the closed world assumption, the answer for the query $\exists x$: *Telephone*(David Finner, x) is "no", but as the database is complete only with respect to the members of the CS department, one cannot exclude the possibility that David Finner does have a phone number, so a more accurate answer in this case should be "unknown".

Example 2. Consider a distributed traffic tax administration system, in which there is one database for each county, maintaining a database of car owners in that county. There is a protocol among the different counties so that when a car owner leaves one county $\mathcal{A}$ to live in another county $\mathcal{B}$, county $\mathcal{A}$ transfers its information to $\mathcal{B}$, but still preserves a record of the car owner and its current status for a certain period of time, to handle all running tax demands. By the nature of this protocol, each database has complete knowledge about all car owners in its county, but in general it has more information than that. Part of the tables of a particular county, say Bronx, is represented in Figure 2.

This database has expertise on car owners of the Bronx. This metaknowledge allows us to derive that all people that are recorded in the relation *Location* as residents of the Bronx are actually *all* the car owners from that county. However, the information about car owners in Queens is not complete hence one cannot draw such conclusions for the residents of Queens.

Examples 1 and 2 illustrate situations in which database information is only *locally* complete, and so applying the CWA is not the right approach. As already shown, it might even lead to some wrong conclusions. The other extreme approach, known as *Open-World Assumption* (OWA) [Abiteboul and Duschka 1998; Grahne 2002] and in which there is no closure at all, is often used for maintaining distributed knowledge, for example, for mediator-based

systems. In this approach, a relational database is considered as a correct but possibly incomplete representation of the domain of discourse. The main weakness of the OWA is that it does not allow users to express locally closed information, and so in the preceding examples, for instance, one cannot state a full knowledge regarding the phone numbers of the CS department members or about the car owners of the Bronx.

In the past, several approaches have been presented to formalize local versions of the closed world assumption in different contexts [Motro 1989; Levy 1996; Etzioni et al. 1997; Doherty et al. 2000]. In this article, we follow the approach in Cortés-Calabuig et al. [2005], which is an extension of Levy's representation [Levy 1996] of locally closed databases and augment the database with *Local Closed World Assumptions* (LCWAs), which are expressions of the following form.

$$\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$$

Here, P is a database predicate and $\Psi[\bar{x}]$ is a first-order formula with free variables in $\bar{x}$. Intuitively, this expression states that the database relation of P is complete for all tuples $\bar{x}$ for which $\Psi[\bar{x}]$ holds in the domain of discourse. That is, for such tuples, $P(\bar{x})$ is true in the domain of discourse iff $\bar{x}$ is stored in the database relation for the relation P. The formula Ψ is called a *window of expertise* of the database predicate P. In Example 1, for instance, the assumption that the relation of telephone numbers is complete for all members of the CS department would be specified by

$$\mathcal{LCWA}(\mathit{Telephone}(p, t), \mathit{Department}(p, \mathit{CS})),$$

and in Example 2 the assumption about the car owners in the Bronx would be expressed as

$$\mathcal{LCWA}(\mathit{CarOwners}(p, c, i), \mathit{Location}(p, \mathit{Bronx})).$$

Databases that contain LCWA's in addition to the database instance are called *locally closed* (or locally complete [Cortés-Calabuig et al. 2006]). A basic assumption underlying this type of databases is, just as for integrity constraints, that knowledge about the domain of expertise of a database is often a more permanent form of knowledge than the transient data in the database instance.

In this article, we study locally closed databases both from a representational and a computational perspective. At the declarative level the contribution of this article is twofold.

(1) We present a first-order generalization of Levy's approach [Levy 1996] for representing partial completeness in relational database systems and define its semantics. We show that this allows us to capture both Reiter's CWA and the OWA. We also investigate extensions such as local closed world assumptions in a multisource setting (as in mediator-based systems).
(2) We study the relationship of our notion of LCWA with other nonmonotone formalisms for representing incomplete knowledge. For that purpose, an equivalent representation is given based on second-order circumscription [McCarthy 1990]. This provides the basis to extend the concept of

LCWA from relational to so-called disjunctive databases, and allows us to compare our proposal with other approaches for LCWA, expressed in terms of higher-order languages (see Doherty et al. [2000]). We also compare our approach to other forms of reasoning with incompleteness in knowledge-base systems, such as Levesque's framework for reasoning with first-order knowledge-bases and modal logic queries [Levesque 1982], and logic-programming-based techniques for maintaining deductive databases, such as Gelfond and Lifschitz's answer-set programming [Gelfond and Lifschitz 1991], and Loyer and Straccia's any-world assumption [Loyer and Straccia 2005].

At the reasoning level, we study some basic reasoning tasks in locally closed databases, namely computing certain and possible answers of queries, and determining whether the database has complete information on a query. More specifically, we study the following topics.

(1) We analyze the computational complexity of computing certain and possible answers to queries. Not surprisingly, these problems are in general intractable.
(2) The intractability results provide the motivation for one of the main contribution of this article, which is the development of a set of efficient *approximate methods* for query answering based on three-valued logic. Approximate reasoning has recently emerged in many domains of computational logic. In databases, approximate query answering provide tractable lower and/or upper approximations to queries in deductive databases [Chaudhuri 1993; Chaudhuri and Kolaitis 1994], databases with null values [Libkin 1998], semistructured databases [Grahne and Thomo 2001], and databases integration [Grahne and Mendelzon 1999]. More recently, approximate reasoning has been introduced to description logics in order to keep complexity of query answering over Web ontologies under some threshold [Stuckenschmidt and van Harmelen 2002; Pan and Thomas 2007]. There are three main parts to our work on approximate reasoning methods.
   —First, we describe a polynomial fixpoint procedure to compute a three-valued structure approximating *all* models of a locally closed database. We prove that such a structure can be used to compute overestimations of possible answers and underestimations of certain answers for all queries.
   —Second, we analyze the *precision* of our method. We show that, for practically useful classes of queries and locally closed databases, the approximate structure is *optimal*, and possible and certain answers extracted from it are *exact*. In other words, for these queries and databases, *approximate* and *standard* query answering coincide. An important corollary of this result is that we determine a class of databases and queries for which query answering is tractable after all.
   —A weakness of the preceding method is the cost of the computation of the three-valued database, which, although polynomial, may be unacceptable for large databases. Even worse, when the underlying database is modified, the three-valued structure must be updated as well, making

this method impractical for databases with nonpersistent data. We address this problem by presenting a method in which the (re-)computation of this three-valued database can be avoided. This method takes as input a query for possible or for certain answers and a collection of local closed world assumptions, and *compiles* them into a new query in the form of a fixpoint query that can be run directly against the database. We prove that this compiled query returns the same answers as the naive fixpoint method.

As an alternative technique, we present an algorithm that, for a more restricted class of so called *hierarchically closed databases*, transforms the query with the local closed world assumptions into a standard first-order query. Since these smarter methods compute the same answers as the naive method, it follows that they are optimal exactly where the naive methods are.

(3) By the previous results, we obtain practically useful approximate methods for query answering based on well-known database technology, and indicate a potentially useful class of queries and databases for which the full query answering problem is tractable.

The article is divided into three parts. The first one, Section 2, is related to representational aspects of the LCWA. In particular, this section introduces the syntax and semantics of local closed world assumptions and locally closed databases. The second part of the article, Sections 3–5, is related to computation in the context of locally closed databases. Section 3 studies the computational complexity of query answering and obtains intractability results. Section 4 presents the approximate methods for query answering, and Section 5 analyses their optimality. The last part of the article, Sections 6–8, contains a discussion of related and future work. Further technical details regarding the accuracy analysis in Section 5 are given in the electronic appendix (accessible through the ACM Digital Library) where also an alternative approach for representing the closed world assumption by second-order formulas is given.

Finally, note that this article elaborates our previous work and integrates it in a uniform framework. The local closed world assumption as used in this article was introduced in Cortés-Calabuig et al. [2005], where it is compared to other approaches for handling incompleteness in database systems. Further representation issues and a fixpoint semantics for locally closed databases are presented in Cortés-Calabuig et al. [2006]. The complexity results and the improved algorithms for approximate query answering in hierarchically closed databases are based on those given in Cortés-Calabuig et al. [2007] and in Cortés-Calabuig et al. [2008], where the efficiency and accuracy of these algorithms are analyzed. This issue is investigated in much greater detail in this article.

2. THE LOCAL CLOSED WORLD ASSUMPTION

In this section we formally introduce the notion of local closed world assumption. First, in Section 2.1 we give some preliminaries regarding the logical view

on databases [Reiter 1982] and in Section 2.2 we introduce the closed world assumptions and their logical meaning. This is the basis for introducing in Section 2.3 locally closed databases and their logical meaning. In Section 2.4 we argue that our notion has both the Open World Assumption (OWA) and Reiter's Closed World Assumption (CWA) as special cases. In Section 2.5, we discuss the relationship with Levy's original idea about representing partial completeness in database systems [Levy 1996], and in Section 2.6 we conclude by pointing out some further representation considerations regarding the LCWA.

2.1 Preliminaries on Database Systems

A vocabulary σ consists of a set of predicate symbols $\mathcal{R}(\sigma)$ with associated arity[3] and a possibly infinite set of constant symbols $\mathcal{C}(\sigma)$.[4] We assume that vocabularies contain the predicates $\mathbf{t}$ (*true*), $\mathbf{f}$ (*false*), and $=$ (equality), each of which will be interpreted in the standard way. Atomic formulas in σ are constructed from the predicates in $\mathcal{R}(\sigma)$ over tuples $\bar{t}$ of constants from $\mathcal{C}(\sigma)$ and object variables. First-order formulas over σ are constructed from the atomic formulas, using the standard recursive rules for $\neg$, $\wedge$, $\vee$, $\exists$, and $\forall$. An occurrence of a subformula φ in a formula ψ is called positive (negative) if it occurs in the scope of an even (odd) number of negations. Second-order formulas over σ are constructed likewise, except that they may contain predicate variables X in atomic expressions $X(\bar{t})$ and in quantifier expressions $\forall X : \varphi$ and $\exists X : \varphi$. We denote by $\Psi[\bar{x}]$ that the free variables of the formula Ψ are a subset of $\bar{x}$. A sentence is a formula without free variables.

Given a vocabulary σ (possibly extended with object or predicate variables), a *σ-structure* I (also called a σ-interpretation) consists of a nonempty domain Dom^I, for each constant or object variable[5] $C \in \sigma$ a domain element $C^I \in Dom^I$ and, for each n-ary predicate symbol or variable $P \in \sigma$, a relation $P^I \subseteq (Dom^I)^n$. For an object variable x and domain element d, we denote by $I[x : d]$ the structure identical to I except that it interprets x by d. Likewise, for an n-ary predicate variable X and n-ary relation R, $I[X : R]$ denotes the structure identical to I except that it interprets X by R. We extend this notation to sequences of variables and values and denote by $I[\bar{x} : \bar{d}]$ the structure identical to I except that it interprets each x_i by d_i.

The structure I is *finite* if so is Dom^I and I is a *σ-Herbrand structure* if $Dom^I = \mathcal{C}(\sigma)$ and $C^I = C$, for each $C \in \mathcal{C}(\sigma)$. When the vocabulary is clear from the context, we shall omit the σ sign. In this context, $\mathcal{C}(\sigma)$ is called the *Herbrand universe* of σ (denoted *HU*). The *Herbrand base* of σ is the set $HB(\sigma)$ of *ground* (i.e., variable-free) atomic formulas of σ. When σ is clear from the context, a Herbrand structure is characterized by the subset of $HB(\sigma)$ consisting of true atoms. In this article, we are mostly interested in finite Herbrand structures.

---

[3]We sometimes write P/n for a predicate symbol P with arity n.

[4]Function symbols are not used in this article.

[5]Note that, unlike some other presentations of first-order logic in which structures interpret language symbols and (variable) assignments interpret variables, here a structure may interpret both sorts of symbols.

For a tuple $\bar{t}$ of constants and variables interpreted in the structure I, we define $\bar{t}^I$ as the tuple $(t_1^I, \ldots, t_n^I)$. Setting $\mathbf{f} \leqslant \mathbf{t}$, $\neg\mathbf{f} = \mathbf{t}$ and $\neg\mathbf{t} = \mathbf{f}$, the truth value of a formula φ in structure I, denoted φ^I, is defined recursively as follows.

$$
\begin{aligned}
&P(\bar{t})^I = \mathbf{t} \text{ if } \bar{t}^I \in P^I \text{; otherwise } P(\bar{t})^I = \mathbf{f};\\
&(\psi \wedge \phi)^I = Min_{\leqslant}(\psi^I, \phi^I);\\
&(\psi \vee \phi)^I = Max_{\leqslant}(\psi^I, \phi^I);\\
&(\neg\psi)^I = \neg(\psi^I);\\
&(\forall x : \psi[x])^I = Min_{\leqslant}\{\psi^{I[x:a]} \mid a \in Dom^I\};\\
&(\exists x : \psi[x])^I = Max_{\leqslant}\{\psi^{I[x:a]} \mid a \in Dom^I\};\\
&(\forall X : \psi[X])^I = Min_{\leqslant}\{\psi^{I[X:R]} \mid R \subseteq (Dom^I)^n\};\\
&(\exists X : \psi[X])^I = Max_{\leqslant}\{\psi^{I[X:R]} \mid R \subseteq (Dom^I)^n\}.
\end{aligned}
$$

Clearly, φ^I is well-defined as soon as all its free variables are interpreted in I. Following the usual conventions, we denote by $I \models \varphi$ that $\varphi^I = \mathbf{t}$ and by $\psi \models \varphi$ that $\varphi^I = \mathbf{t}$ for each structure I in which $\psi^I = \mathbf{t}$. By slight abuse of notation, we usually write $(\psi[\bar{d}])^I$ to denote $(\psi[\bar{x}])^{I[\bar{x}:\bar{d}]}$.

Definition 1 (*Database Schema and Instance*). A database schema Σ is a finite set of predicate symbols. A *database instance* D is a Herbrand σ_D-structure for some finite vocabulary σ_D with $\mathcal{C}(\sigma_D) = Dom^D$. We say that D has schema Σ if $\mathcal{R}(\sigma_D) = \Sigma$.

The (finite) domain Dom^D is called the *domain* of the database instance and contains at least all constants in the relations (or *tables*) of D (and usually only those). For some variable-free atomic formula A, we write $A \in D$ to denote that $A = P(\bar{d})$ where $\bar{d} \in P^D$. The database instance D is completely determined by the pair $(Dom^D, \{A \in HB(\sigma_D) \mid A \in D\})$. Thus, in the examples that follows, we will often specify a database instance D by a domain and a set of atoms. If the domain of D is not explicitly mentioned, it consists of the set of constants that appear in D.

Example 3. The database of Example 1 consists of binary relations $Tel(\cdot,\cdot)$ (between people and their telephone number) and $Dept(\cdot,\cdot)$ (between people and the department they belong to). It can be abbreviated as follows.

$$
D = \left\{ \begin{array}{l} Tel(LD, 6531421),\ Tel(LD, 0923314),\ Tel(BD, 5985625),\ Tel(TD, 5845213),\\ Dept(BD, CS),\ Dept(LD, Phil),\ Dept(TD, CS),\ Dept(DF, Bio) \end{array} \right\}
$$

Thus, $Dom^D = \{LD, BD, TD, CS, Phil, DF, Bio, 6531421, 0923314, 5985625, 5845213\}$ and $Dept^D = \{(BD, CS), (LD, Phil), (TD, CS), (DF, Bio)\}$.

Structures are a convenient way to formalize the *content* of a database: the collection of named objects and tables of the database. Structures are also the standard way to formalize *semantics*, as they serve as a formal description of a *possible state* of the *domain of discourse*. When interpreted in this way, a database instance, which is a structure, represents the *unique* possible state of the real world, the domain of discourse. A database instance thus represents *complete knowledge* of the domain, specifying what objects exist in the domain

of discourse and in what relations they occur. In this article, we will consider databases representing *partial knowledge* on the domain of discourse. In our setting, a database instance still formalizes the content of the database, but cannot any longer be viewed as the (unique) possible state of the world.

An alternative formalization of a database is as a first-order theory consisting of ground atoms, augmented with the *closed world assumption* [Reiter 1978]. This view can be formalized in first-order logic. Before recalling this theory, we first introduce a convenient notation. In what follows, $\bar{t} = \bar{d}$ is a shorthand for $t_1 = d_1 \wedge, \ldots, \wedge t_n = d_n$.

Notation. 1 Let D be a database instance. For any n-ary predicate $P \in \Sigma$ and n-tuple $\bar{t}$ of terms, the formula $\bigvee_{\bar{d} \in P^D}(\bar{t} = \bar{d})$ expresses that $\bar{t}$ belongs to P^D. We shall abbreviate this formula by "$P(\bar{t}) \in D$".

Example 4. In terms of Example 3, "$Dept(x, y) \in D$" abbreviates the following formula.

$$((x = BD) \wedge (y = CS)) \vee ((x = LD) \wedge (y = Phil)) \vee$$
$$((x = TD) \wedge (y = CS)) \vee ((x = DF) \wedge (y = Bio)).$$

Definition 2 ($\mathcal{M}_{st}(D)$). [Abiteboul et al. 1995; Reiter 1982]. The *extended relational theory* of a database instance D is a set $\mathcal{M}_{st}(D)$ of formulas, consisting of the following sentences.

Soundness:	$\bigwedge_{A \in D} A$
Completeness:	$\bigwedge_{P \in \Sigma} \forall \bar{x} : (P(\bar{x}) \supset (P(\bar{x}) \in D))$
Domain Closure Axiom ($\mathsf{DCA}(Dom^D)$):	$\forall x : (\bigvee_{C \in Dom^D} x = C)$
Unique Name Axiom ($\mathsf{UNA}(Dom^D)$):	$\bigwedge_{C \neq C' \in Dom^D} C \neq C'$

The first-order formalization of the CWA consists of the completeness axioms, the Domain Closure Axiom and the Unique Name Axiom. Viewing a database as a structure or as a theory under the CWA is equivalent, as follows from the following well-known proposition.

PROPOSITION 1. *Each model of $\mathcal{M}_{st}(D)$ is isomorphic to D.*

A database query denotes in each database instance a relation, called its answer. It is natural therefore to define a query as a symbolic set expression $\{\bar{x} \mid \mathcal{Q}\}$ with $\bar{x}$ a tuple of distinct variables and $\mathcal{Q}$ a first-order formula with free variables among $\bar{x}$. A query will be denoted more compactly (and more conventionally) as $\mathcal{Q}(\bar{x})$. The value, also called the answer $\{\bar{x} \mid \mathcal{Q}\}^I$ of the query in structure or database I, is the relation $\{\bar{d} \mid I[\bar{x} : \bar{d}] \models \mathcal{Q}\}$. Note that variables x_i not occurring in $\mathcal{Q}$ are unconstrained and assume arbitrary values in the query answer. We always assume that all constants in this query are contained in the domain Dom^D of the database instance. It follows from the previous proposition that for each query $\mathcal{Q}(\bar{x})$ and ground terms $\bar{t}$,

$$D \models \mathcal{Q}(\bar{t}) \text{ iff } \mathcal{M}_{st}(D) \models \mathcal{Q}(\bar{t}). \tag{1}$$

The assumption of a finite domain Dom^D that is given and fixed for the database does not hold in reality. We come back to this issue in Section 7.

2.2 Local Closed World Assumptions

We now introduce the syntax and semantics of local closed world assumptions.

Definition 3 (*Local Closed World Assumption (LCWA)*). A *local closed world assumption (LCWA)* over a vocabulary σ_D is an expression of the form

$$\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}]),$$

where $P \in \Sigma$ is a predicate symbol, called the LCWA's *object* and $\Psi[\bar{x}]$, called the LCWA's *window of expertise*, is a first-order formula over σ_D with free variables among $\bar{x}$.

The intuitive reading of the expression in Definition 3 is the following: "for all objects $\bar{x}$ such that $\Psi[\bar{x}]$ holds in the *real world*, if an atom of the form $P(\bar{x})$ is true in the real world, then $P(\bar{x})$ occurs *in the database*". Note that in $P(\bar{x})$ the values of the variables $\bar{x}$ are constrained by Ψ. For this reason we call Ψ a *window of expertise* of the predicate P.

Example 5. Some local closed world assumptions for the database of Example 3.

(1) The expression $\mathcal{LCWA}(Dept(x, d)), d = CS)$ states that the database contains all members of Computer Science. That is, for every member m of that department, the table $Dept^D$ contains the tuple (C_m, CS) where the constant C_m is the name of m.
(2) The expression $\mathcal{LCWA}(Tel(x, y), Dept(x, CS))$ states that all phone numbers of all members of the Computer Science department are known and occur in the database. That is, for every phone number n owned by a member m of CS, the atom $Tel(C_m, n)$ appears in the database.
(3) $\mathcal{LCWA}(Tel(x, y), x = LD)$ expresses that D contains all phone numbers of Lien Desmet.

Basically, a local closed world assumption states a logical relationship between a database relation and facts of the real world, and hence, its meaning depends on the database. For a given database, this relationship can be expressed by a first-order formula.

Definition 4 (*Semantics of a LCWA*). Let $\theta = \mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$ be a local closed world assumption and D a database over database schema Σ. The *extended relational formula* of θ in D is the formula

$$\mathcal{M}_D(\theta) = \forall \bar{x} : (\Psi[\bar{x}] \supset (P(\bar{x}) \supset (P(\bar{x}) \in D))).$$

A σ_D-structure M is a model of θ iff M is a model $\mathcal{M}_D(\theta)$.

Observe that the extended relational formula of a local closed world assumption contains the subformula $P(\bar{x}) \in D$ that depends on the database. We cannot characterize its meaning in a way that is independent of the content

of the database. Stated differently, a change in the database modifies the extended relational formula of a local closed world assumption. It follows that a local closed world assumption is a *nonmonotonic* construct.

Example 6. Given the expression $\mathcal{LCWA}(Tel(x, y), x = LD)$, the database of Example 3 implies that 016311560 is not a phone number of Lien Desmet, whereas the updated database obtained by adding $Tel(LD, 016311560)$ implies that 016311560 is a phone number of Lien Desmet.

Example 7. Applying Definition 4 on the LCWA of Item (2) of Example 5, one obtains

$$\begin{aligned}\mathcal{M}_D(\theta) \;=\; & \forall x : \forall y : (Dept(x, CS) \supset (Tel(x, y) \supset \\ & ((x = LD \wedge y = 6531421) \;\vee\; (x = LD \wedge y = 0923314) \vee \\ & (x = BD \wedge y = 5985625) \;\vee\; (x = TD \wedge y = 5845213)))).\end{aligned}$$

Since $Dept(LD, CS)$ does not hold, this simplifies to the following formula.

$$\begin{aligned}\forall x : \forall y : (Dept(x, CS) \;\supset\; & (Tel(x, y) \supset (x = BD \wedge y = 5985625) \\ & \vee\; (x = TD \wedge y = 5845213)))\end{aligned}$$

Two extreme cases of local closed world assumptions are the following.

—LCWA with window of expertise that contains all tuples of the domain: $\mathcal{LCWA}(P(\bar{x}), \mathbf{t})$. This LCWA expresses that when $P(\bar{x})$ is true in the real world, it belongs to the database. In other words, D in this case has complete knowledge on P.

—LCWA with empty window of expertise: $\mathcal{LCWA}(P(\bar{x}), \mathbf{f})$. This LCWA does not express any closure. In fact, $\mathcal{M}_D(\mathcal{LCWA}(P(\bar{x}), \mathbf{f}))$ is tautologically true.

A useful *modularity* property of the local closed world assumption is that a finite set of LCWA expressions for the same object predicate can be composed into one (disjunctive) LCWA expression for the same predicate. Conversely, one may split an LCWA with a disjunctive window of expertise over the disjunction and preserve equivalence. This is expressed in the following proposition.

PROPOSITION 2. *Let* $\theta = \mathcal{LCWA}(P(\bar{x}), \bigvee_{i=1}^{n} \Psi_i[\bar{x}])$ *and* $\theta_i = \mathcal{LCWA}(P(\bar{x}), \Psi_i[\bar{x}])$, $i = 1, \ldots, n$. *Then* $\{\theta\}$ *and* $\{\theta_1, \ldots, \theta_n\}$ *are equivalent in every database* D, *that is,* $\mathcal{M}_D(\theta)$ *and* $\bigwedge_{i=1}^{n} \mathcal{M}_D(\theta_i)$ *are equivalent for every* D.

PROOF. Indeed,

$$\begin{aligned}\bigwedge_{j=1}^{n} \mathcal{M}_D(\theta_j) &\equiv \bigwedge_{j=1}^{n} \forall \bar{x} : (\Psi_j[\bar{x}] \supset (P(\bar{x}) \supset (P(\bar{x}) \in P^D))) \\ &\equiv \forall \bar{x} : \left(\bigvee_{j=1}^{n} \Psi_j[\bar{x}] \supset (P(\bar{x}) \supset (P(\bar{x}) \in P^D)) \right) \equiv \mathcal{M}_D(\theta). \quad \square\end{aligned}$$

We therefore assume without a loss of generality that each predicate symbol in Σ is the object of exactly *one* LCWA expression (possibly $\mathcal{LCWA}(P[\bar{x}], \mathbf{f})$). For

a predicate P, Ψ_P will denote the window of expertise of P in this combined LCWA.

2.3 Locally Closed Databases

Definition 5 (*Locally Closed Database*). A *locally closed database* $\mathfrak{D}$ with schema Σ is a pair $(D, \mathcal{L})$ of a database instance D with schema Σ and a finite set $\mathcal{L}$ of local closed world assumptions, such that Dom^D, the domain of D, contains all constants in $\mathcal{L}$. We denote $\sigma_{\mathfrak{D}} = \sigma_D$ and $Dom^{\mathfrak{D}} = Dom^D$.

Thus, a locally closed database contains a standard database instance but this structure will not be the only model of the database.

Definition 6 (*Base Predicates*). The *base predicates* of a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ are $\mathbf{t}, \mathbf{f}$, and all predicates P such that $\mathcal{L}$ contains the local closed world assumption $\mathcal{LCWA}(P(\bar{x}), \mathbf{t})$.

In the examples that follow, we will often use domain-independent predicates such as $=$ and arithmetical relations $<, >, \leq, \geq$. For the sake of simplicity, we assume here that these predicates are *base* predicates of the database. Thus, for each one of these predicates, $\mathcal{L}$ contains the local closed world assumption expressing complete knowledge on it, and D contains a (finite) database relation representing the (projection of the) predicate in Dom^D. In particular, D contains the identity relation on Dom^D as a relation for $=$, and the projections of the standard natural number relations onto the set of natural numbers in the domain of D. For instance, $<^D$ is $\{(n, m) \mid n, m \in Dom^D \cap \mathbb{N} \wedge n < m\}$. This approach is not complication-free. Certain queries may have unintended answers when solved with respect to these projected relations. For instance, the query $x < 100$ would return the numbers less than hundred that occur in Dom^D, which is potentially different than the numbers less than 100. In Section 7, we investigate additional conditions on queries and Dom^D that guarantee correctness.

A model M of a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ is a Herbrand $\sigma_{\mathfrak{D}}$-structure satisfying each atom $A \in D$ and each $\theta \in \mathcal{L}$. To formalize the semantics of a locally closed database $\mathfrak{D}$ in a way similar to the semantics of standard databases as given in Definition 2, we replace the database completeness axioms by the extended relational theory of the local closed world axioms.

Definition 7 ($\mathcal{M}(\mathfrak{D})$). Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over Σ. The *extended relational theory* of $\mathfrak{D}$ is a set $\mathcal{M}(\mathfrak{D})$ of formulas, consisting of the axioms for the built-in predicates and the following sentences.

Soundness:	$\bigwedge_{A \in D} A$
Local Completeness:	$\bigwedge_{\theta \in \mathcal{L}} \mathcal{M}_D(\theta)$.
Domain Closure Axiom ($\mathsf{DCA}(Dom^{\mathfrak{D}})$):	$\forall x : (\bigvee_{C \in Dom^{\mathfrak{D}}} x = C)$
Unique Name Axiom ($\mathsf{UNA}(Dom^{\mathfrak{D}})$):	$\bigwedge_{C \neq C' \in Dom^{\mathfrak{D}}} C \neq C'$

We denote by $M \models \mathfrak{D}$ that M is a model of $\mathfrak{D}$. If every model of $\mathfrak{D}$ is also a model of a formula φ, we say that $\mathfrak{D}$ *entails* φ (or φ *follows* from $\mathfrak{D}$), and denote this by $\mathfrak{D} \models \varphi$. In general, the theory $\mathcal{M}(\mathfrak{D})$ of $\mathfrak{D}$ expresses incomplete

knowledge about the real world. Thus, in general, it has several (nonisomorphic) models (and the actual world corresponds to one of those models).

We now discuss the basic semantic properties of locally closed databases. The first proposition shows that such a database is equivalent to its extended relational theory.

PROPOSITION 3. *Each model of $\mathcal{M}(\mathfrak{D})$ isomorphic to a model of $\mathfrak{D}$ and vice versa each model of $\mathfrak{D}$ is a model of $\mathcal{M}(\mathfrak{D})$.*

PROOF. By the fact that $\mathcal{M}(\mathfrak{D})$ contains $\mathsf{UNA}(Dom^{\mathfrak{D}})$ and $\mathsf{DCA}(Dom^{\mathfrak{D}})$. □

We therefore have the following generalization, for locally closed databases, of the formula (1).

$$\mathfrak{D} \models \mathcal{Q}(\bar{t}) \text{ iff } \mathcal{M}(\mathfrak{D}) \models \mathcal{Q}(\bar{t}) \tag{2}$$

Since the number of Herbrand models of $\mathcal{M}(\mathfrak{D})$ is finite and all of them can be computed, query answering is decidable. Of course, the naive method of generating all models and computing answers to queries in all models is impractical. In general, the number of models may be very high and is bounded by $2^{|HB(\sigma_{\mathfrak{D}})|}$. Thus, this number may be exponential in the size of $Dom^{\mathfrak{D}}$, the number of relation symbols of Σ, and double exponential in the maximal arity of these symbols. We study the complexity of query answering and provide smarter methods in the following sections.

For standard databases, the structure D was the only model of $\mathcal{M}_{st}(D)$. In the context of locally closed databases, this property is lost but we maintain the following weaker version.

PROPOSITION 4. *A locally closed database $\mathfrak{D} = (D, \mathcal{L})$ is consistent, and D is its least model.*

PROOF. Clearly, D satisfies $\mathfrak{D}$ and since each (Herbrand) model M of $\mathfrak{D}$ satisfies all atoms of D, $D \subseteq M$. □

2.4 Relationship to CWA and OWA

Next we show that our concept of locally closed databases is a generalization of both relational databases with the Open World Assumption (OWA) and of relational databases with Reiter's Closed World Assumption (CWA).

PROPOSITION 5. *A locally closed database $\mathfrak{D} = (D, \emptyset)$ corresponds to the database D under the OWA [Abiteboul and Duschka 1998; Grahne 2002].*

PROOF. In both cases the database is totally incomplete, consisting only of the elements in the database instance. □

We will call this an *open* database. Equivalently, an open database can be represented by the locally closed database $(D, \{\mathcal{LCWA}(P(\bar{x}), \mathbf{f}) \mid P \in \Sigma\})$ containing the LCWA with the empty (false) window of expertise for each predicate in Σ.

PROPOSITION 6. *For a database instance D, let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database in which $\mathcal{L} = \{\mathcal{LCWA}(P(\bar{x}), \mathbf{t}) \mid P \in \Sigma\}$. Then $\mathcal{M}_{st}(D)$ and $\mathcal{M}(\mathfrak{D})$ are equivalent.*

PROOF. By Definitions 2 and 7, as for each predicate P occurring in D, $\mathcal{M}_D(\mathcal{LCWA}(P(\overline{x}), \mathbf{t}))$ is equivalent to the database completeness assumption regarding P. □

2.5 Relation to Levy's Approach

We now establish the relationship between our approach and Levy's original idea about representing partial completeness in database systems [Levy 1996]. In Levy's terminology, a locally closed database is called a *partial database*. Just like in our case, it consists of a database instance and a set of local closed world assumptions, called *local completeness expressions*. To formalize the semantics of such an expression, Levy distinguishes between two sets of relations, *virtual* and *available*. The virtual relations R represent the predicates in the real world, while the available relations R' represent the database tables. Local closed world assumptions are represented in a database notation by expressions of the form $LC(R', R, C)$, where R' and R are respectively available and virtual relations, and C, the *constraint* in Levy's terminology, is a conjunction of virtual atoms not including R with free variables $\bar{x}$ and $\bar{y}$ where $\bar{x}$ is a tuple of variables standing for R's attributes participating in the constraint and $\bar{y}$ a tuple of variables standing for the attributes of the other relations participating in the constraint.

Using Levy's original example, a statement that a database about movies is complete for all the movies after 1965 is represented by

$$LC(Movie', Movie, year \geqslant 1965).$$

Here, *Movie* is a database predicate with attributes (*Title*, *Director*, *Year*) and *year* is the variable standing for the attribute *Year*. Translated in the logical notation of this article, this yields

$$\mathcal{LCWA}(Movie(t, d, y), y \geqslant 1965).$$

Interpreting Levy's semantics for partial databases in the setting of this article, he defines a model of a partial database D with Local Completeness Expressions (LCE) $LC(R', R, C(\bar{x}, \bar{y}))$ as any structure M with the domain of D such that for each virtual predicate R, $R^D = (R')^M \subseteq R^M$ and $M \models \forall \bar{x}\bar{z} : (\exists \bar{y} : C(\bar{x}, \bar{y}) \supset (R(\bar{x}, \bar{z}) \supset R'(\bar{x}, \bar{z})))$. Thus, such a model represents both a possible state of the world (in the virtual predicates) and an image of the database (in the available predicates). Hence, the extended relational theory of a partial database $\mathfrak{D} = (D, \mathcal{L})$ with $\mathcal{L}$ a set of LCE's can be formalized by the following theory $\mathcal{M}'(\mathfrak{D})$.

Definition of available predicates. $\bigwedge_{P \in \Sigma} \forall \bar{x} : (P'(\bar{x}) \leftrightarrow P(\bar{x}) \in D)$

Soundness. $\bigwedge_{P \in \Sigma} \forall \bar{x} : (P'(\bar{x}) \supset P(\bar{x}))$

LCE. $\bigwedge_{LC(P'(\bar{x},\bar{z}), P(\bar{x},\bar{z}), C[\bar{x},\bar{y}]) \in \mathcal{L}} \forall \bar{x}\bar{z} : (\exists \bar{y} : C(\bar{x}, \bar{y}) \supset (P(\bar{x}, \bar{z}) \supset P'(\bar{x}, \bar{z})))$

Domain Closure Axiom. $(\mathsf{DCA}(Dom^D))$: $\forall x : (\bigvee_{C \in Dom^D} x = C)$

Unique Name Axiom. $(\mathsf{UNA}(Dom^D))$: $\bigwedge_{C \neq C' \in Dom^D} C \neq C'$

The preceding lines represent, respectively, the content of the database D by available predicates P', the soundness of the database, the meaning of the local

completeness expressions, and the database domain. Note that, here, by introducing the available predicates representing the database tables, all axioms except the definition of these available predicates are database-independent.

We now show that the semantics of partial databases and of locally closed databases coincides.

Notation 2. Let M be a model and σ a vocabulary; $M|_\sigma$ denotes the projection of M on the symbols of σ.

Definition 8 (*Equivalence of Theories*). Let $\sigma \subseteq \sigma_1 \cap \sigma_2$. We say that theories T_1 over σ_1, and T_2 over σ_2 are *equivalent in* σ, if for each σ_1-model M of T_1, there is a σ_2-model N of T_2 such that $M|_\sigma = N|_\sigma$, and vice versa.

PROPOSITION 7. *For each locally closed database $\mathfrak{D}$ over Σ, it holds that $\mathcal{M}(\mathfrak{D})$ and $\mathcal{M}'(\mathfrak{D})$ are equivalent in Σ.*

PROOF. By substituting $P(\bar{x}) \in D$ for $P'(\bar{x})$ in the soundness formula (second line) and simplifying the resulting formula, we can obtain $\bigwedge_{A \in D} A$. By substituting $P(\bar{x}) \in D$ for $P'(\bar{x})$ and $\Psi[\bar{x}, \bar{z}]$ for $\exists \bar{y} : C(\bar{x}, \bar{y})$ in the sentences formalizing the LCE's, we obtain $\bigwedge_{\theta \in \mathcal{L}} \mathcal{M}_D(\theta)$. The resulting formula is nothing else than $\mathcal{M}(\mathfrak{D})$ augmented with an explicit definition of the predicates P'. The latter predicates have no further occurrences except in their definitions. It follows that this formula is equivalent in Σ to $\mathcal{M}(\mathfrak{D})$, and so the proposition follows. □

With this syntactic and semantic machinery in order, Levy studies the problem of deciding whether a partial database has sufficient information to be able to solve a query in a complete way, that is: whether the answers for a query computed from the database are exactly those that are true in the real world. In this article we will refer to this problem as deciding *Closed World Information (CWI)* on a query (see Definition 10). Levy tackles this problem by reducing it to the problem of determining independence of queries from updates. Exploiting results on the latter problem, he is able to show decidability of the former for certain subclasses of his formalism.

Apart from the (innocent) difference that we use logical notations rather than Levy's database notations, we will study different tasks, in particular query answering, also when there is no CWI on a query. Moreover, our LCWA's with first-order windows of expertise are more expressive than those in the formalism of Levy and in other work on partial completeness in database systems (see Section 6). Arguably, a complete logical treatment of the problem at hand should go beyond such syntactical restrictions.

2.6 Some Further Representation Considerations

We conclude this section with briefly recalling some extensions that are discussed in more detail in Cortés-Calabuig et al. [2005].

LCWA with multiple objects. LCWA's can share the same window of expertise. One can extend the notion of LCWA with expressions of the form $\theta = \mathcal{LCWA}(\{P_1(\bar{x}_1), \ldots, P_n(\bar{x}_n)\},\ \Psi[\bar{x}])$. Such an expression is a compact way of expressing the set of LCWA's $\mathcal{LCWA}(P_i(\bar{x}_i), \exists \bar{x} \setminus \bar{x}_i : \Psi[\bar{x}])$.

LCWA with several databases. Instead of a single database D, multiple databases $D_1, D_2, \ldots$ can be involved in which case a particular closed world assumption can be with regard to some subset of the databases. That makes it worthwhile to add an extra argument to an LCWA and to have expressions of the form $\theta = \mathcal{LCWA}(\{D_1, \ldots, D_n\}, P(\bar{x}), \Psi[\bar{x}])$. The meaning is a straightforward generalization of the original meaning where $P(\bar{x}) \in D$) is replaced by $P(\bar{x}) \in \bigcup_{i=1}^{n} D_i$).

LCWA and the soundness assumption. A common, implicit assumption about our notion of locally closed databases is that they are sound, that is: database instances do not contain erroneous information. As pointed out in Levy [1996], it is easy to dualize the concept of locally closed databases to the concept of *locally sound databases*. Such an assumption can be elegantly formalized using a formula that resembles the LCWA. Indeed, a local soundness assumption for P under Ψ is obtained from the meaning of $\mathcal{LCWA}(P(\overline{x}), \Psi[\overline{x}])$, that is,

$$\forall \bar{x} : (\Psi[\overline{x}] \supset (P(\overline{x}) \supset (P(\overline{x}) \in P^D)))$$

by switching the subformulas of the right implication

$$\forall \bar{x} : (\Psi[\overline{x}] \supset ((P(\overline{x}) \in P^D) \supset P(\overline{x}))).$$

Unconditional soundness of the database is obtained by replacing Ψ by the propositional constant $\mathbf{t}$. This is equivalent to the soundness formula $\bigwedge_{A \in D} A$ in Definition 7.

Bibliographic note. Definitions 3 and 4 are simplifications of those in Cortés-Calabuig et al. [2005] to LCWA's with a single predicate object. The results in Sections 2.2, 2.3, and 2.4 are either presented in Cortés-Calabuig et al. [2005] or are a simplification of similar results in that paper. The relation between the LCWA and Levy's approach is not discussed elsewhere.

3. QUERY ANSWERING IN LOCALLY CLOSED DATABASES

So far, we have described how to *represent* partial completeness in databases. We now turn to *reasoning* with this kind of databases, that is: query answering in locally closed databases. In this section, we formally define this problem and the related problem of determining complete information on queries. The computational complexity of these problems is investigated with respect to $Dom^{\mathfrak{D}}$, the finite domain of the database. The intractability results that are obtained motivate the work in later sections where efficient approximate methods underestimating certain answers and overestimating possible answers are developed as well as conditions under which these approximate methods produce complete answers.

3.1 Query Answering and Complete Information on Queries

Definition 9 (*Certain and Possible Answers*). Let Γ be a first-order theory over a vocabulary σ, $\mathcal{Q}(\bar{x})$ a query over σ, and $\bar{t}$ a tuple of constants.

—$\bar{t}$ is a *certain answer* in Γ for $\mathcal{Q}(\bar{x})$, if $\Gamma \models \mathcal{Q}(\bar{t})$.
—$\bar{t}$ is a *possible answer* in Γ for $\mathcal{Q}(\bar{x})$, if $\Gamma \cup \{\mathcal{Q}(\bar{t})\}$ is satisfiable (equivalently, if $\Gamma \not\models \neg\mathcal{Q}(\bar{t})$.)

In the sequel, given a theory Γ we denote by $Cert_\Gamma(\mathcal{Q}(\bar{x}))$ the set of certain answers of $\mathcal{Q}(\bar{x})$ in Γ and by $Poss_\Gamma(\mathcal{Q}(\bar{x}))$ the set of possible answers of $\mathcal{Q}(\bar{x})$ in Γ. Where Γ is $\mathcal{M}(\mathfrak{D})$, the extended relational theory of $\mathfrak{D}$, we simply write $Cert_\mathfrak{D}(\mathcal{Q}(\bar{x}))$ and $Poss_\mathfrak{D}(\mathcal{Q}(\bar{x}))$.

The following straightforward proposition shows that base predicates in a locally closed database behave as standard database predicates.

PROPOSITION 8. *Let $\mathcal{Q}(\bar{x})$ be a query containing only base predicates of $\mathfrak{D}$. Then*

$$Cert_\mathfrak{D}(\mathcal{Q}(\bar{x})) = Poss_\mathfrak{D}(\mathcal{Q}(\bar{x})) = \{\bar{d} \mid D \models \mathcal{Q}(\bar{d})\}.$$

Another interesting problem for a query $\mathcal{Q}(\bar{x})$ in a locally closed database $\mathfrak{D}$ is whether $\mathfrak{D}$ has complete knowledge on $\mathcal{Q}(\bar{x})$. The idea of complete information on queries has also been called *Closed World Information* (CWI) on a query, and it was used by Levy [1996] in the context of incomplete databases. In Etzioni et al. [1997], this notion is considered in the context of logical agents. It can be defined as follows.

Definition 10 (*Closed World Information, CWI*). A locally closed database $\mathfrak{D}$ over Σ has *closed world information* on a query $\mathcal{Q}(\bar{x})$ if for each tuple $\bar{t}$ of constants in $Dom^\mathfrak{D}$, either $\mathfrak{D} \models \mathcal{Q}(\bar{t})$ or $\mathfrak{D} \models \neg\mathcal{Q}(\bar{t})$.

Obviously, when $\mathfrak{D}$ has complete information about $\mathcal{Q}(\bar{x})$ then certain and possible answers coincide, that is, $Cert_\mathfrak{D}(\mathcal{Q}(\bar{x})) = Poss_\mathfrak{D}(\mathcal{Q}(\bar{x}))$. Such queries are of practical importance, since there is no uncertainty on their answers.

PROPOSITION 9. *$\mathfrak{D}$ conveys CWI on query $\mathcal{Q}(\bar{x})$ iff*

$$Cert_\mathfrak{D}(\mathcal{Q}(\bar{x})) = Poss_\mathfrak{D}(\mathcal{Q}(\bar{x})) = \{\bar{d} \mid D \models \mathcal{Q}(\bar{d})\}.$$

PROOF. Since D is a model of $\mathfrak{D}$, the following equation holds.

$$\{\bar{d} \mid D \models \mathcal{Q}(\bar{d})\} \subseteq Poss_\mathfrak{D}(\mathcal{Q}(\bar{x})) = Cert_\mathfrak{D}(\mathcal{Q}(\bar{x})) \subseteq \{\bar{d} \mid D \models \mathcal{Q}(\bar{d})\} \quad \square$$

By Proposition 9 it follows that queries with CWI can be answered directly in the database D. This was Levy's motivation to study CWI.

Note 1. Observe that the LCWA and CWI are related concepts that capture different phenomena. The LCWA expresses completeness of a part of a *database table* in a relational database, while the CWI identifies completeness of a *query* posed to the database.

Frequently, LCWA's induce CWI on queries. For example, a locally closed database $\mathfrak{D} = (D, \mathcal{L})$, such that $\mathcal{LCWA}(P(x), x=a) \in \mathcal{L}$, conveys CWI on $P(a)$, no matter what D is. As the following proposition shows, this observation can be generalized.

PROPOSITION 10. *If $\theta = \mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$ is an LCWA of $\mathfrak{D}$ and, for some formula $\Phi[\bar{x}]$, $\mathfrak{D}$ conveys CWI on $\Phi[\bar{x}] \wedge \Psi[\bar{x}]$, then $\mathfrak{D}$ conveys CWI on the queries*

$\Phi[\bar{x}] \wedge \Psi[\bar{x}] \wedge (\neg)P(\bar{x})$*, where* $(\neg)P(\bar{x})$ *denotes the positive or negative literal containing* $P(\bar{x})$*.*

PROOF. Since we have CWI on the first part of the formula, it holds that for each tuple $\bar{t}$ of $Dom^{\mathfrak{D}}$, the sentence $\Phi[\bar{t}] \wedge \Psi[\bar{t}]$ is either false in all models of $\mathfrak{D}$ or true in all of them. In the first case, $\Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t})$ is also false in all models of $\mathfrak{D}$ and $\mathfrak{D} \models \neg(\Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t}))$, so CWI holds in this case. In the second case, $\Psi[\bar{t}]$, the window of expertise is true and we have complete information about $P(\bar{t})$ and either $\mathfrak{D} \models P(\bar{t})$ or $\mathfrak{D} \models \neg P(\bar{t})$. Hence either $\mathfrak{D} \models \Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t})$ or $\mathfrak{D} \models \neg(\Phi[\bar{t}] \wedge \Psi[\bar{t}] \wedge (\neg)P(\bar{t}))$ and CWI also holds in this case. □

Note 2. In relation to Proposition 10, we note the following.

(1) For the case that $\Phi[\bar{x}] = \mathbf{t}$, the proposition implies that CWI holds for $\Psi[\bar{x}] \wedge (\neg)P(\bar{x})$ whenever CWI holds for the window of expertise $\Psi[\bar{x}]$ of predicate P. This is the case when the window of expertise contains only base predicates.
(2) The condition that $\mathfrak{D}$ should have CWI on $\Phi[\bar{x}] \wedge \Psi[\bar{x}]$ is necessary. Assume that $\mathfrak{D}$ contains $\mathcal{LCWA}(P(x), Q(x))$ and no LCWA for Q. Clearly, there is no CWI on the query $Q(x) \wedge P(x)$. For example, if $Q(c) \notin D$, then $c \in Poss_{\mathfrak{D}}(Q(x) \wedge P(x)) \setminus Cert_{\mathfrak{D}}(Q(x) \wedge P(x))$, whether $P(c) \in D$ or not.

The next proposition shows that the class of queries with CWI is closed under certain operations.

PROPOSITION 11. *Let* $\mathfrak{D} = (D, \mathcal{L})$ *be a locally closed database and* $\Psi[\bar{x}]$, $\Phi[\bar{y}]$ *be queries on which* $\mathfrak{D}$ *conveys CWI. Then* $\mathfrak{D}$ *conveys CWI on* $\neg\Psi[\bar{x}]$, $\Psi[\bar{x}] \wedge \Phi[\bar{y}]$, $\Psi[\bar{x}] \vee \Phi[\bar{y}]$*, and, when* $x \in \bar{x}$ *and* $\bar{x}' = \bar{x} \setminus \{x\}$*, on* $(\exists x : \Psi)[\bar{x}']$ *and* $(\forall x : \Psi)[\bar{x}']$*.*

PROOF. As an example we prove the case of $(\exists x : \Psi)[\bar{x}']$ (assuming that CWI exists for $\Psi[\bar{x}]$). Let $\bar{t}'$ be a tuple of terms of the size of $\bar{x}'$ and assume that $\mathfrak{D} \not\models (\exists x : \Psi)[\bar{t}']$. Then, for some model M of $\mathfrak{D}$, we have $M \not\models (\exists x : \Psi)[\bar{t}']$, or equivalently, $M \models (\forall x : \neg\Psi)[\bar{t}']$. Thus, for all $c \in Dom^{\mathfrak{D}}$, $M \models \neg\Psi[(\bar{t}', c)]$. Since there is CWI for the query $\Psi[\bar{x}]$, there is CWI for $\neg\Psi[\bar{x}]$ as well. Therefore, for all models N of $\mathfrak{D}$, for all $c \in Dom^{\mathfrak{D}}$, we have $N \models \neg\Psi[(\bar{t}', c)]$, from which it follows that $\mathfrak{D} \models (\forall x : \neg\Psi)[\bar{t}']$, or equivalently, $\mathfrak{D} \models \neg(\exists x : \Psi)[\bar{t}']$ and CWI holds for the latter query. □

Example 8. Consider the following set of local closed world assumptions.

$$\mathcal{L} = \left\{ \begin{array}{lll} \mathcal{LCWA}(P_1(x), \mathbf{t}) & \mathcal{LCWA}(P_2(x), \mathbf{t}) & \mathcal{LCWA}(Q(x), P_1(x) \wedge P_2(x)) \\ \mathcal{LCWA}(Q(x), S(x)) & \mathcal{LCWA}(S(x), Q(x)) & \mathcal{LCWA}(R(x), Q(x)) \end{array} \right\}$$

By the last proposition, some of the formulas to which $\mathcal{M}(D, \mathcal{L})$ determines CWI can be inductively defined by the following stages.

(1) $(\neg)P_1(x)$, $(\neg)P_2(x)$,
(2) $P_1(x) \wedge P_2(x)$, $(\neg)P_1(x) \wedge P_2(x)$, $P_1(x) \wedge (\neg)P_2(x)$, $(\neg)P_1(x) \wedge (\neg)P_2(x)$,

(3) $(\neg)Q(x) \wedge P_1(x) \wedge P_2(x)$,
(4) $(\neg)S(x) \wedge Q(x) \wedge P_1(x) \wedge P_2(x)$ and so forth.

3.2 Complexity Results

In this section, we investigate the data complexity of querying locally closed databases over a vocabulary σ. First, we identify a useful tractable class. An n-ary query $\mathcal{Q}(\bar{x})$ is called *monotone*, if for each pair I, I' such that $Dom^I = Dom^{I'}$, $C^I = C^{I'}$ for each $C \in \mathcal{C}(\sigma)$ and $P^I \subseteq P^{I'}$ for each n-ary predicate $P \in \mathcal{R}(\sigma)$, it holds that $\mathcal{Q}(\bar{d})^I \leqslant \mathcal{Q}(\bar{d})^{I'}$, for all tuples $\bar{d} \in (Dom^D)^n$.

PROPOSITION 12. *Let $\mathfrak{D}$ be a locally closed database. For every monotone query $\mathcal{Q}(\bar{x})$ of arity n, $Cert_{\mathfrak{D}}(\mathcal{Q}(\bar{x}))$, that is, the set $\{\bar{d} \in (Dom^{\mathfrak{D}})^n \mid \mathfrak{D} \models \mathcal{Q}(\bar{d})\}$, is equal to $\{\bar{d} \in (Dom^{\mathfrak{D}})^n \mid D \models \mathcal{Q}(\bar{d})\}$. Moreover, this set can be computed in polynomial time in the size of the data.*

PROOF. Consider a tuple $\bar{d}$ such that $D \models \mathcal{Q}(\bar{d})\}$. Since D is the least model of $\mathfrak{D}$ and $\mathcal{Q}$ is monotone, it holds that $M \models \mathcal{Q}(\bar{d})\}$ for every model M of $\mathfrak{D}$, that is, $\mathfrak{D} \models \mathcal{Q}(\bar{d})\}$ and $\bar{d}$ is a certain answer. Polynomial time follows from well-known complexity results of relational calculus query answering in Vardi [1982]. □

Monotone queries include positive queries, that is, queries in which all database predicates occur positively, and widely studied classes of queries such as conjunctive queries with inequalities.

In contrast to the last result, in Proposition 13 given next we show that query answering in locally closed databases is in general a computationally hard problem.

Following the usual measure of complexity in databases, the results that follow are specified in terms of data complexity, that is, in terms of the size $|Dom^{\mathfrak{D}}|$ of the domain of the database instance (assuming that all the rest is fixed). Accordingly, we consider the following decision problems.

$$\text{Poss}_{\mathcal{L}}(\mathcal{Q}(\bar{x})) = \{(D, \bar{t}) \mid \bar{t} \in Poss_{(D,\mathcal{L})}(\mathcal{Q}(\bar{x}))\},$$
$$\text{Cert}_{\mathcal{L}}(\mathcal{Q}(\bar{x})) = \{(D, \bar{t}) \mid \bar{t} \in Cert_{(D,\mathcal{L})}(\mathcal{Q}(\bar{x}))\}.$$

The rationale behind the definition of these decision problems is that it seems natural to assume that local closed world assumptions $\mathcal{L}$, just like integrity constraints, are fairly constant during the lifetime of a database $\mathfrak{D}$ compared to the data in D, and that queries do not grow beyond certain limits. Therefore, we consider the complexity of fixed parameter problems of $\text{Poss}_{\mathcal{L}}(\mathcal{Q}(\bar{x}))$ and $\text{Cert}_{\mathcal{L}}(\mathcal{Q}(\bar{x}))$, where both $\mathcal{L}$ and $\mathcal{Q}$ are fixed.

PROPOSITION 13. *The decision problem $\text{Poss}_{\mathcal{L}}(\mathcal{Q}(\bar{x}))$ is in* NP *for all $\mathcal{L}$ and $\mathcal{Q}(\bar{x})$ and is* NP*-hard for some of them. $\text{Cert}_{\mathcal{L}}(\mathcal{Q}(\bar{x}))$ is in* coNP *for each $\mathcal{L}$ and $\mathcal{Q}(\bar{x})$ and is* coNP*-hard for some of them.*

PROOF. There is a one-to-one correspondence between models of $\mathfrak{D}$ and supersets D' of D satisfying $\mathcal{L}$. An algorithm to check whether $\bar{t}$ is a possible or certain answer of $\mathcal{Q}(\bar{x})$ in $\mathfrak{D} = (D, \mathcal{L})$ is to choose nondeterministically such a superset D' of D, and check whether D' satisfies $\mathcal{Q}(\bar{t})$ and each $\theta \in \mathcal{L}$. As these

checks are polynomial in the size of the domain of D, it follows that $\text{Poss}_{\mathcal{L}}(\mathcal{Q}(\bar{x}))$ is in NP and $\text{Cert}_{\mathcal{L}}(\mathcal{Q}(\bar{x}))$ is in coNP. Hardness is shown by a reduction from the graph kernel problem as follows: let $\Sigma = \{\mathit{Edge}/2, \mathit{Kernel}/1, P/1\}$ and consider the following local closed world assumptions $\mathcal{L}$:

$$\{\ \mathcal{LCWA}(\mathit{Edge}(x, y), \mathbf{t}),\ \ \mathcal{LCWA}(P(c), \neg\Phi)\ \}$$

where Φ is the formula

$$\forall x : \forall y : (\mathit{Kernel}(x) \wedge\ \mathit{Kernel}(y) \supset \neg \mathit{Edge}(x, y))\ \bigwedge \\ \forall x : (\neg \mathit{Kernel}(x) \supset \exists y : (\mathit{Kernel}(y) \wedge \mathit{Edge}(y, x))).$$

Clearly, Φ expresses that *Kernel* is a kernel of the graph described by *Edge*. For a given graph G, let D be the database with the vertices of G as domain, the edges of G represented by *Edge* and $\mathit{Kernel}^D = P^D = \emptyset$. It is easy to show that $\mathfrak{D} = (D, \mathcal{L})$ has a model in which P is a relation containing c iff G has a kernel. Since deciding whether a graph has a kernel is an NP-complete problem, deciding whether c is a possible answer to the query $P(x)$ is NP-hard, and deciding whether c is a certain answer to $\neg P(x)$ is coNP-hard. □

Next we examine the complexity of determining CWI. For a given set $\mathcal{L}$ of LCWA's and query $\mathcal{Q}(\bar{x})$, consider the following decision problem.

$$\text{CWI}_{\mathcal{L}}(\mathcal{Q}(\bar{x})) = \{D \mid (D, \mathcal{L}) \text{ has CWI on } \mathcal{Q}(\bar{x})\}$$

As the following proposition shows, the decision problem whether a locally closed database has complete knowledge on a given query is also not tractable.

PROPOSITION 14. *The decision problem* $\text{CWI}_{\mathcal{L}}(\mathcal{Q}(\bar{x}))$ *is in* coNP *for each* $\mathcal{L}$ *and* $\mathcal{Q}(\bar{x})$*, and is* coNP*-hard for some of them.*

PROOF. Observe that D is a model of $\mathfrak{D}$. Therefore, $\mathfrak{D}$ has CWI on a query $\mathcal{Q}(\bar{x})$ iff for each model D' of $\mathfrak{D}$, it holds that $\{\bar{t} \mid D' \models \mathcal{Q}(\bar{t})\} = \{\bar{t} \mid D \models \mathcal{Q}(\bar{t})\}$. To compute this, we nondeterministically choose a superset $D' \supseteq D$, verify whether each $\theta \in \mathcal{L}$ is satisfied, and verify for all tuples $\bar{t}$ whether $\mathcal{Q}(\bar{t})^D = \mathcal{Q}(\bar{t})^{D'}$; if a D' is found for which this does not hold then $\mathfrak{D}$ has no CWI on $\mathcal{Q}(\bar{x})$. The second deterministic part of this process is polynomial in the size of D.

What remains to be shown is the existence of coNP-hard instances for fixed parameters $\mathcal{L}$ and $\mathcal{Q}(\bar{x})$. This follows from the fact that, in the case of the kernel-databases constructed in the proof of Proposition 13, $\mathfrak{D}$ has complete information on $P(c)$ iff $\neg P(x)$ has c as a certain answer. The latter decision problem is coNP-hard. □

Proposition 14 gives us the complexity of deciding whether there is CWI on a query $\mathcal{Q}(\bar{x})$ in a specific database $\mathfrak{D} = (D, \mathcal{L})$. In Levy [1996], Levy studies a more ambitious problem and presents an efficient (polynomial) decision procedure for determining whether, for a given set $\mathcal{L}$ of local closed world assumptions and query $\mathcal{Q}(\bar{x})$, there is CWI on $\mathcal{Q}(\bar{x})$ in *all* locally closed databases containing $\mathcal{L}$. This decision procedure takes advantage of the fact that Levy's windows of expertise are special cases of (positive) conjunctive queries (called by Levy variable-interval queries) and $\mathcal{Q}(\bar{x})$ is the union of (positive) conjunctive

queries. Not surprisingly, such a decision procedure does not exist for the more expressive formalism of the current article.

PROPOSITION 15. *The question whether all locally closed databases* $(\cdot, \mathcal{L})$ *convey CWI on a query* $\mathcal{Q}(\bar{x})$ *is undecidable.*

PROOF. The proof is a variant of the co-NP-hardness proof; the main idea is that the problem at hand can be reduced to the validity checking problem of first-order formulas in the class of finite structures, and this problem is, by Trakhtenbrot's theorem, undecidable. Consider a relational database schema Σ such that $P/1 \in \Sigma$. Take $\mathcal{Q}(\bar{x}) = P(c)$ and $\mathcal{L} = \{\theta\}$ with $\theta = \mathcal{LCWA}(P(c), \varphi)$, where φ is a sentence not containing P. We observe that $\neg\varphi$ has a finite model iff there exists a database $(D, \mathcal{L})$ over Σ that has no CWI on $P(c)$. Indeed, if $\neg\varphi$ has a finite model M, then take D to be M extended with the empty relation for P. Clearly, both D and the extension of D in which c is added to P's table are models of $(D, \mathcal{L})$, and hence this database has no CWI on $P(c)$. Conversely, a database $(D, \mathcal{L})$ that has no CWI on $P(c)$ does not contain $P(c)$ but has a finite model M in which $P(c)$ is true. This model satisfies $\mathcal{M}_D(\theta) \equiv \varphi \supset (P(c) \supset P(c) \in D)$, and hence, $M \models \neg\varphi$. It follows that there is CWI on $P(c)$ in all databases $(D, \mathcal{L})$ iff φ is satisfied in all finite structures. By Trakhtenbrot's theorem, this is undecidable (see Trakhtenbrot [1963]). □

So far, the results in this section give little reason for optimism regarding practical applicability of local closed world assumptions. But, as it turns out, in many applications, there is no need to have *all* certain answers to a query; often, it suffices to have a sufficiently large subset of them. For instance, if a company searches an (incomplete) database for a provider of some urgently required service, it will be satisfied by finding *some* candidate providers. Likewise, in many applications, it would not harm if the answers to a possible query contain a few extra "impossible" elements. This is the case, for example, when a company wants to advertise one of its services and queries a database for a group of potential clients. It would not care to receive some *additional* companies that may not really be interested in its services. In both of these situations, tractable approximate methods may be very useful. This is the purpose of the next section.

The other, more conventional approach to the complexity problem is to restrict the expressiveness of the language so that efficient query processing would be possible. As it turns out, shortly we obtain such results as well, though in a slightly indirect way: we will show that for certain classes of queries and local closed world assumptions, the approximate methods are *optimal* in the sense that they compute exactly the certain and possible answers to queries. Thus, these combinations of queries and local closed world assumptions provide tractable subformalisms.

Bibliographical note. The main results in this section have been published before in Cortés-Calabuig et al. [2007] (Propositions 13, 14 and 9), and Cortés-Calabuig et al. [2008] (Propositions 9 and 15). Some proofs, however, have been rewritten to make them more accessible to the reader.

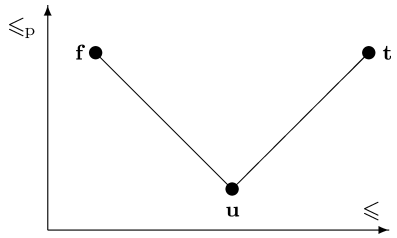

Fig. 3. Hasse diagram of $\mathcal{THREE}$.

4. APPROXIMATIVE REASONING

The basic idea of the approximative reasoning is to compute a three-valued structure that provides a "good" approximation of all models of $\mathfrak{D}$ and then to evaluate queries with respect to this structure. In the next sections, we present a (polynomial) algorithm to compute such an approximation and use it to obtain approximative answers for queries.

We shall then point out to two major weaknesses. The first one is that a naive application of the algorithm requires to recompute the approximating structure for each database modification. To address it, we use fixpoint queries that leave the three-valued approximating structure implicit. This allows us to compute query answers based on the content of the database together with some symbolic manipulations on the LCWA's. The second weakness is with respect to the precision of the approximate reasoning. In Section 5 we address this issue and show that for a broad class of locally closed databases our methods provide optimal answers that can be obtained in a tractable manner.

4.1 Three-Valued Structures

We first review the needed concepts of three-valued logics. The truth values of the corresponding three-valued semantics, $\mathcal{THREE} = \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, standing for true, false, and unknown, are usually arranged in two orders: the truth order, $\leqslant$, which is a linear order in which $\mathbf{f} \leqslant \mathbf{u} \leqslant \mathbf{t}$, and the precision order $\leqslant_p$, which is a partial order on $\mathcal{THREE}$, where $\mathbf{u}$ is the least element, and $\mathbf{t}$ and $\mathbf{f}$ are incomparable maximal elements. The structure of $\mathcal{THREE}$ can be drawn as a double Hasse diagram (Figure 3).

Conjunction in $\mathcal{THREE}$ is defined by the $\leqslant$-glb (the greatest lower bound according to the truth order) of this structure; disjunction is defined by the $\leqslant$-lub (the least upper bound according to the truth order), and the negation operator is associated with the $\leqslant$-involution, that is: $\neg\mathbf{t} = \mathbf{f}$, $\neg\mathbf{f} = \mathbf{t}$, and $\neg\mathbf{u} = \mathbf{u}$. A three-valued relation (of arity n) on some domain Dom is a function from Dom^n to $\mathcal{THREE}$. A three-valued structure $\mathcal{K}$ of vocabulary σ consists of a domain $Dom^{\mathcal{K}}$, for each constant or variable symbol $C \in \sigma$ an appropriate domain element $C^{\mathcal{K}}$ in $Dom^{\mathcal{K}}$, and for each predicate symbol P a three-valued relation $P^{\mathcal{K}}$ on $Dom^{\mathcal{K}}$. The predicates $\mathbf{t}$, $\mathbf{f}$, and $=$ are interpreted in the standard way. A structure $\mathcal{K}$ is two-valued if $\mathbf{u}$ is not in the reach of its relations $P^{\mathcal{K}}$.

The concept of a Herbrand structure generalizes to the three-valued case. An equivalent representation of a three-valued Herbrand structure is as a function from the Herbrand base $HB(\sigma)$, the set of ground atoms of σ, to $\mathcal{THREE}$. The collection of three-valued Herbrand interpretations of σ is denoted $\mathfrak{L}^c_\sigma$ and its subset of two-valued structures is denoted $\mathfrak{L}_\sigma$. We will drop σ when clear from the context. On the set $\mathfrak{L}^c$, the truth order $\leqslant$ and precision order $\leqslant_{\mathrm{p}}$ are defined by pointwise extension of the corresponding orders in $\mathcal{THREE}$, that is, $I \leqslant J$ ($I \leqslant_{\mathrm{p}} J$) if for each atom $P(\bar{a})$ in the Herbrand base, $P(\bar{a})^I \leqslant P(\bar{a})^J$ ($P(\bar{a})^I \leqslant_{\mathrm{p}} P(\bar{a})^J$). In $\mathfrak{L}^c$, $\leqslant$ is a lattice order and $\leqslant_{\mathrm{p}}$ is a chain-complete order.[6] It is well-known that a monotone operator on a chain-complete poset has a least fixpoint. This property will be exploited in the following section.

The truth assignment for compound sentences can be extended to three-valued structures in several ways. In this article, we will see two of them: the standard Kleene truth assignments and supervaluations (see Note 4 that follows). The standard Kleene truth assignment [Kleene 1952] is defined through the same recursive rules as two-valued truth assignment. For instance,

$$(\psi \wedge \phi)^{\mathcal{K}} = Min_{\leqslant}(\psi^{\mathcal{K}}, \phi^{\mathcal{K}});$$
$$(\neg\psi)^{\mathcal{K}} = \neg(\psi^{\mathcal{K}});$$
$$(\forall x : \psi[x])^{\mathcal{K}} = Min_{\leqslant}\{\psi[a]^{\mathcal{K}} \mid a \in Dom^{\mathcal{K}}\}.$$

A well-known and, in this article, useful result is that Kleene's truth assignment can be simulated by two-valued truth assignment.

Definition 11 (*Simulation*). For vocabulary σ, define σ' as the set of all constant symbols of σ together with, for each predicate $P \in \mathcal{R}(\sigma)$, the predicate symbols P^c and $P^{c\neg}$.[7]

—Let $\mathcal{K}$ be a three-valued σ-structure. We say that a two-valued σ'-structure I *simulates* $\mathcal{K}$, iff $\mathcal{K}$ and I have the same domain and assign the same interpretations to constant symbols, and for each predicate $P \in \mathcal{R}(\sigma)$, $(P^c)^I = \{\bar{d} \mid P^{\mathcal{K}}(\bar{d}) = \mathbf{t}\}$, and $(P^{c\neg})^I = \{\bar{d} \mid P^{\mathcal{K}}(\bar{d}) = \mathbf{f}\}$.

—For any σ-formula $\varphi[\bar{x}]$, let $\varphi^c[\bar{x}]$ be the formula obtained by substituting $P^c(\bar{t})$ for positive occurrences of atoms $P(\bar{t})$ and substituting $\neg P^{c\neg}(\bar{t})$ for negative occurrences of atoms $P(\bar{t})$, for all $P \in \mathcal{R}(\sigma)$.

The following proposition is well-known.

PROPOSITION 16. *If I simulates $\mathcal{K}$, then for each formula $\varphi[\bar{x}]$ and suitable tuple of domain elements $\bar{d}$, it holds that $(\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{t}$ iff $(\varphi^c[\bar{d}])^I = \mathbf{t}$ and $(\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{f}$ iff $((\neg\varphi^c[\bar{d}]))^I = \mathbf{f}$.*

As a corollary we obtain tractability of three-valued truth evaluation and query answering.

COROLLARY 1. *Given a finite three-valued σ-structure $\mathcal{K}$, for each formula $\varphi[\bar{x}]$, the sets $\{\bar{d} \mid (\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{t}\}$, $\{\bar{d} \mid (\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{f}\}$, and $\{\bar{d} \mid (\varphi[\bar{d}])^{\mathcal{K}} = \mathbf{u}\}$ can be computed in polynomial time in the size of $\mathcal{K}$ (and exponential in the size of φ).*

[6] In a chain-complete order, each chain, that is, each totally ordered subset, has a least upperbound.
[7] Intuitively, P^c stands for *certainly* P and $P^{c\neg}$ for *certainly not* P.

Note 3. An important property of Kleene's truth assignment is its $\leqslant_p$-monotonicity. That is, $\mathcal{K} \leqslant_p \mathcal{K}'$ implies that $\varphi^{\mathcal{K}} \leqslant_p \varphi^{\mathcal{K}'}$ for every sentence φ.

Note 4. An alternative truth assignment satisfying the same monotonicity property is *supervaluation* [van Fraassen 1966]. The supervaluation $sv_{\mathcal{K}}(\varphi)$ of a sentence φ in a three-valued structure $\mathcal{K}$ is defined as $sv_{\mathcal{K}}(\varphi) = lub_{\leqslant_p}\{\varphi^M \mid M \in \mathfrak{L} : \mathcal{K} \leqslant_p M\}$. It is easy to prove that for each $\mathcal{K}$ and φ, $\varphi^{\mathcal{K}} \leqslant_p sv_{\mathcal{K}}(\varphi)$. The price for this augmented precision is complexity: for a given φ, deciding whether $sv_{\mathcal{K}}(\varphi) = \mathbf{t}$ for finite $\mathcal{K}$, is in coNP, and coNP-hard for some φ.

4.2 Approximating Theories by Three-Valued Structures

In what follows, we view a database as a logic theory containing $\mathsf{UNA}(\sigma) \wedge \mathsf{DCA}(\sigma)$. Let σ be a vocabulary consisting of finitely many predicates and constants.

Definition 12 (*Optimal Approximation, $\mathcal{O}_\Gamma$*). Let Γ be a satisfiable theory based on σ containing $\mathsf{UNA}(\sigma) \wedge \mathsf{DCA}(\sigma)$.[8] We say that a three-valued Herbrand σ-interpretation $\mathcal{K}$ *approximates* Γ (from below) iff for every two-valued Herbrand model M of Γ, $\mathcal{K} \leqslant_p M$. The *optimal approximation for* Γ is the three-valued Herbrand structure $glb_{\leqslant_p}(\{M \mid M \models \Gamma\})$ where M ranges over all the two-valued Herbrand models of Γ. This structure will be denoted $\mathcal{O}_\Gamma$.

Note 5. The structure $\mathcal{O}_\Gamma$ is the most precise of all three-valued Herbrand σ-structures approximating Γ. It is well-defined since the set of its Herbrand models is nonempty and every nonempty set $S \subseteq \mathfrak{L}^c$ has a greatest $\leqslant_p$-lower bound.

An obvious alternative characterization of $\mathcal{O}_\Gamma$ is given in the following proposition.

PROPOSITION 17. *For each ground atom $P(\bar{a}) \in HB(\sigma)$, $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{t}$ iff $\Gamma \models P(\bar{a})$, and $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{f}$ iff $\Gamma \models \neg P(\bar{a})$.*

Since all models of a theory containing $\mathsf{UNA}(\sigma) \wedge \mathsf{DCA}(\sigma)$ are isomorphic to a Herbrand structure, the following proposition holds.

PROPOSITION 18. *Let $\mathcal{K}$ be an approximation of Γ. For any sentence φ, if $\varphi^{\mathcal{K}} = \mathbf{t}$, then $\Gamma \models \varphi$ and if $\varphi^{\mathcal{K}} = \mathbf{f}$, then $\Gamma \models \neg\varphi$.*

The inverse, of course, does not hold, not even when $\mathcal{K} = \mathcal{O}_\Gamma$. For a simple example, consider a vocabulary σ with a predicate symbol P and a constant a and let $\Gamma = \emptyset$. It holds that $P(a)^{\mathcal{O}_\emptyset} = \mathbf{u}$. It also holds that $\emptyset \models P(a) \vee \neg P(a)$ while $(P(a) \vee \neg P(a))^{\mathcal{O}_\emptyset} = \mathbf{u}$. Likewise, it does not suffice that $\varphi^{\mathcal{K}} = \mathbf{u}$ to guarantee that $\Gamma \cup \{\varphi\}$ or $\Gamma \cup \{\neg\varphi\}$ is satisfiable. For example, $(P(a) \wedge \neg P(a))^{\mathcal{O}_\emptyset} = \mathbf{u}$, but this formula is not satisfiable.

Definition 13. (*Certain and Possible Answers with respect to a 3-Valued Interpretation*). Given a query $\mathcal{Q}(\bar{x})$ in σ and a three-valued σ-interpretation $\mathcal{K}$, denote by $\bar{d}$ a tuple of domain elements.

[8] Hence, all its models are (isomorphic to) Herbrand models.

—$\bar{d}$ is a *certain answer* in $\mathcal{K}$ for $\mathcal{Q}(\bar{x})$, if $\mathcal{Q}(\bar{d})^{\mathcal{K}} = \mathbf{t}$. We denote the set of certain answers by $Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x}))$.
—$\bar{d}$ is a *possible answer* in $\mathcal{K}$ for $\mathcal{Q}(\bar{x})$, if $\mathcal{Q}(\bar{d})^{\mathcal{K}} \geq \mathbf{u}$. We denote the set of possible answers by $Poss_{\mathcal{K}}(\mathcal{Q}(\bar{x}))$.

Given that truth values of sentences can be computed in polynomial time, we have the following tractability result.

PROPOSITION 19. *For each finite three-valued σ-structure $\mathcal{K}$ and a query $\mathcal{Q}(\bar{x})$ over σ, the sets $Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x}))$ and $Poss_{\mathcal{K}}(\mathcal{Q}(\bar{x}))$ are polynomially computable in the size of $\mathcal{K}$ (and exponentially computable in the size of $\mathcal{Q}(\bar{x})$).*

A locally closed database $\mathfrak{D} = (D, \mathcal{L})$ corresponds to a theory $\mathcal{M}(\mathfrak{D})$ which is satisfiable and includes $\mathsf{UNA}(\sigma_{\mathfrak{D}}) \wedge \mathsf{DCA}(\sigma_{\mathfrak{D}})$. In what follows, by an approximation $\mathcal{K}$ of $\mathfrak{D}$, we mean a three-valued structure $\mathcal{K}$ that approximates $\mathcal{M}(\mathfrak{D})$. As a corollary to the Notes 3 and 5, and Proposition 18, we have the following proposition.

PROPOSITION 20. *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database and $\mathcal{K}$ an approximation of $\mathfrak{D}$. It holds that*

$$Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x})) \subseteq Cert_{\mathcal{O}_{\mathcal{M}(\mathfrak{D})}}(\mathcal{Q}(\bar{x})) \subseteq \mathrm{Cert}_{\mathfrak{D}}(\mathcal{Q}(\bar{x})) \subseteq \mathrm{Poss}_{\mathfrak{D}}(\mathcal{Q}(\bar{x})) \subseteq Poss_{\mathcal{O}_{\mathcal{M}(\mathfrak{D})}}(\mathcal{Q}(\bar{x})) \subseteq Poss_{\mathcal{K}}(\mathcal{Q}(\bar{x})).$$

It thus appears that an approximation $\mathcal{K}$ of $\mathfrak{D}$ allows us to compute an underestimate of the certain answers and an overestimate of the possible answers of a query, in polynomial time. From these results it is clear that a tractable method to compute three-valued approximations produces a tractable sound approximative query answering method. Such a method is provided in the next section.

In the same spirit, an approximation $\mathcal{K}$ of $\mathfrak{D}$ gives us also a tractable, sound, but incomplete method to verify CWI for queries. Indeed, by Proposition 9 we have the next proposition.

PROPOSITION 21. *Let $\mathcal{K}$ be an approximation of $\mathfrak{D}$. $\mathfrak{D}$ has CWI on $\mathcal{Q}(\bar{x})$ if $Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x})) = Poss_{\mathcal{K}}(\mathcal{Q}(\bar{x}))$.*

Given $\mathcal{K}$, testing whether $Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x})) = Poss_{\mathcal{K}}(\mathcal{Q}(\bar{x}))$ amounts to check whether $\{\bar{d} \in HU^n | \mathcal{Q}(\bar{d})^{\mathcal{K}} = \mathbf{u}\}$ is empty. This can be done in polynomial time in the size of the database.

4.3 Constructing Approximations by Fixpoint Computations

Let $\mathfrak{D}$ be a locally closed database with a domain $Dom = \mathcal{C}(\sigma_{\mathfrak{D}})$. In order to construct a three-valued approximation for $\mathfrak{D}$, we first introduce a fixpoint operator on $\mathfrak{L}^c$, the set of three-valued Herbrand structures with domain Dom.

Definition 14 (*The Operator $\mathcal{A}pp_{\mathfrak{D}}$*). The operator $\mathcal{A}pp_{\mathfrak{D}} : \mathfrak{L}^c \to \mathfrak{L}^c$ maps a three-valued structure $\mathcal{K}$ to $\mathcal{K}' = \mathcal{A}pp_{\mathfrak{D}}(\mathcal{K})$ such that, for every predicate P of

Σ and every tuple $\bar{d}$:

$$P(\bar{a})^{\mathcal{K}'} = \begin{cases} \mathbf{t} & \text{if } P(\bar{d}) \in D, \\ \mathbf{f} & \text{if } P(\bar{d}) \notin D \text{ and there exists } \mathcal{LCWA}(P(\bar{x}), \Psi_P[\bar{x}]) \in \mathcal{L} \\ & \text{such that } \Psi_P[\bar{d}]^{\mathcal{K}} = \mathbf{t}, \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

The idea for constructing an approximation of $\mathfrak{D}$ is to start from the structure with total ignorance (the valuation $\mathcal{K}_\perp$ that assigns $\mathbf{u}$ to every ground atom), and to iterate $\mathcal{A}pp_{\mathfrak{D}}$, thereby gradually extending the definite knowledge using the database and its LCWA's.

PROPOSITION 22. *$\mathcal{A}pp_{\mathfrak{D}}$ is a $\leqslant_{\mathrm{p}}$-monotone operator on the chain-complete poset $\mathfrak{L}^c$, thus it has a $\leqslant_{\mathrm{p}}$-least fixpoint. Moreover, the least fixpoint can be computed in polynomial time in the size of the database (although exponential in the size of the windows of expertise).*

PROOF. It is straightforward to verify that $\mathcal{A}pp_{\mathfrak{D}}$ is $\leqslant_{\mathrm{p}}$-monotone. By (an extension of) the well-known Knaster-Tarski theorem, it follows that the sequence of three-valued structures, starting with the least precise structure and constructed by iterating $\mathcal{A}pp_{\mathfrak{D}}$, is monotonically increasing in precision and reaches $\mathcal{A}pp_{\mathfrak{D}}$'s $\leqslant_{\mathrm{p}}$-least precise fixpoint. Polynomial complexity follows from the fact that per application of the operator, the number of queries to be solved is polynomial in the size of the database and each query can be solved in polynomial time, while the number of iterations is at most polynomial in the size of the database. □

Definition 15 (*The 3-Valued Interpretation $\mathcal{C}_{\mathfrak{D}}$*). The three-valued interpretation that is the $\leqslant_{\mathrm{p}}$-least fixpoint of $\mathcal{A}pp_{\mathfrak{D}}$ is denoted as $\mathcal{C}_{\mathfrak{D}}$.

PROPOSITION 23. *For base predicates P, it holds that $P^{\mathcal{C}_{\mathfrak{D}}} = P^D$.*

Example 9. Next are some examples of computing $\mathcal{C}_{\mathfrak{D}}$.

(1) Consider $D = \emptyset$, $Dom^D = \{a\}$, and $\theta_1 = \mathcal{LCWA}(P(x), R(x))$. In the first iteration, none of the first two rules of the operator applies: the first one is not applicable since the database is empty; the second because $P(a)$'s window of expertise $R(a)$ does not evaluate to $\mathbf{t}$. As a consequence, $R(a)^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$ and $P(a)^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$.

(2) Consider $D = \emptyset$, $Dom^D = \{a\}$, $\theta_1 = \mathcal{LCWA}(Q(x), \mathbf{t})$, and $\theta_2 = \mathcal{LCWA}(P(x), \neg Q(x))$. In the first iteration, the windows of expertise of $Q(a)$ is true and D is empty, hence the second rule applies and we derive $Q(a)^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{f}$. In the second iteration, the window of expertise of $P(a)$ holds, and the second rule yields $P(a)^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{f}$. Here, a fixpoint is reached.

(3) Consider again the database of Example 1 and the first two LCWA expressions of Example 5: $\mathcal{LCWA}(Dept(x, y), y = CS)$. In the first iteration, we obtain that $Dept(a, b)^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{t}$ for all the tuples (a, b) in $Dept^D$. In the second iteration, we can derive that $Dept(a, CS)^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{f}$ for all $a \notin \{BD, TD\}$. For the remaining tuples (a, b) not covered by the previous two cases, we have $Dept(a, b)^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$.

The following theorem shows that $\mathcal{C}_\mathfrak{D}$ is a sound approximation of $\mathfrak{D}$.

THEOREM 1 (SOUNDNESS). *$\mathcal{C}_\mathfrak{D}$ approximates $\mathfrak{D}$.*

PROOF. For each Herbrand model M of $\mathfrak{D}$, we have to show that $\mathcal{C}_\mathfrak{D} \leqslant_\mathrm{p} M$. According to the definition of $\mathcal{A}pp_\mathfrak{D}$, any true atom in $\mathcal{A}pp_\mathfrak{D}(M)$ belongs to the database and hence is true in M. By the same definition, an atom $P(\bar{d})$, false in $\mathcal{A}pp_\mathfrak{D}(M)$ does not belong to the database; moreover, $\Psi_P[\bar{d}]$ is true in M. Since M satisfies the local closed world assumption, $P(\bar{d})$ is also false in M. Hence $\mathcal{A}pp_\mathfrak{D}(M) \leqslant_\mathrm{p} M$ and M is a prefixpoint of $\mathcal{A}pp_\mathfrak{D}$. Since $\mathcal{C}_\mathfrak{D}$ is the least fixpoint of $\mathcal{A}pp_\mathfrak{D}$, it is also the least prefixpoint, and hence $\mathcal{C}_\mathfrak{D} \leqslant_\mathrm{p} M$. □

As a corollary to Proposition 20, we obtain the soundness theorem of the approximate method.

COROLLARY 2. *For any locally closed database $\mathfrak{D} = (D, \mathcal{L})$, it holds that*

$$Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})) \subseteq \mathrm{Cert}_\mathfrak{D}(\mathcal{Q}(\bar{x})) \subseteq \mathrm{Poss}_\mathfrak{D}(\mathcal{Q}(\bar{x})) \subseteq Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})).$$

The previous results give us a tractable method for computing possible and certain answers to queries: by first computing $\mathcal{C}_\mathfrak{D}$ and then evaluating queries against it. The latter can be implemented using standard techniques from databases, by transforming queries and solving them in a database *simulating* $\mathcal{C}_\mathfrak{D}$, as explained in Proposition 16. Since the transformed query has the same size as the original query, there is no loss of efficiency compared to standard (two-valued) database querying. By application of Proposition 9, we also obtain a tractable approximate method for checking CWI.

Tractability, however, has a price. As the following example shows, by applying this method, in certain cases, we lose accuracy.

Example 10. Let us compare $\mathcal{C}_\mathfrak{D}$ and $\mathcal{O}_\mathfrak{D}$ of $\mathfrak{D} = (D, \mathcal{L})$ where $D = \emptyset$, $Dom^D = \{a\}$, $\theta = \mathcal{LCWA}(Q(x), P(x) \vee \neg P(x))$. This database has models in which $P(a)$ is true and others in which $P(a)$ is false but, because of its LCWA, $Q(a)$ is false in all of them. Thus $P(a)^{\mathcal{O}_\mathfrak{D}} = \mathbf{u}$ and $Q(a)^{\mathcal{O}_\mathfrak{D}} = \mathbf{f}$. However, since $P(a) \vee \neg P(a)$ evaluates to $\mathbf{u}$ in each structure $\mathcal{K}$ for which $P(a)^\mathcal{K} = \mathbf{u}$, we have that $Q(a)^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$. The answer for the query $\neg Q$ in $\mathcal{C}_\mathfrak{D}$ is $\mathbf{u}$, while it is $\mathbf{t}$ when posed with respect to $\mathfrak{D}$ or $\mathcal{O}_\mathfrak{D}$.

In Section 5, the loss of accuracy and when it can be prevented will be studied in more detail. But we first address an important drawback of the approach, the need to recompute $\mathcal{C}_\mathfrak{D}$ each time the database changes.

4.4 Simulating Approximate Query Answering by Fixpoint Queries

The recomputation of $\mathcal{C}_\mathfrak{D}$ after each update of D in the locally closed database $\mathfrak{D} = (D, \mathcal{L})$ over Σ can be (partially) avoided by using fixpoint formulas or Datalog programs that symbolically describe the construction of $\mathcal{C}_\mathfrak{D}$. Using these expressions, certain or possible answers to queries can be computed by transforming the query into a fixpoint query or in a query with respect to some Datalog program. Running the resulting query directly against D avoids the explicit construction of $\mathcal{C}_\mathfrak{D}$.

We will represent a query for certain or possible answers by a fixpoint expression in the logic LFP^{simult} of simultaneous fixpoints [Libkin 1995]. For reasons of clarity, we use a slightly unusual notation for such expressions and write them in the form of

$$[\mathbf{lfp}_{R_i,\Delta}](\bar{t}),$$

where $\bar{t}$ is a tuple of terms of length n_i, R_i is a predicate variable of arity n_i, and Δ is a collection of rules

$$\{\forall \bar{x}_j : R_j(\bar{x}_j) \leftarrow \varphi_j[\bar{x}_j] \mid 1 \leq j \leq n\},$$

where each φ_j is an arbitrary first-order formula over $\Sigma \cup \{R_1, \ldots, R_n\}$, $R_1, \ldots, R_n$ are predicate variables with only positive occurrences in φ_j, and the arity of each R_j is n_j, the length of $\bar{x}_j$[9].

We now extend the two-valued truth assignment to fixpoint expressions. Let I be a $\sigma_{\mathfrak{D}}$-structure and $[\mathbf{lfp}_{R_i,\Delta}](\bar{t})$ a fixpoint expression without free variables (but possibly containing domain elements of I). We define $([\mathbf{lfp}_{R_i,\Delta}](\bar{t}))^I = \mathbf{t}$ iff $\bar{t}^I \in \mathcal{R}_i$, where $\mathcal{R}_i$ is the i'th argument in the least fixpoint $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ of the operator $\Gamma_{\Delta,I}$ associated to Δ and I. The operator $\Gamma_{\Delta,I}$ operates on tuples $(S_1, \ldots, S_n)$ of relations of arity $n_1, \ldots, n_n$ and such that $\Gamma_{\Delta,I}(S_1, \ldots, S_n) = (S'_1, \ldots, S'_n)$ for

$$S'_i = \{\bar{d} \mid I[R_1 : S_1, \ldots, R_n : S_n] \models \varphi_i[\bar{d}]\}.$$

Here, $I[R_1 : S_1, \ldots, R_n : S_n]$ denotes the structure extending I by interpreting each R_i by S_i. The defined operator is indeed a monotone lattice operator and its least fixpoint is well-defined.

It is worth noting that Δ is an extended Datalog program as defined in Van Gelder [1993] or a *positive definition* as defined in FO[ID] [Denecker and Ternovska 2004] and that its semantics, that is, its least fixpoint, coincides with the well-founded model of Δ. In Datalog, the fixpoint query $[\mathbf{lfp}_{R_i,\Delta}](\bar{t})$ would be posed as the query $R_i(\bar{t})$ with respect to the Datalog program Δ.

In the context of I, the collection of rules Δ coinductively defines the tuple $(\mathcal{R}'_1, \ldots, \mathcal{R}'_n)$ as the greatest fixpoint of Γ_Δ. A greatest fixpoint expression is of the form

$$[\mathbf{gfp}_{R_i,\Delta}](\bar{t}).$$

If $[\mathbf{gfp}_{R_i,\Delta}](\bar{t})$ contains no free variables, we have $([\mathbf{gfp}_{R_i,\Delta}](\bar{t}))^I = \mathbf{t}$ if $\bar{t}^I \in \mathcal{R}'_i$.

Least fixpoints and greatest fixpoints are connected through a duality result. Given the set Δ, define Δ' as the collection of rules

$$\{\forall \bar{x}_j : R'_j(\bar{x}_j) \leftarrow \varphi'_j[\bar{x}_j] \mid 1 \leq j \leq n\},$$

where $\varphi'_j[\bar{x}_i]$ denotes the formula obtained from $\neg\varphi_j[\bar{x}_j]$ by replacing each occurrence of an atom $R_i(\bar{t})$ by $\neg R'_i(\bar{t})$. The formulas φ'_j are positive in the R'_k's. It

[9]In the standard notation of LFP^{simult}, the fixpoint expression $[\mathbf{lfp}_{R_i,\Delta}](\bar{t})$ would be denoted $[\mathbf{lfp}_{R_1(\bar{x}_1),\ldots,R_n(\bar{x}_n)}(\varphi_1[\bar{x}_1], \ldots, \varphi_n[\bar{x}_n]), R_i](\bar{t})$, denoting that $\bar{t}$ belongs to relation R_i where $R_1, \ldots, R_n$ are defined by simultaneous monotone induction by the formulas $\varphi_1, \ldots, \varphi_n$ respectively. Both notations are syntactic variants. The rule notation is more convenient for linking predicates and the formulas defining them and, more importantly, for showing the close relationship to Datalog.

is well known that for $(\mathcal{R}_1, \ldots, \mathcal{R}_n)$ the least fixpoint of Γ_Δ, and for $(\mathcal{R}'_1, \ldots, \mathcal{R}'_n)$ the greatest fixpoint of $\Gamma_{\Delta'}$, the relations $\mathcal{R}_i, \mathcal{R}'_i$ are complements. It follows that $[\mathbf{lfp}_{R_i,\Delta}](\bar{t})$ and $\neg[\mathbf{gfp}_{R'_i,\Delta'}](\bar{t})$ are logically equivalent.

We now proceed with formalizing approximate queries using such fixpoint queries. Assume that the database schema Σ contains predicates $P_1, \ldots, P_n$. For each of these predicate symbols, we introduce four new predicate variables P_i^c, P_i^p, $P_i^{c\neg}$, and $P_i^{p\neg}$ of the same arity as P_i, to represent the information about P_i that is present in the least fixpoint $\mathcal{C}_\mathfrak{D}$: P_i^c and P_i^p for, respectively, the certain and the possible tuples of P_i and $P_i^{c\neg}$ and $P_i^{p\neg}$ for, respectively, the tuples that certainly and possibly do not belong to P_i. Along this intuition, it is clear that P_i^p and $P_i^{c\neg}$, respectively P_i^c and $P_i^{p\neg}$ represent each other's complements.

To each formula Ψ with predicate symbols among $P_1, \ldots, P_n$, we associate two formulas, representing the certain instances and the possible instances of the formula when interpreted as a query.

—Ψ^c is the formula obtained by substituting $P_i^c(\bar{t})$ for each positive occurrence of $P_i(\bar{t})$ in Ψ, and substituting $\neg P_i^{c\neg}(\bar{t})$ for each negative occurrence of $P_i(\bar{t})$ in Ψ. As mentioned in Proposition 16, this formula can be used to compute three-valued answers for Ψ.

—Ψ^p is, inversely, the formula obtained by substituting $P_i^p(\bar{t})$ for each positive occurrence of $P_i(\bar{t})$ in Ψ, and substituting $\neg P_i^{p\neg}(\bar{t})$ for each negative occurrence of $P_i(\bar{t})$ in Ψ.

Observe that Ψ^c contains only predicate variables of the form P_i^c and $P_i^{c\neg}$, and Ψ^p only $P_i^p(\bar{t})$ and $P_i^{p\neg}(\bar{t})$ and that both Ψ^c and Ψ^p contain only positive occurrences of the new predicate variables.

Definition 16 (*Certain Rules,* $\Delta^c_{\mathcal{Q},\mathcal{L}}$). Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over Σ and $\mathcal{Q}(\bar{x})$ a query. In addition to the predicate variables P_i^c and $P_i^{c\neg}$, we introduce two extra predicate variables Q^c and $Q^{c\neg}$ with arity the number of free variables of $\mathcal{Q}(\bar{x})$ and define the set of *certain rules* $\Delta^c_{\mathcal{Q},\mathcal{L}}$ as the following collection of rules:

—two rules for the query:

$$Q^c(\bar{x}) \leftarrow \mathcal{Q}(\bar{x})^c$$
$$Q^{c\neg}(\bar{x}) \leftarrow (\neg\mathcal{Q}(\bar{x}))^c$$

—for each database predicate P_i, two rules:

$$P_i^c(\bar{x}_i) \leftarrow P_i(\bar{x}_i)$$
$$P_i^{c\neg}(\bar{x}_i) \leftarrow \neg P_i(\bar{x}_i) \wedge (\Psi_{P_i}[\bar{x}_i])^c$$

where Ψ_{P_i} is the window of expertise of P_i.

We define the certain answer query for $\mathcal{Q}(\bar{x})$ as $[\mathbf{lfp}_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and the possible answer query for $\mathcal{Q}(\bar{x})$ as $\neg[\mathbf{lfp}_{Q^{c\neg},\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$.

Intuitively, these definitions define Q^c as the collection of certain instances of $\mathcal{Q}(\bar{x})$ and $Q^{c\neg}$ as the collection of certain instances of $\neg\mathcal{Q}(\bar{x})$. The rule for P_i^c captures the idea that elements of P_i^c, that is, certain elements of P_i, are those

in P_i's table, and the rule for $P_i^{c\neg}$ that tuples for which P_i is certainly false are those that do not occur in the database and for which the window of expertise is certainly satisfied. The two fixpoint queries are legal LFP^{simult} queries. In particular, all new symbols occur positively in the formulas of the rules.

Similarly, we can define a set of possible rules.

Definition 17 (*Possible Rules,* $\Delta^p_{\mathcal{Q},\mathcal{L}}$). Under the same assumptions as in Definition 16 we introduce variables Q^p and $Q^{p\neg}$ and define the set of possible rules $\Delta^p_{\mathcal{Q},\mathcal{L}}$ as the following collection of rules:

—two rules for the query:

$$Q^p(\bar{x}) \leftarrow \mathcal{Q}(\bar{x})^p$$
$$Q^{p\neg}(\bar{x}) \leftarrow (\neg\mathcal{Q}(\bar{x}))^p$$

—for each database predicate P_i, two rules:

$$P_i^p(\bar{x}_i) \leftarrow P_i(\bar{x}_i) \vee (\neg\Psi_{P_i}[\bar{x}_i])^p$$
$$P_i^{p\neg}(\bar{x}_i) \leftarrow \neg P_i(\bar{x}_i)$$

Note 6. Clearly, $\Delta^p_{\mathcal{Q},\mathcal{L}}$ is equivalent to the dual of $\Delta^c_{\mathcal{Q},\mathcal{L}}$. It follows, then, that $\neg[\mathbf{lfp}_{Q^{c\neg},\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and $[\mathbf{gfp}_{Q^p,\Delta^p_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ are equivalent, and so are $[\mathbf{lfp}_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and $\neg[\mathbf{gfp}_{Q^{p\neg},\Delta^p_{\mathcal{Q},\mathcal{L}}}](\bar{x})$.

Example 11. Consider again the database instance of Example 2. For convenience we abbreviate the CarOwners predicate as *CarO* and its tuples as $(PS, M, Q1)$, $(JS, V, B1)$, and $(MC, B, B2)$ and the Location predicate as *Loc* and its tuples as (PS, Qs), (JS, Bx), and (MC, Bx).

Assume we have the following two local closed world assumptions.

(1) $\mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx))$, expressing that the database contains all car owners living in the Bronx, and
(2) $\mathcal{LCWA}(Loc(n, l), (l = Bx \wedge \exists m, id : CarO(n, m, id))$, expressing that the database knows all people living in the Bronx that own a car.

Our interest is in what is certain and possible for the *CarO* relation. There are two equivalent ways to express this. The first one uses the set of certain rules (in the rules that follow, implicit universal quantification is assumed):

$$\Delta^c_{\mathcal{Q},\mathcal{L}} = \left\{ \begin{array}{l} Q^c(n, m, id) \leftarrow CarO^c(n, m, id). \\ Q^{c\neg}(n, m, id) \leftarrow CarO^{c\neg}(n, m, id). \\ CarO^c(n, m, id) \leftarrow CarO(n, m, id). \\ CarO^{c\neg}(n, m, id) \leftarrow \neg CarO(n, m, id) \wedge Loc^c(n, Bx). \\ Loc^c(n, l) \leftarrow Loc(n, l). \\ Loc^{c\neg}(n, l) \leftarrow \neg Loc(n, l) \wedge l = Bx \wedge \exists m, id : CarO^c(n, m, id). \end{array} \right\}$$

Interestingly, this rule set is nonrecursive. It follows that its least and greatest fixpoint coincide. Also, the query $[\mathbf{lfp}_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](n, m, id)$ returns all elements of Q^c, and this relation can be computed by simply unfolding its rule body. By two unfolding steps, $Q^c(n, m, id)$ is rewritten to the database query $CarO(n, m, id)$.

Thus, the certain answer is, not surprisingly, the set of database tuples of $CarO$.

$$\{(PS, M, Q1), (JS, V, B1), (MC, B, B2)\}$$

Similarly, to obtain the certainly false tuples for the query, we unfold the fixpoint expression $[\mathbf{lfp}_{Q^{c\neg}, \Delta^c_{\mathcal{Q},\mathcal{L}}}](n, m, id)$, yielding the database query $\neg CarO(n, m, id) \wedge Loc(n, Bx)$. Its answer is the following set:

$$\{(JS, m, id) \mid (m, id) \neq (V, B1)\} \cup \{(MC, m, id) \mid (m, id) \neq (B, B2)\},$$

which, for the two inhabitants of the Bronx, contains all tuples not in the database. Finally, the set of possible but uncertain answers is the complement of the union of the certain answers and the certainly false answers. This set is specified by the query $\neg CarO(n, m, id) \wedge (CarO(n, m, id) \vee \neg Loc(n, Bx))$, which can be simplified to $\neg CarO(n, m, id) \wedge \neg Loc(n, Bx)$. Its answer is the set

$$\{(PS, m, id) \mid (m, id) \neq (M, Q1)\}.$$

Indeed, as Peter Steward does not live in the Bronx, we do not know whether he owns other cars. Moreover, the database does not contain the integrity constraints that a license plate is used for only one car, hence he can own any of the models in the domain while the license plate can also be any of those in the domain.

Alternatively, to compute the previous sets we could also use the set of possible rules:

$$\Delta^p_{\mathcal{Q},\mathcal{L}} = \left\{ \begin{array}{l} Q^p(n, m, id) \leftarrow CarO^p(n, m, id). \\ Q^{p\neg}(n, m, id) \leftarrow CarO^{p\neg}(n, m, id). \\ CarO^p(n, m, id) \leftarrow CarO(n, m, id) \vee Loc^{p\neg}(n, Bx). \\ CarO^{p\neg}(n, m, id) \leftarrow \neg CarO(n, m, id). \\ Loc^p(n, l) \leftarrow Loc(n, l) \vee l \neq Bx \vee \forall m, id : CarO^{p\neg}(n, m, id). \\ Loc^{p\neg}(n, l) \leftarrow \neg Loc(n, l). \end{array} \right\}$$

Again, this is a nonrecursive rule set. The expression $[\mathbf{gfp}_{Q^p, \Delta^p_{\mathcal{Q},\mathcal{L}}}](n, m, id)$ can be unfolded to the database query $CarO(n, m, id) \vee \neg Loc(n, Bx)$, which is indeed the negation of database query solving $[\mathbf{lfp}_{Q^{c\neg}, \Delta^c_{\mathcal{Q},\mathcal{L}}}](n, m, id)$. Similarly, $[\mathbf{gfp}_{Q^{p\neg}, \Delta^p_{\mathcal{Q},\mathcal{L}}}](n, m, id)$ rewrites to $\neg CarO(n, m, id)$, the negation of the database query representing the certain answers of $CarO(n, m, id)$.

THEOREM 2. *Given $\mathfrak{D} = (D, \mathcal{L})$ a locally closed database over Σ, $\mathcal{Q}(\bar{x})$ an arbitrary query, and $(\mathcal{R}^c_{\mathcal{Q}}, \mathcal{R}^{c\neg}_{\mathcal{Q}}, \mathcal{R}^c_1, \mathcal{R}^{c\neg}_1, \ldots, \mathcal{R}^c_n, \mathcal{R}^{c\neg}_n)$ the relations defined by $\Delta^c_{\mathcal{Q},\mathcal{L}}$ in D. It holds that:*

—$\mathcal{R}^c_i = \{\bar{d} \mid P_i(\bar{d})^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}\}$, *for each* $1 \leq i \leq n$,
—$\mathcal{R}^{c\neg}_i = \{\bar{d} \mid P_i(\bar{d})^{\mathcal{C}_\mathfrak{D}} = \mathbf{f}\}$, *for each* $1 \leq i \leq n$,
—$\mathcal{R}^c_{\mathcal{Q}} = \{\bar{d} \mid \mathcal{Q}(\bar{d})^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}\}$, *and*
—$\mathcal{R}^{c\neg}_{\mathcal{Q}} = \{\bar{d} \mid \mathcal{Q}(\bar{d})^{\mathcal{C}_\mathfrak{D}} = \mathbf{f}\}$.

PROOF. The proof is by comparison of the fixpoint computations of the operator $\mathcal{A}pp_\mathfrak{D}$ that constructs $\mathcal{C}_\mathfrak{D}$ and the operator $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$. The first one constructs

a sequence of three-valued $\sigma_{\mathfrak{D}}$-structures $\mathcal{K}_\alpha$, the second one a sequence of tuples of relations, or equivalently, a sequence of $\sigma'_{\mathfrak{D}}$-structures I_α where $\sigma'_{\mathfrak{D}}$ contains the new predicates P_i^c and $P_i^{c\neg}$. Note that the predicates Q^c, $Q^{c\neg}$ have no recursive occurrences in the bodies of $\Delta^c_{\mathcal{Q},\mathcal{L}}$, and hence, do not influence the construction of the P_i^c's and the $P_i^{c\neg}$'s. Hence, for the moment, allow us to ignore them.

We will show that each I_α *simulates* $\mathcal{K}_\alpha$ in the sense of Section 4.1. It is obvious that I_0 simulates $\mathcal{K}_0$, so it suffices to show that applying $\mathcal{A}pp_{\mathfrak{D}}$ and $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ preserves this property. Assume that I simulates $\mathcal{K}$ and $I' = \Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}(I), \mathcal{K}' = \mathcal{A}pp_{\mathfrak{D}}(\mathcal{K})$. We need to show that I' simulates $\mathcal{K}'$.

—For predicate P_i, $P_i(\bar{d})^{\mathcal{K}'} = \mathbf{t}$ iff $P_i(\bar{d}) \in D$ iff $P_i^c(\bar{d})^{I'} = \mathbf{t}$.
—Since I simulates $\mathcal{K}$, $\Psi_{P_i}[\bar{d}]^{\mathcal{K}} = \mathbf{t}$ iff $(\Psi^c_{P_i}[\bar{d}])^I = \mathbf{t}$. Therefore, $P_i(\bar{d})^{\mathcal{K}'} = \mathbf{f}$ iff $P_i(\bar{d}) \notin D$ and $\Psi_{P_i}[\bar{d}]^{\mathcal{K}} = \mathbf{t}$ iff $P_i(\bar{d}) \notin D$ and $(\Psi^c_{P_i}[\bar{d}])^I = \mathbf{t}$ iff $P_i^{c\neg}(\bar{d})^{I'} = \mathbf{t}$.

It follows that I' indeed simulates $\mathcal{K}'$.

The consequence of this is that when $\mathcal{A}pp_{\mathfrak{D}}$ reaches a fixpoint, say at α (i.e., $\mathcal{K}_\alpha = \mathcal{C}_{\mathfrak{D}}$), then $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ reaches a fixpoint on all predicates P_i^c and $P_i^{c\neg}$ at α, and the fixpoint of $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ simulates $\mathcal{C}_{\mathfrak{D}}$, the fixpoint of $\mathcal{A}pp_{\mathfrak{D}}$. After one more iteration, $\Gamma_{\Delta^c_{\mathcal{Q},\mathcal{L}}}$ reaches a fixpoint also on the predicates Q^c and $Q^{c\neg}$. It holds that $\bar{d} \in \mathcal{R}^c_{\mathcal{Q}}$ iff $(\mathcal{Q}^c(\bar{d}))^{I_\alpha} = \mathbf{t}$ iff $(\mathcal{Q}(\bar{d}))^{\mathcal{K}_\alpha} = \mathbf{t}$ (by Proposition 16). Likewise, $\bar{d} \in \mathcal{R}^{c\neg}_{\mathcal{Q}}$ iff $(\mathcal{Q}(\bar{d}))^{\mathcal{K}_\alpha} = \mathbf{f}$. This finishes the proof. □

COROLLARY 3. $[\mathbf{lfp}_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x}) = \mathit{Cert}_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}(\bar{x}))$ *and* $[\mathbf{gfp}_{Q^p,\Delta^p_{\mathcal{Q},\mathcal{L}}}](\bar{x}) = \mathit{Poss}_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}(\bar{x}))$.

The proof of the previous theorem shows that a naive bottom-up computation of the fixpoint query $[\mathbf{lfp}_{Q^c,\Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ boils down to the same as first computing $\mathcal{C}_{\mathfrak{D}}$ and then evaluating $\mathcal{Q}(\bar{x})$. In fact, the worst-case complexity of this method is not better than that of the naive method. Of course, in practice, it is typically not necessary to compute all relations: it suffices to compute the relations that are relevant for the query. Moreover, goal directed methods such as unfolding (see next section), magic sets [Bancilhon et al. 1986], or tabling [Swift 1999] will often construct only fractions of those relations. This was illustrated in Example 11. On the other hand, if the database is relatively stable, in the sense that many queries are to be solved with respect to the same database, it might prove a good idea to compute $\mathcal{C}_{\mathfrak{D}}$ after all, and pose all queries directly against it. The gain of computing $\mathcal{C}_{\mathfrak{D}}$ grows with the following parameters:

(1) the number of queries that can be answered with it (which is dictated by the ratio of queries posed to the database over the number of updates performed to it),
(2) the size of the windows of expertise and the complexity of the dependency relations, for these determine to a great extent the complexity of the fixpoint queries.

Note that, after an update of D, we must update (the simulation of) $\mathcal{C}_{\mathfrak{D}}$ but this can be done by applying incremental methods for *view updating* on the rule set $\Delta^c_{\mathcal{L}}$.

A special case arises when $\Delta^c_{\mathcal{Q},\mathcal{L}}$ is nonrecursive. This is studied in the next section.

4.5 Hierarchically Closed Databases

Definition 18 (*Hierarchically Closed Database*). Given a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ over Σ. The *LCWA dependency graph* of $\mathcal{L}$ is the directed graph on Σ, containing a directed edge from predicate Q to P iff there exists $\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}]) \in \mathcal{L}$ such that Q occurs negatively in Ψ. A *hierarchically closed* database $\mathfrak{D}$ is one in which the LCWA dependency graph is cycle-free.

By inspection of the constructions of $\Delta^c_{\mathcal{Q},\mathcal{L}}$ and $\Delta^c_{\mathcal{Q},\mathcal{L}}$ in Definitions 16 and 17, the next proposition follows.

PROPOSITION 24. *A locally closed database $\mathfrak{D}$ is hierarchically closed iff for arbitrary query $\mathcal{Q}(\bar{x})$, both the set of certain rules $\Delta^c_{\mathcal{Q},\mathcal{L}}$ and the set of possible rules $\Delta^p_{\mathcal{Q},\mathcal{L}}$ are nonrecursive.*

An important observation is that positive occurrences of predicates in windows of expertise do not lead to cycles in the LCWA-dependency graph or to recursion in the query definitions $\Delta^c_{\mathcal{Q},\mathcal{L}}$ and $\Delta^p_{\mathcal{Q},\mathcal{L}}$. The relevance of this is that, in practical applications, recursive $\Delta^p_{\mathcal{Q},\mathcal{L}}$ will likely be quite rare. For example, if we reconsider the examples of the Computer Science telephone database or the database of the traffic tax administration, we see a pattern that such databases are specialized in a specific topic: members of the department, or car owners in the county. The predicates representing these topics yield the windows of expertise of other database predicates. These predicates occur positively in the windows of expertise, and moreover, windows of expertise are relatively simple formulas. If this pattern is general, as we expect, then locally closed databases with cyclic LCWA-dependency graphs will be rare indeed.

The observation is important for several reasons. One is that, as shown in the next section, for hierarchically closed world databases, the constructed $\mathcal{C}_{\mathfrak{D}}$ will often coincide with the optimal approximation $\mathcal{O}_{\mathfrak{D}}$. Second, $\Delta^c_{\mathcal{Q},\mathcal{L}}$ and $\Delta^p_{\mathcal{Q},\mathcal{L}}$ have a unique fixpoint and hence, least and greatest (and partial) fixpoints coincide. In particular, both the certain answer query $[\mathbf{lfp}_{Q^c, \Delta^c_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ and the (alternative) possible answer query $[\mathbf{gfp}_{Q^p, \Delta^p_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ can be transformed in equivalent first-order queries by unfolding the definitions of all predicates. Next we represent a rewrite algorithm that takes as input a query $\mathcal{Q}(\bar{x})$ and transforms it into queries for certain or possible answers that can be posed directly against D.

Example 12. Let $\Sigma = \{P, Q\}$ and $\mathfrak{D} = (D, \mathcal{L})$ where

$$D = \{P(a), Q(c)\}, \quad \mathcal{L} = \{\mathcal{LCWA}(P(x), Q(x)),\ \mathcal{LCWA}(Q(x), x{=}c)\}.$$

$\mathfrak{D}$ is clearly hierarchically closed. Now, we consider Algorithm 1 for the query $\mathcal{Q}(x) = P(x)$.

Algorithm 1 : Query Rewriting Function R^c (resp., R^p)

1: **Input:** a set $\mathcal{L}$ of LCWA's of a hierarchically closed database and a query $\mathcal{Q}(\bar{x})$ over Σ.
2: Define $R^c(\mathcal{Q}(\bar{x}))$ and $R^p(\mathcal{Q}(\bar{x}))$ by simultaneous induction on the structure of $\mathcal{Q}(\bar{x})$ as follows:
If P is $\mathbf{t}, \mathbf{f}$ or $=$ then $R^c(P(\bar{t})) := R^p(P(\bar{t})) := P(\bar{t})$.
Else
—$R^c(P(\bar{t})) := P(\bar{t})$.
—$R^p(P(\bar{t})) := (P(\bar{t}) \vee \neg R^c(\Psi_P(\bar{t}))$.
—$R^c(\neg\varphi) := \neg R^p(\varphi)$.
—$R^p(\neg\varphi) := \neg R^c(\varphi)$.
—R^c and R^p distribute over $\wedge, \vee, \exists, \forall$.
3: **Output:** a query $R^c(\mathcal{Q}(\bar{x}))$ (resp., $R^p(\mathcal{Q}(\bar{x}))$) over Σ.

—For certain query answers, we evaluate $R^c(P(x)) = P(x)$ with respect to D. The answer of this query is $\{a\}$.
—For possible answers we start with $R^p(P(x)) = (P(x) \vee \neg R^c(Q(x))$. After evaluating $R^c(Q(x))$ we get $P(x) \vee \neg Q(x)$. Evaluating this expression with respect to D we obtain $Dom^D \setminus \{c\}$.

These sets are exactly the certain and possible answers of the query $P(x)$ in $\mathfrak{D}$.

Example 13. The procedures R^c and R^p correspond to the unfolding process as mentioned in Example 11. Not surprisingly, if we apply them to the query $CarO(n, m, id)$ of Example 11, we obtain

$$R^c(CarO(n, m, id)) = CarO(n, m, id),$$
$$R^p(CarO(n, m, id)) = CarO(n, m, id) \vee \neg Loc(n, Bx)$$

which are exactly the queries obtained in the example.

THEOREM 3 (SOUNDNESS OF R^c AND R^p). *The functions R^c and R^p are well-defined. For each query $\mathcal{Q}(\bar{x})$, $\{\bar{d} \mid D \models R^c(Q[\bar{d}])\} = Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x}))$ and $\{\bar{d} \mid D \models R^p(Q[\bar{d}])\} = Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x}))$.*

PROOF SKETCH. It is not difficult to see that the operation $R^c(\mathcal{Q}(\bar{x}))$ basically simulates the process of, first, the construction of $\Delta^c_{\mathcal{Q},\mathcal{L}}$ from $\mathcal{L}$ and $\mathcal{Q}(\bar{x})$, and second, the subsequent unfolding of the atom $Q^c(\bar{x})$. In the construction of $\Delta^c_{\mathcal{Q},\mathcal{L}}$, all positive occurrences of database predicates P_i in $\mathcal{Q}(\bar{x})$ are first replaced by P_i^c; in the unfolding of $Q^c(\bar{x})$, these occurrences are rereplaced by P_i. Correspondingly, R^c preserves positive occurrences of database predicates. In the construction of $\Delta^c_{\mathcal{Q},\mathcal{L}}$, all negative occurrences of database predicates $P_i(\bar{t})$ in $\mathcal{Q}(\bar{x})$ are replaced by $\neg P_i^{c\neg}(\bar{t})$, and in the unfolding phase, these negative literals are replaced by $P_i(\bar{t}) \vee \neg\Psi^c_{P_i}[\bar{t}]$. As for the rewrite procedure, R^c calls R^p for negative occurrences of database predicates, and this procedure replaces such negative occurrences by the corresponding formula $P_i(\bar{t}) \vee \neg R^c(\Psi_{P_i}[\bar{t}])$.

By iterating the preceding argument, we see that $R^c(\Psi_{P_i}[\bar{t}])$ is the result of unfolding $\neg P_i^{c\neg}(\bar{t})$.

Similarly, we have that $R^p(\mathcal{Q}(\bar{x}))$ simulates the construction of $\Delta^p_{\mathcal{Q},\mathcal{L}}$ and the subsequent unfolding of the atom $Q^p(\bar{x})$. Given the correctness and termination of the unfolding process in nonrecursive definitions, the soundness of R^c and R^p follows from Theorem 2. □

PROPOSITION 25. *Algorithm* 1 *produces a query whose size is:*[10]

—*constant in the size of the database D;*
—*linear in the size of the original query Q, where the linear factor C is:*
  —*polynomial in the maximal size of the windows of expertise;*
  —*exponential in the depth of the dependency relation between relations (i.e., the rank of the highest predicate).*

PROOF. The size of a rewritten query is obviously constant in $|D|$ since R^c and R^p do not depend on D. Termination of the rewriting process implemented by R^c and R^p follows from the fact that $\mathfrak{D}$ is hierarchically closed. Let us define the rank of a database predicate P_i as the longest path in the LCWA-dependency graph from one of $\mathbf{t}, \mathbf{f}$ to P_i. Reducing an atom of rank n to a formula with atoms of at most rank $n-1$ blows up the atom with a factor less than W, the size of the largest window of expertise. Continuing the reduction until the end blows up the atom with a factor W^n where n is the highest rank of a database predicate. So, a database atom in $\mathcal{Q}(\bar{x})$ is blown up to a formula of maximally W^n. This is the factor C. □

As a corollary, we obtain that the algorithm retains tractability in the size of the database. Since solving first-order queries is AC^0 in data complexity [Libkin 1995], we obtain the following result.

THEOREM 4 (COMPLEXITY). *Given a hierarchically closed database $\mathfrak{D}$ and a query $\mathcal{Q}$, the computation time of $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x}))$ and $Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x}))$ by Algorithm* 1 *is AC^0 in data complexity.*

4.6 Towards Approximate Query Answering with Integrity Constraints

So far, local closed world assumptions were the only source of negative information (i.e., of provably false database atoms). In practice, another important source of negative information are integrity constraints, functional dependency constraints in particular. Assume that for the n^{th} argument position of n-ary predicate P functionally depends on the argument positions $(1, \ldots, n-1)$. In that case, the presence of the atom $P(a_1, \ldots, a_{n-1}, a)$ in the database entails the falsity of any atom $P(a_1, \ldots, a_{n-1}, b)$ for which $b \neq a$.

In this section, we show how the approximate reasoning method can be extended for handling functional dependency constraints. For the rest of this section, let a locally closed database $\mathfrak{D}$ consist of a database instance D, a set of local closed world assumptions $\mathcal{L}$, and a set of functional dependency constraints F. The constraint that position l of predicate P functionally depends

[10]The size of a formula is expressed in number of atoms.

on argument positions $i_1, \ldots, i_k$ will be denoted $P : (i_1, \ldots, i_k) \to l$. The extended relational formula of such a constraint is the following first-order formula.

$$\forall x_1, \ldots, x_n, y_1, \ldots, y_n : P(x_1, \ldots, x_n) \wedge P(y_1, \ldots, y_n) \wedge x_{i_1} = y_{i_1} \wedge \cdots \wedge x_{i_k} = y_{i_k} \supset x_l = y_l$$

For such a database $\mathfrak{D}$ and query $\mathcal{Q}(\bar{x})$, we define $\Delta^c_{\mathcal{Q},\mathfrak{D}}$ as in Definition 16, except that for each $P : (i_1, \ldots, i_k) \to l \in F$, the following rule is added.

$$P^{c\neg}(x_1, \ldots, x_n) \leftarrow P^c(y_1, \ldots, y_n) \wedge x_{i_1} = y_{i_1} \wedge \cdots \wedge x_{i_1} = y_{i_1} \wedge x_l \neq y_l$$

This rule infers that $P(x_1, \ldots, x_n)$ is false if D contains an atom $P(y_1, \ldots, y_n)$ which is identical with the previous one on argument positions $i_1, \ldots, i_k$ and different on position l.

Example 14. In Example 11, each identification plate identifies the car, and hence the car brand. This is the constraint $CarO : 3 \to 2$. Its extended relational formula is given by the following formula:

$$\forall n, m, id, n_1, m_1 : CarO(n, m, id) \wedge CarO(n_1, m_1, id) \supset m = m_1$$

The rule set in the fixpoint expression can now be extended to

$$\Delta^c_{\mathcal{Q},\mathcal{L}} = \left\{ \begin{array}{l} Q^c(n, m, id) \leftarrow CarO^c(n, m, id). \\ Q^{c\neg}(n, m, id) \leftarrow CarO^{c\neg}(n, m, id). \\ CarO^c(n, m, id) \leftarrow CarO(n, m, id). \\ CarO^{c\neg}(n, m, id) \leftarrow \neg CarO(n, m, id) \wedge Loc^c(n, Bx). \\ Loc^c(n, l) \leftarrow Loc(n, l). \\ Loc^{c\neg}(n, l) \leftarrow \neg Loc(n, l) \wedge l = Bx \wedge \exists m, id : CarO^c(n, m, id). \\ \mathbf{CarO^{c\neg}}(\mathbf{n}, \mathbf{m}, id) \leftarrow CarO^{\mathbf{c}}(\mathbf{n_1}, \mathbf{m_1}, id) \wedge \mathbf{m} \neq \mathbf{m_1}. \end{array} \right\}$$

The last rule is the additional one. The two rules for $CarO^{c\neg}$ in this set can be replaced by the unique rule

$$\begin{array}{rl} CarO^{c\neg}(n, m, id) \leftarrow & \neg CarO(n, m, id) \wedge Loc^c(n, Bx) \vee \\ & \exists n_1, m_1 : CarO^c(n_1, m_1, id) \wedge m \neq m_1. \end{array}$$

Unfolding the query $\neg[\mathbf{lfp}_{Q^p, \Delta^p_{\mathcal{Q},\mathcal{L}}}](n, m, id)$ for the possible answers of $CarO(n, m, id)$ yields

$$(CarO(n, m, id) \vee Loc(n, Bx)) \wedge \forall n_1, m_1 : \neg CarO(n_1, m_1, id) \vee m = m_1.$$

Or equivalently

$$(CarO(n, m, id) \vee Loc(n, Bx)) \wedge \forall n_1, m_1 : CarO(n_1, m_1, id) \supset m = m_1.$$

In particular, for Peter Steward, the possible answers in the extended database are as follows.

$$\{(PS, M, Q1), (PS, V, B1), (PS, B, B2)\}$$

If, in addition, a car could belong to at most one person (i.e., $CarO : 3 \to 1$), the additional rule is

$$CarO^{c\neg}(n, m, id) \leftarrow CarO^c(n_1, m_1, id) \wedge n \neq n_1.$$

Possible answers for $CarO(PS, n, m)$ are now reduced to $\{(PS, M, Q1)\}$, that is, there is CWI on this query.

THEOREM 5. *Given a locally closed database $\mathfrak{D}$ and query $\mathcal{Q}(\bar{x})$. Let $\mathcal{C}$ denote the answers of $[\mathbf{lfp}_{Q^{+}, \Delta^{c}_{\mathcal{Q},\mathfrak{D}}}](\bar{x})$ in D, and $\mathcal{P}$ the answers of $\neg[\mathbf{lfp}_{Q^{\neg}, \Delta^{c}_{\mathcal{Q},\mathfrak{D}}}](\bar{x})$ in D. It holds that*

$$\mathcal{C} \subseteq Cert_{\mathfrak{D}}(\mathcal{Q}(\bar{x})) \subseteq Poss_{\mathfrak{D}}(\mathcal{Q}(\bar{x})) \subseteq \mathcal{P}.$$

PROOF SKETCH. It is easy to extend the fixpoint operator $\mathcal{A}pp_{\mathfrak{D}}$ to incorporate functional dependencies. The extended operator is defined as

$$P(\bar{a})^{\mathcal{K}'} = \begin{cases} \mathbf{t} & \text{if } P(\bar{d}) \in D, \\ \mathbf{f} & \text{if } P(\bar{d}) \notin D \text{ and there exists } \mathcal{LCWA}(P(\bar{x}), \Psi_P[\bar{x}]) \in \mathcal{L} \\ & \text{such that } \Psi_P[\bar{d}]^{\mathcal{K}} = \mathbf{t}, \\ \mathbf{f} & \text{if } P(\bar{d}) \notin D \text{ and } P : (i_1, \ldots, i_k) \to l \in \mathfrak{D} \text{ and} \\ & \text{for some } P(\bar{d})\text{: } P^{\mathcal{K}}(\bar{d}) = \mathbf{t} \text{ and } d_{i_1} = a_{i_1}, \ldots, d_{i_k} = a_{i_k} \text{ and } d_l \neq a_l, \\ \mathbf{u} & \text{otherwise.} \end{cases}$$

It is easy to extend the proof of Theorem 1 to show that this operator produces an approximation of all models of $\mathfrak{D}$ (which now contains also functional dependency constraints). To show that the extended fixpoint query is correct, one can show that the operator of the extended rule set simulates $\mathcal{A}pp_{\mathfrak{D}}$. This is a straightforward extension of the proof of Theorem 2. □

Bibliographic note. The use of approximations and fixpoint methods for the LCWA as well as the concept of hierachically closed databases were first introduced in Cortés-Calabuig et al. [2006], where also a naive algorithm for query answering was presented. Simulating approximate query answering by fixpoint queries and the related complexity results were first presented in Cortés-Calabuig et al. [2008]. The discussion on incorporating integrity constraints (Section 4.6) is novel.

5. ACCURACY OF APPROXIMATE QUERY ANSWERING

A weakness of the soundness Theorem 2 is that it gives no guarantee whatsoever on the accuracy of the returned answers of the approximate query answering method. As the next example illustrates, in certain "degenerated" cases, all accuracy in approximate query answering might be lost.

Example 15. Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database with a vocabulary σ. Assume that σ contains a unary predicate symbol P and the constant a but that D contains no tuples about P, and $\mathcal{L}$ contains for each database predicate Q (including P) the LCWA $\mathcal{LCWA}(Q(\bar{x}), P(a) \vee \neg P(a))$. Since this window of expertise is a tautology, this database has complete knowledge on all its predicates. Hence $\mathfrak{D}$ is equivalent with the standard database D under CWA. On the other hand, it is easy to see that $(P(a) \vee \neg P(a))^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$. It follows that for each atom $Q(\bar{d})$, we have $Q(\bar{d})^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$ whenever $Q(\bar{d}) \notin D$. This boils down to the fact that $\mathcal{C}_{\mathfrak{D}}$ represents the database under OWA. Many queries cannot be accurately answered using $\mathcal{C}_{\mathfrak{D}}$. For example, $(\neg Q(\bar{d}))^{\mathcal{C}_{\mathfrak{D}}} = \mathbf{u}$ whenever $Q(\bar{d}) \notin D$, although $\mathfrak{D}$ implies $\neg Q(\bar{d})$.

Example 15 justifies the need to investigate the accuracy of our methods of approximating query answers. We will investigate conditions on $\mathfrak{D}$ that guarantee that $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})) = \mathrm{Cert}_\mathfrak{D}(\mathcal{Q}(\bar{x}))$. We will not be concerned here with the accuracy of possible answers. Indeed, it follows from the next proposition that such conditions follow immediately from those for certain answers.

PROPOSITION 26. *For a locally closed database $\mathfrak{D}$ and a query $\mathcal{Q}(\bar{x})$, we have $Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})) = \mathrm{Poss}_\mathfrak{D}(\mathcal{Q}(\bar{x}))$ iff $Cert_{\mathcal{C}_\mathfrak{D}}(\neg\mathcal{Q}(\bar{x})) = \mathrm{Cert}_\mathfrak{D}(\neg\mathcal{Q}(\bar{x}))$.*

PROOF. By Definition 9, for a tuple $\bar{t}$ of constants of the appropriate arity, $\bar{t} \in Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x}))$ iff $\bar{t} \notin Cert_{\mathcal{C}_\mathfrak{D}}(\neg\mathcal{Q}(\bar{x}))$. Similarly, $\bar{t} \in \mathrm{Poss}_\mathfrak{D}(\mathcal{Q}(\bar{x}))$ iff $\bar{t} \notin \mathrm{Cert}_\mathfrak{D}(\neg\mathcal{Q}(\bar{x}))$. Hence, $Poss_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})) = \mathrm{Poss}_\mathfrak{D}(\mathcal{Q}(\bar{x}))$ iff $Cert_{\mathcal{C}_\mathfrak{D}}(\neg\mathcal{Q}(\bar{x})) = \mathrm{Cert}_\mathfrak{D}(\neg\mathcal{Q}(\bar{x}))$. □

In the remainder of this section, we will study when the accuracy of the approximate methods can be guaranteed. As the material concerning our analysis is rather technical, and since the more substantial results of this section build on a considerable amount of preliminary definitions, we put in the electronic appendix everything that is not essential for the formulation (and the understanding) of our main results, as well as the relevant proofs.

5.1 Dependencies and Modularity in Locally Closed Databases

The construction of $\mathcal{C}_\mathfrak{D}$ is through a propagation process over the windows of expertise. These formulas thus induce different sorts of dependencies between database predicates. From these, we can derive a modularity property that will prove useful in the accuracy analysis.

As defined in Definition 18, the LCWA-dependency graph of $\mathfrak{D}$ consists of edges (Q, P) such that Q occurs negatively in a window of expertise of P. Its transitive closure will be denoted $\prec^-$. If $Q \prec^- P$ we say that P depends negatively on Q. Recall from Proposition 24 that $\prec^-$ is cycle-free iff for any query $\mathcal{Q}(\bar{x})$, the set $\Delta^c_{\mathcal{Q},\mathcal{L}}$ (and also $\Delta^p_{\mathcal{Q},\mathcal{L}}$) is nonrecursive. The *full LCWA-dependency graph* is the graph on Σ consisting of edges (Q, P), such that Q occurs (positively or negatively) in a window of expertise of P. Its transitive closure is denoted $\prec$. If $Q \prec P$ we say that P depends on Q. A *strict hierarchically closed database* is one with a cycle-free full LCWA-dependency graph. Sometimes, given a query $\mathcal{Q}(\bar{x})$, it will be convenient to extend the LCWA-dependency graph with $\mathcal{Q}(\bar{x})$ by adding the query as a node and adding edges to it from each predicate P that occurs negatively in $\mathcal{Q}(\bar{x})$. Also the full LCWA-dependency graph can be extended in this way, but here edges are added from each predicate with an arbitrary occurrence in $\mathcal{Q}(\bar{x})$.

Example 16. The locally closed database of Example 11, with

$$\mathcal{L} = \left\{ \begin{array}{l} \mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx)), \\ \mathcal{LCWA}(Loc(n, l), (l = Bx \wedge \exists n, m : CarO(n, m, id)) \end{array} \right\}$$

is hierarchically closed but not strict hierarchically closed.

Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database. Let $\mathcal{P}$ be a subset of Σ which is $\prec$-downward closed, that is, if $P \in \mathcal{P}$ and $R \prec P$ then $R \in \mathcal{P}$. Note that

predicates of $\mathcal{P}$ may occur in the windows of expertise of predicates $Q \in \Sigma \setminus \mathcal{P}$ but not vice versa. For such a set, it makes sense to consider the restriction of $\mathfrak{D}$ to the predicates in $\mathcal{P}$.

Notation 3. The pair $(D|_{\mathcal{P}}, \mathcal{L}|_{\mathcal{P}})$ consisting of the restrictions of the structure D to the predicates in $\mathcal{P}$ and the restriction of $\mathcal{L}$ to the LCWA's of the predicates in $\mathcal{P}$ is denoted $\mathfrak{D}|_{\mathcal{P}}$. Clearly, $\mathfrak{D}|_{\mathcal{P}}$ is a locally closed database with the same domain as $\mathfrak{D}$. The $\prec$-downward closed sets $\{Q \mid \forall P \in \mathcal{P}\ Q \prec P\}$ and $\mathcal{P} \cup \{Q \mid \forall P \in \mathcal{P}\ Q \prec P\}$ are denoted $\prec \mathcal{P}$ and $\prec_{\cup} \mathcal{P}$, respectively.

LEMMA 1 (EXTENDIBILITY). *If $\mathcal{P}$ is a $\prec$-downward closed set of predicates, then each model M of $\mathfrak{D}|_{\mathcal{P}}$ can be expanded to a model N of $\mathfrak{D}$ by setting $Q^N = Q^D$, for every $Q \in \Sigma \setminus \mathcal{P}$.*

PROOF. Let M and N be as before. Clearly, N satisfies all database atoms, all LCWA's of $\mathfrak{D}|_{\mathcal{P}}$, and all LCWA's for predicates $Q \in \Sigma \setminus \mathcal{P}$ (since $N \models \forall \bar{x} : (Q(\bar{x}) \supset Q(\bar{x}) \in D)$). Hence, $N \models \mathfrak{D}$. □

Lemma 1 allows us to derive some interesting modularity properties of query answering.

PROPOSITION 27. *For any query $\mathcal{Q}(\bar{x})$, we have*

$$Cert_{\mathfrak{D}}(\mathcal{Q}(\bar{x})) = Cert_{\mathfrak{D}|_{\prec \mathcal{Q}}}(\mathcal{Q}(\bar{x})),$$
$$Poss_{\mathfrak{D}}(\mathcal{Q}(\bar{x})) = Poss_{\mathfrak{D}|_{\prec \mathcal{Q}}}(\mathcal{Q}(\bar{x})).$$

PROOF. It follows from the extendibility Lemma 1 that for any $\prec$-downward closed set $\mathcal{P}$ and any $\mathcal{P}$-formula φ, $\mathfrak{D} \models \varphi$ iff $\mathfrak{D}|_{\mathcal{P}} \models \varphi$, and $\mathfrak{D} \cup \{\varphi\}$ is satisfiable iff $\mathfrak{D}|_{\mathcal{P}} \cup \{\varphi\}$ is satisfiable. Both equations follow immediately. □

As a corollary to this proposition, we have the following proposition on CWI.

COROLLARY 4. *$\mathfrak{D}$ has CWI on $\mathcal{Q}(\bar{x})$ iff $\mathfrak{D}|_{\prec \mathcal{Q}(\bar{x})}$ has CWI on $\mathcal{Q}(\bar{x})$.*

PROPOSITION 28. *For any query $\mathcal{Q}(\bar{x})$, we have that $\mathcal{O}_{\mathfrak{D}}|_{\prec \mathcal{Q}} = \mathcal{O}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}$ and $\mathcal{C}_{\mathfrak{D}}|_{\prec \mathcal{Q}} = \mathcal{C}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}$.*

PROOF. The first equation is an easy consequence of the extendibility Lemma 1. For the second, observe that in computing the predicates of $\mathcal{P}$, the operator $\mathcal{A}pp_{\mathfrak{D}}$ only uses $\mathfrak{D}|_{\mathcal{P}}$. □

Note 7. A direct consequence of Proposition 28 is that $Cert_{\mathcal{O}_{\mathfrak{D}}}(\mathcal{Q}(\bar{x})) = Cert_{\mathcal{O}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}}(\mathcal{Q}(\bar{x}))$ and $Cert_{\mathcal{C}_{\mathfrak{D}}}(\mathcal{Q}(\bar{x})) = Cert_{\mathcal{C}_{(\mathfrak{D}|_{\prec \mathcal{Q}})}}(\mathcal{Q}(\bar{x}))$.

The last propositions show that the accuracy of approximate answers for a query $\mathcal{Q}(\bar{x})$ can be studied in the potentially much smaller database $\mathfrak{D}|_{\prec \mathcal{Q}}$.

5.2 Sufficient Conditions for Assuring Completeness

In this section we specify conditions for assuring that an approximation of query answers is precise. A basic concept of this accuracy analysis is that of *squared queries*. A query is squared with respect to a database $\mathfrak{D}$ when all its certain answers are true in the optimal approximation $\mathcal{O}_{\mathfrak{D}}$. We define this

concept more in general for theories containing UNA $\wedge$ DCA. Formally, we have the following.

Definition 19 (*Squared Queries*). Let Γ be a satisfiable first-order theory over σ containing $\mathsf{DCA}(\sigma) \wedge \mathsf{UNA}(\sigma)$, and let HU be the Herbrand universe for σ. We say that a query $\mathcal{Q}(\bar{x})$ of arity n is *squared* in Γ if for every $\bar{d} \in HU^n$, if $\Gamma \models \mathcal{Q}(\bar{d})$ then $\mathcal{Q}(\bar{d})^{\mathcal{O}_\Gamma} = \mathbf{t}$.

Since $\mathcal{Q}(\bar{d})^{\mathcal{O}_\Gamma} = \mathbf{t}$ entails that $\Gamma \models \mathcal{Q}(\bar{d})$, the query $\mathcal{Q}(\bar{x})$ is squared iff the stronger condition holds that for every $\bar{d} \in HU^n$, $\Gamma \models \mathcal{Q}(\bar{d})$ iff $\mathcal{Q}(\bar{d})^{\mathcal{O}_\Gamma} = \mathbf{t}$. Yet another way to define that $\mathcal{Q}(\bar{x})$ is squared is that for arbitrary $\bar{d} \in HU^n$, $\mathcal{Q}(\bar{d})^{\mathcal{O}_\Gamma} = \mathbf{u}$ implies that $\Gamma \cup \{\neg\mathcal{Q}(\bar{d})\}$ is satisfiable. However, this does not mean that $\Gamma \cup \{\mathcal{Q}(\bar{d})\}$ is satisfiable for each $\bar{d} \in HU^n$. For instance, $P(x) \wedge \neg P(x)$ is squared in the empty $\mathfrak{D}$ but $P(d) \wedge \neg P(d)$ is not satisfiable.

The following proposition shows that squaredness is the key concept for the study accuracy of certain answers.

PROPOSITION 29. *The query $\mathcal{Q}(\bar{x})$ is squared in Γ iff $Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}(\bar{x})) = Cert_\Gamma(\mathcal{Q}(\bar{x}))$. The query $\neg\mathcal{Q}(\bar{x})$ is squared in Γ iff $Poss_{\mathcal{O}_\Gamma}(\mathcal{Q}(\bar{x})) = Poss_\Gamma(\mathcal{Q}(\bar{x}))$.*

PROOF. By Proposition 20, $Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}(\bar{x})) \subseteq Cert_\Gamma(\mathcal{Q}(\bar{x}))$. Vice versa, if $\bar{d} \in Cert_\Gamma(\mathcal{Q}(\bar{x}))$, then $\Gamma \models \mathcal{Q}(\bar{d})$. By squaredness, $\mathcal{Q}(\bar{d})^{\mathcal{O}_\Gamma} = \mathbf{t}$, hence $\bar{d} \in Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}(\bar{x}))$. □

The previous above proposition shows that the replacement of a locally closed database by its optimal approximation induces no loss of certain answers on squared queries. The next one shows that the approximation need not to be optimal in all predicates when the theory has a least model I. When the latter is the case, observe that for any atom $P(\bar{a})$, $P(\bar{a})^I = \mathbf{t}$ iff $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{t}$.

Notation 4. We write $P \in^- \mathcal{Q}(\bar{x})$ if P has a negative occurrence in $\mathcal{Q}(\bar{x})$ and $P \in^+ \mathcal{Q}(\bar{x})$ if P has a positive occurrence in $\mathcal{Q}(\bar{x})$.

PROPOSITION 30. *Let $\mathcal{Q}(\bar{x})$ be squared in Γ. Assume that Γ has a least model I and let $\mathcal{K}$ be an approximation of Γ such that for each atom $P(\bar{a})$, $P(\bar{a})^I = \mathbf{t}$ iff $P(\bar{a})^{\mathcal{K}} = \mathbf{t}$. If $P^{\mathcal{K}} = P^{\mathcal{O}_\Gamma}$ for every predicate $P \in^- \mathcal{Q}(\bar{x})$, then $Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x})) = Cert_\Gamma(\mathcal{Q}(\bar{x}))$.*

PROOF. Since $\mathcal{K} \leqslant_p \mathcal{O}_\Gamma$, we have that $Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x})) \subseteq Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}(\bar{x})) = Cert_\Gamma(\mathcal{Q}(\bar{x}))$. To prove the other direction, let us observe that all predicates such that $P^{\mathcal{K}} \neq P^{\mathcal{O}_\Gamma}$ occur only positively in $\mathcal{Q}(\bar{x})$ and moreover $P^{\mathcal{O}_\Gamma} \leq P^{\mathcal{K}}$ (where $\leq$ is the truth order, not the precision order). Indeed, for any atom $P(\bar{a})$, if $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{u}$, then since $\mathcal{K} \leqslant_p \mathcal{O}_\Gamma$, we have $P(\bar{a})^{\mathcal{K}} = \mathbf{u}$, and if $P(\bar{a})^{\mathcal{O}_\Gamma} = \mathbf{t}$, then $P(\bar{a})^I = \mathbf{t} = P(\bar{a})^{\mathcal{K}}$. It follows from a standard monotonicity property of standard first-order logic that for arbitrary $\bar{d}$, $\mathcal{Q}(\bar{d})^{\mathcal{O}_\Gamma} \leq \mathcal{Q}(\bar{d})^{\mathcal{K}}$. Hence $Cert_{\mathcal{K}}(\mathcal{Q}(\bar{x})) \supseteq Cert_{\mathcal{O}_\Gamma}(\mathcal{Q}(\bar{x}))$. □

The proposition expresses that we can compute certain answers for $\mathcal{Q}(\bar{x})$ using a weaker than optimal approximation $\mathcal{K}$ which is unknown on atoms $P(\bar{a})$ that are false in $\mathcal{O}_\Gamma$, as long as the predicates P occur only positively in the query.

As the following theorem shows, squaredness may guarantee maximal accuracy of approximate query answering.

THEOREM 6 (COMPLETENESS). *It holds that* $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})) = Cert_\mathfrak{D}(\mathcal{Q}(\bar{x}))$ *if* $\mathcal{Q}(\bar{x})$ *is squared in locally closed database* $\mathfrak{D}$ *and* $P^{\mathcal{C}_\mathfrak{D}} = P^{\mathcal{O}_\mathfrak{D}}$*, for every database predicate* $P \in^- \mathcal{Q}(\bar{x})$.

PROOF. $\mathfrak{D}$ has a least model, namely D; moreover, by construction, $P(\bar{a})^{\mathcal{C}_\mathfrak{D}} = \mathbf{t}$ iff $P(\bar{a}) \in D$. Hence Proposition 30 applies and the theorem follows. □

This theorem gives a condition under which the use of an approximation avoids the loss of certain answers. It points to two potential reasons why precision can get lost: the query can be nonsquared, or the computed approximation can be too imprecise. The latter is clearly the case in Example 15. This is a database with CWA and each query is squared with respect to it. On the other hand, for each $P(\bar{a}) \notin D$, the query $\neg P(\bar{a})$ is inaccurately answered since $(\neg P(\bar{a}))^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$ while $\mathfrak{D} \models \neg P(\bar{a})$. A case where not the inaccuracy of $\mathcal{C}_\mathfrak{D}$ is the problem is the query $P(\bar{a}) \vee \neg P(\bar{a})$. This tautological query is entailed by any database $\mathfrak{D}$ while it evaluates to unknown whenever $P(\bar{a})^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$.

The theorem thus raises two further questions. Firstly, when is a query squared? Secondly, when is the approximation $\mathcal{C}_\mathfrak{D}$, as defined in Section 4, optimal with respect to some predicate? We start with investigating the first question.

PROPOSITION 31. *The data complexity of deciding whether a query is squared in a locally closed database* $\mathfrak{D}$ *is in* Δ_2^P *and is* NP*-hard for some queries.*

PROOF. To verify squaredness, we need to compute $\mathcal{O}_\mathfrak{D}$ and, for all $\bar{d} \in HU^n$, $\mathcal{Q}[\bar{d}]^{\mathcal{O}_\mathfrak{D}}$ and whether $\mathfrak{D} \models \mathcal{Q}[\bar{d}]$. $\mathcal{O}_\mathfrak{D}$ can be constructed with a polynomial number of calls to an NP-oracle. Then for a polynomial number of tuples $\bar{d} \in HU^n$, we have an NP-call to decide whether $\mathfrak{D} \models \mathcal{Q}[\bar{d}]$ and we can compute $\mathcal{Q}[\bar{d}]^{\mathcal{O}_\mathfrak{D}}$ in polynomial time. Hence, the problem is in Δ_2^P.

To prove that there are NP-hard instances, consider again the class of locally closed databases $\mathfrak{D} = (D, \mathcal{L})$ from Proposition 13. In each, $Edge^D$ is a graph and $Kernel^D$ and P^D are empty. *Edge* is a base predicate (i.e., its window of expertise is $\mathbf{t}$), *Kernel* is open (i.e., its window of expertise is $\mathbf{f}$), and the databases have complete knowledge on $P(c)$ if *Kernel* is not a kernel of *Edge*. It follows that $\mathfrak{D} \models \neg P(c)$ iff the graph has no kernel. On the other hand, $P(c)$ is always unknown in the optimal approximation. Indeed, $\neg P(c)$ is true in D. On the other hand, the structure D' obtained from D by setting $P(c)$ to true is also a model of $\mathfrak{D}$, since $Kernel^{D'} = \emptyset$ is not a kernel. Consequently, $\neg P(c)$ is squared iff $Edge^D$ has no kernel. Hence, to decide whether $\neg P(c)$ is squared for this class of databases is an NP-complete problem. □

This negative results motivates a deeper analysis and the identification of sufficient conditions for squaredness. The following proposition is a direct consequence of the definition of squaredness and the properties of the conjunction operator.

PROPOSITION 32. *A literal is squared in any theory. If $\mathcal{Q}(\bar{x})$ is a conjunction of formulas squared in Γ, then $\mathcal{Q}(\bar{x})$ is squared in Γ. In particular, a conjunction of literals is squared in any theory.*

PROOF. If $\mathcal{Q}[\bar{x}]$ is a literal and $\Gamma \models \mathcal{Q}[\bar{d}]$, that is, $\mathcal{Q}[\bar{d}]^I = \mathbf{t}$ for every two-valued Herbrand model I of Γ, then $\mathcal{Q}[\bar{d}]^{\mathcal{O}_\Gamma} = \mathbf{t}$ by definition of $\mathcal{O}_\Gamma$. The rest of the proposition follows from the fact that if $\Gamma \models \psi \wedge \phi$ then $\Gamma \models \psi$ and $\Gamma \models \phi$. □

A more general syntactic condition for squaredness in locally closed databases (i.e., when $\Gamma = \mathfrak{D}$) can be formulated using the following notion.

Definition 20 (*Mutually Exclusive Conjunctions*). A set of conjunctions $\{C_1[\bar{x}], \ldots, C_n[\bar{x}]\}$ is *mutually exclusive* with respect to a three-valued structure $\mathcal{K}$ if for each $\bar{d} \in HU^n$, at most one $C_i[\bar{d}]$ is not false in $\mathcal{K}$.

Definition 21 (*Positive Free Predicates*). We call a predicate P *negatively bound* in query $\mathcal{Q}(\bar{x})$ if P occurs negatively in $\mathcal{Q}(\bar{x})$ or a predicate occurring negatively in $\mathcal{Q}(\bar{x})$ depends on P. Otherwise, P is called *positive free* in $\mathcal{Q}(\bar{x})$. That is, $P \notin^- \mathcal{Q}(\bar{x})$ and there is no $R \in^- \mathcal{Q}(\bar{x})$ such that $P \prec R$. The sets of negatively bound predicates and of positive free predicates are denoted $\mathcal{N}eg\mathcal{B}(\mathcal{Q}(\bar{x}))$ and $\mathcal{P}os\mathcal{F}(\mathcal{Q}(\bar{x}))$, respectively.

THEOREM 7 (SQUAREDNESS). *A query $\mathcal{Q}(\bar{x})$ is squared in a locally closed database $\mathfrak{D}$ when it is of the form $\forall \bar{y} : (C_1 \vee \cdots \vee C_n)$, where each C_i is a conjunction such that: (i) each nonliteral conjunct C_{i_k} of the C_i consists of predicates which are either positive free in $\mathcal{Q}(\bar{x})$ or two-valued in $\mathcal{O}_\mathfrak{D}$, and (ii) the set of conjunctions is mutually exclusive with respect to $\mathcal{O}_\mathfrak{D}$.*

PROOF. See the electronic appendix. □

Now we turn our attention to the second question, the optimality of the approximation $\mathcal{C}_\mathfrak{D}$ with respect to some predicate P.

THEOREM 8 (OPTIMALITY). *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database. It holds that $P^{\mathcal{C}_\mathfrak{D}} = P^{\mathcal{O}_\mathfrak{D}}$ if $\prec^-$ is acyclic in $\prec^-_\cup P$[11] and for each $Q \in \prec^-_\cup P$:*

—$Q \notin^+ \Psi_Q$,
—for all $R \prec^- Q$, $Q \notin^+ \Psi_R$,
—$\Psi_Q[\bar{x}]$ is squared in $\mathfrak{D}$.

PROOF. See the electronic appendix. □

We now illustrate the need of some of the conditions of the theorem. We already saw the necessity of squaredness of windows of expertise to obtain an accurate $P^{\mathcal{C}_\mathfrak{D}}$ in Example 15.

Example 17. For the database $\mathfrak{D} = (\{\}, \{\mathcal{LCWA}(P(x), x = a \wedge P(a))\})$, we have that $P(a)^{\mathcal{O}_\mathfrak{D}} = \mathbf{f}$ but $P(a)^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$. Indeed, $P(a)$ is unknown in the initial step of the construction and remains unknown when applying $\mathcal{A}pp_\mathfrak{D}$. Here, it

[11]Recall that this set is defined as $\{P\} \cup \{Q | Q \prec^- P\}$.

is the first condition about the predicates in $\prec^{-}_{\cup} P$ of the optimality theorem that is not satisfied.

Example 18. The database $\mathfrak{D}=(\{\}, \{\mathcal{LCWA}(P(x), \neg Q(x)), \mathcal{LCWA}(Q(x), P(x))\})$ is equivalent with the theory $\{\forall x : (\neg P(x) \vee Q(x)), \forall x : (\neg Q(x) \vee \neg P(x))\}$ which is equivalent to $\{\forall x : \neg P(x)\}$. Hence, it holds for each $d \in HU$ that $P(d)^{\mathcal{O}_\mathfrak{D}} = \mathbf{f}$ and $Q(d)^{\mathcal{O}_\mathfrak{D}} = \mathbf{u}$ while $P(d)^{\mathcal{C}_\mathfrak{D}} = Q(d)^{\mathcal{C}_\mathfrak{D}} = \mathbf{u}$. Here, the second condition about the predicates in $\prec^{-}_{\cup} P$ is violated since $Q \prec^{-} P \in \prec^{-}_{\cup} P$ and $P \in^{+} \Psi_Q$.

We can summarize the preceding in the following result about the accuracy of certain answers.

THEOREM 9 (COMPLETENESS). *It holds that* $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})) = Cert_\mathfrak{D}(\mathcal{Q}(\bar{x}))$ *if* $\mathcal{Q}(\bar{x})$ *is squared in* $\mathfrak{D}$ *and* $\prec^{-}$ *is acyclic in* $\prec^{-}_{\cup} \mathcal{Q}(\bar{x})$ *and for every database predicate* $Q \prec^{-} \mathcal{Q}(\bar{x})$*:*

—$Q \notin^{+} \Psi_Q$,
—*for all* $R \prec^{-} Q$, $Q \notin^{+} \Psi_R$,
—$\Psi_Q[\bar{x}]$ *is squared in* $\mathfrak{D}$.

PROOF. This is a corollary of Theorem 6 and Theorem 8. □

Apart from checking squaredness, the costs of checking the conditions in Theorem 9 are negligible. Theorem 7 gives a condition for checking the squaredness of formulas of the form $\forall \bar{y} : (C_1 \vee \cdots \vee C_n)$ where the C_i are conjunctions, namely:

(1) each nonliteral conjunct C_{i_k} of C_i consists of predicates which are either positive free in $\mathcal{Q}(\bar{x})$ or two-valued in the optimal approximation $\mathcal{O}_\mathfrak{D}$, and
(2) the conjunctions C_i are mutually exclusive with respect to $\mathcal{O}_\mathfrak{D}$.

The main practical problem to check squaredness using this theorem is that we have to check two-valuedness of negatively bound predicates in $\mathcal{O}_\mathfrak{D}$ and mutual exclusiveness in $\mathcal{O}_\mathfrak{D}$. $\mathcal{O}_\mathfrak{D}$ is typically not available and is hard to compute. Fortunately, since $\mathcal{C}_\mathfrak{D} \leqslant_\mathrm{p} \mathcal{O}_\mathfrak{D}$, we can use $\mathcal{C}_\mathfrak{D}$ instead of $\mathcal{O}_\mathfrak{D}$. For example, all base predicates of $\mathfrak{D}$ are two-valued in $\mathcal{C}_\mathfrak{D}$. However, it can happen that a predicate is two-valued in $\mathcal{O}_\mathfrak{D}$ but three-valued in $\mathcal{C}_\mathfrak{D}$, in which case the squaredness of a query might not be discovered. An extreme case is in Example 15 where every predicate is two-valued in $\mathcal{O}_\mathfrak{D}$ and none in $\mathcal{C}_\mathfrak{D}$. In this database, the query $P(a) \vee \neg P(a)$ is squared but this cannot be discovered using $\mathcal{C}_\mathfrak{D}$.

Some types of queries that can be discovered to be squared are the following:

—conjunctions of literals, possibly prefixed with universal quantifiers;
—positive queries;
—formulas in which all predicates are two-valued in $\mathcal{C}_\mathfrak{D}$;
—queries whose form is similar to *decision trees* or *binary decision diagrams*, in which all test formulas in the nonleaf nodes contain only two-valued predicates in $\mathcal{C}_\mathfrak{D}$ and the leaves consist of conjunctions of literals. An example of such a query would be

$$\textbf{if } \exists x : P(x, y, z) \quad \textbf{then if } \neg Q(y, z) \textbf{ then } R(y, z) \textbf{ else } \neg R(y, z) \textbf{ else } R(z, y)$$

where the predicates P and Q are two-valued in $\mathcal{O}_\Gamma$. It denotes the following formula.

$$\begin{array}{c} \exists x : P(x, y, z)) \wedge \neg Q(y, z) \wedge R(y, z) \vee \\ \exists x : P(x, y, z)) \wedge Q(y, z) \wedge \neg R(y, z) \vee \\ \neg \exists x : P(x, y, z) \wedge R(z, y) \end{array}$$

Example 19. Consider again any database instance $\mathfrak{D}$ with the LCWA's of Example 11.

$$\mathcal{L} = \left\{ \begin{array}{l} \mathcal{LCWA}(Loc(n, l), (l = Bx \wedge \exists n, m : CarO(n, m, id))), \\ \mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx)) \end{array} \right\}$$

The windows of expertise are positive formulas, hence the contained predicates are positive free. It follows that both are squared in $\mathfrak{D}$. Also, $\prec^-$ is the empty graph, hence it is acyclic. It follows that $\mathcal{C}_\mathfrak{D}$ is optimal.

The queries $CarO(n, m, id)$ and $\neg CarO(n, m, id)$ are literals, hence they are squared. It follows that the approximate methods for computing their certain and possible answers are optimal.

The previous class of locally closed databases and queries is interesting, but it can be substantially increased when $\mathfrak{D}$ at least partially conveys CWI on its own windows of expertise.

THEOREM 10 (COMPLETENESS UNDER CWI). *It holds that* $Cert_{\mathcal{C}_\mathfrak{D}}(\mathcal{Q}(\bar{x})) = Cert_\mathfrak{D}(\mathcal{Q}(\bar{x}))$*, if: (i)* $\prec$ *is acyclic in* $\mathcal{N}eg\mathcal{B}(\mathcal{Q}(\bar{x}))$ *and* $\mathfrak{D}$ *conveys CWI on* Ψ_P *for each* $P \in \mathcal{N}eg\mathcal{B}(\mathcal{Q}(\bar{x}))$ *and (ii) only base predicates of* $\mathfrak{D}$ *occur both positively and negatively in* $\mathcal{Q}(\bar{x})$ *and in* Ψ_P*, for each* $P \prec^- \mathcal{Q}(\bar{x})$*.*

PROOF. See the electronic appendix. □

An interesting aspect of this theorem is that the subclass of queries in which only base predicates have both positive and negative occurrences is closed under negation, so that the theorem guarantees that also possible answers to such queries can be optimally answered.

Example 20. A database with the following LCWA's:

$$\mathcal{L}_1 = \left\{ \begin{array}{l} \mathcal{LCWA}(Loc(n, l), \mathbf{t}), \\ \mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx)) \end{array} \right\}$$

satisfies the conditions of Theorem 8 since $\prec^- = \emptyset$ and positive formulas are squared. It follows that $\mathcal{C}_\mathfrak{D} = \mathcal{O}_\mathfrak{D}$. In addition, this database conveys CWI on its windows of expertise. Such a database satisfies the conditions of Theorem 10 for the query $\mathcal{Q} := \neg Loc(LD, Bx) \vee \neg CarO(LD, V, V40)$ and hence, this query can be accurately answered using the approximate methods. On the other hand, a database $\mathfrak{D}$ with the LCWA's of Example 11:

$$\mathcal{L}_2 = \left\{ \begin{array}{l} \mathcal{LCWA}(Loc(n, l), (l = Bx \wedge \exists n, m : CarO(n, m, id))), \\ \mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx)) \end{array} \right\}$$

also satisfies the conditions of Theorem 8 since $\prec^{-} = \emptyset$ and positive formulas are squared. Hence, again $\mathcal{C}_{\mathfrak{D}} = \mathcal{O}_{\mathfrak{D}}$. Note, however, that $\mathfrak{D}$ does not convey CWI on its windows of expertise. Moreover, for the preceding query, $\prec$ is cyclic in $\mathcal{N}eg\mathcal{B}(\mathcal{Q}) = \{Loc/2, CarO/3\}$ and $\mathfrak{D}$ has no CWI on the windows of expertise of these predicates hence the conditions of Theorem 10 are not satisfied and the query cannot be accurately answered.

The conditions of Theorem 10 on the syntactic form of queries and windows of expertise are weaker than those of Theorem 9, at the expense of a stronger acyclicity condition on $\prec$ and the requirement of CWI on certain windows of expertise. A problem in applying this theorem is that determining whether a database conveys CWI on a window of expertise is a hard problem (Proposition 14). A way out would be to limit the syntax of windows of expertise to the formulas that can be constructed using Proposition 10 and Proposition 11.

To summarize, the results in this section allow us to prove the optimality of the approximate certain answers in the context of queries $\mathcal{Q}(\bar{x})$ and (a subset of the) windows of expertise in the form of Theorem 7, for example, positive formulas, conjunctions of literals, decision-tree-like formulas with base predicates in the tests, etc. Even larger classes of queries can be precisely answered with respect to locally closed databases with CWI on its windows of expertise. This is a rather rich class of queries and databases, allowing many forms of cycles in the LCWA-dependency graph.

Regarding the precision of approximate query answering beyond the conditions of the last completeness theorems, we presented examples where there is a drastic loss of precision, but these were artificial examples which were carefully "engineered" for example, by including tautologies of two-valued logic, unnecessary cycles, or by using the extended relational formula of one LCWA as a window of expertise of some other object predicate. In practice it seems unlikely that LCWA's will be of this form. This may suggest that the approximate reasoning methods are usually quite precise in practice, but currently we do not have sufficient experience to evaluate this hypothesis.

Bibliographic note. Some preliminary results on the accuracy of the returned answers first appeared in Cortés-Calabuig et al. [2008] (mainly those of Section 5.1 and Theorem 6). These results have been refined, extended, and rearranged in this section, and the proofs rewritten to facilitate the reading. The negative complexity result of Proposition 31 and the entire analysis of completeness under CWI, which culminates in Theorem 10, are new.

6. RELATED WORK

Interestingly, the principle of localization of complete knowledge is naturally applied in other areas of computer science as well. This is the case, for instance, in the context of image processing, where Intille and Bobick [1995] realize that conventional methods of tracking objects in complex and dynamic scenes are unlikely to perform well, and propose to used "closed world" analysis in a space-time region of an image where contextual information like the number and type of objects within the region is assumed to be known. The basic idea is therefore

similar to ours: selecting a small region that can be analyzed locally using image processing algorithms for tracking with high degree of confidence.

In this section we concentrate on related works in the context of relational database systems. The research about representing and reasoning with partially complete information in such systems is wide and diverse, and the relations between the different approaches are not always clear. We do not intend to give here an exhaustive description of all related work, but focus on previous research that we consider to be close in spirit to our own goals, and explain the relations to our approach.

6.1 Local Forms of CWA in Databases

Several different local forms of CWA have been proposed in the literature. We already mentioned Levy's [1996] work. These approaches are formalized using various styles of syntax and semantics and with different goals in mind, making it quite difficult to compare them. Therefore, we explain these approaches in the terminology of our article and focus on their relation with our work.

One of the first studies of local forms of CWA was by Motro [1989], who investigates partial forms of validity (soundness) and completeness in relational databases. Both *validity constraints* and *completeness constraints* are represented there as queries (views). Intuitively, a completeness constraint is a query $\Psi[\bar{x}]$ such that its answers in the real world are also the answers computed from the database D. A validity constraint expresses the inverse. A corresponding algorithm takes as input a given set $\mathcal{V}$ of validity constraints, a set $\mathcal{C}$ of completeness constraints, and an arbitrary query $\mathcal{Q}(\bar{x})$. The algorithm computes both a lower approximation query $\mathcal{Q}_l[\bar{x}]$ and an upper approximation query $\mathcal{Q}_u[\bar{x}]$, such that any answer for $\mathcal{Q}(\bar{x})$ computed from D that satisfies $\mathcal{Q}_l[\bar{x}]$ is a correct answer (i.e, it is an answer to the query in the real world), and any correct answer for $\mathcal{Q}(\bar{x})$ in the real world that satisfies $\mathcal{Q}_u[\bar{x}]$ is computable from D. The algorithm requires that the query and all the constraints would be existentially quantified conjunctions of atoms.

Let us now investigate the semantics of this formalism. For the sake of simplicity, we focus on a database where the set of validity constraints $\mathcal{V}$ contains $P(\bar{x})$ for every $P \in \Sigma$, that is, the database is sound. We allow arbitrary first-order formulas as completeness constraints and denote that $\Psi[\bar{x}]$ is a completeness constraint by $CC(\Psi[\bar{x}])$. A completeness constraint $\Psi[\bar{x}] \in \mathcal{C}$ means that its answers in *the real world* can be computed from the database. The information conveyed about the real world by this constraint is formalized by the following first-order formula.

$$\forall \bar{x} : \left(\Psi[\bar{x}] \supset \bigvee_{\{\bar{d} | D \models \Psi[\bar{d}]\}} \bar{x} = \bar{d} \right)$$

This axiom expresses that if $\Psi[\bar{d}]$ is true in the real world then $\bar{d}$ is one of the answers of the query $\Psi[\bar{x}]$ in the database instance D. Thus, the extended relational theory of a database $(D, \mathcal{C})$ (assuming its soundness) is the theory consisting of DCA and UNA, the database atoms, and the preceding formulas. Note that the problem of whether such a database conveys CWI on a query

reduces to the problem of whether the query can be answered using the views in $\mathcal{C}$ (see Levy et al. [1995]).

It is easy to see that completeness constraints are semantically different than our LCWA's in the sense that there exist LCWA's that cannot be expressed as completeness constraints and vice versa.

Example 21. Consider the assumption $\mathcal{LCWA}(P, Q)$. An equivalent (set of) completeness constraint(s) Ψ should have the property that for every database instance D, $\mathfrak{D} = (D, \{\mathcal{LCWA}(P, Q)\})$ and $\mathfrak{D}' = (D, \{\Psi\})$ are equivalent. Take $D = \{\}$, in which case $\mathcal{M}(\mathfrak{D}) \equiv \neg Q \vee \neg P$. There is only one completeness constraint Ψ (modulo equivalence) that makes $\mathfrak{D}'$ equivalent to $\mathfrak{D}$, and this is $\Psi \equiv Q \wedge P$. Next, if we take $D = \{P\}$, then $\mathcal{M}(\mathfrak{D}) \equiv P$, while $\mathcal{M}(\mathfrak{D}')$ contains the axiom $\neg P \vee \neg Q$ which entails $\neg Q$. Hence, this LCWA and completeness constraint are not equivalent. Conversely, it is easy to show by a similar technique that the completeness constraint $P \wedge Q$ cannot be expressed as one or more LCWA's.

A limited class of LCWA's and completeness constraints that are semantically equivalent are

$$\mathcal{LCWA}(P(\bar{x}), \varphi[\bar{x}]) \text{ and } CC(P(\bar{x}) \wedge \varphi[\bar{x}]),$$

where $\varphi[\bar{x}]$ contains only base predicates. Take as an example the following expressions.

$$\mathcal{LCWA}(Dept(x, y), y = CS) \text{ and } CC(Dept(x, y) \wedge y = CS)$$

The latter is equivalent to $CC(Dept(x, CS))$. The equivalence is shown as follows.

$$\begin{aligned}\mathcal{M}_D(CC(P(\bar{x}) \wedge \varphi[\bar{x}])) &\equiv \forall \bar{x} : (P(\bar{x}) \wedge \varphi[\bar{x}] \supset \textstyle\bigvee_{\{\bar{d} | D \models P(\bar{a}) \wedge \varphi[\bar{d}]\}} \bar{x} = \bar{d}) \\ &\equiv \textstyle\bigwedge_{\{\bar{d} | D \models \varphi[\bar{d}]\}} P(\bar{d}) \supset (P(\bar{a}) \in D) \\ &\equiv \mathcal{M}_D(\mathcal{LCWA}(P(\bar{x}), \varphi[\bar{x}]))\end{aligned}$$

Here, the second equivalence follows from the CWI on $\varphi[\bar{x}]$. Thus, both databases are equivalent.

Let us evaluate the merits of both types of local forms of CWA. It appears from the previous discussion that LCWA's cannot be expressed as completeness constraints if they have a window of expertise without CWI. Example 11 illustrates this. There, we have

$$\begin{aligned}&\mathcal{LCWA}(CarO(n, m, id), Loc(n, Bx)), \\ &\mathcal{LCWA}(Loc(n, l), (l = Bx \wedge \exists n, m : CarO(n, m, id))).\end{aligned}$$

No CWI is available for either of the windows of expertise of these local closed world assumptions. Hence, LCWA's like these cannot be expressed by completeness constraints. For an example of a completeness constraint that cannot be expressed as a LCWA, consider a database with a relation $Tel(p, d, nr)$, representing that p works in department d and has telephone number nr, and the completeness constraint $CC(\exists nr : Tel(p, CS, nr))$, expressing that the database knows at least one telephone number for each person of CS that has a

telephone. As in Example 21, this completeness constraint cannot be expressed as a LCWA.

The preceding discussion suggests that it might be interesting to combine the two sorts of local CWA's. In such a generalized formalism, LCWA's might have the form

$$\mathcal{LCWA}(\varphi[\bar{x}], \Psi[\bar{x}]),$$

where φ and Ψ are formulas, and the extended relational formula is given by

$$\forall \bar{x} : \left(\Psi[\bar{x}] \supset (\varphi[\bar{x}] \supset \bigvee_{\{\bar{d} | D \models \varphi[\bar{d}]\}} \bar{x} = \bar{d}) \right).$$

This obviously generalizes both LCWA's and completeness constraints. An example that could be dealt with in this generalized formalism and in neither of the original ones is a variant of the telephone example. Assume that we have split up the predicate $Tel(p, d, nr)$ in two predicates $Dept(p, d)$ and $Tel(p, nr)$. In this case, we can express that at least one telephone number is known for a person of CS by the following generalized LCWA.

$$\mathcal{LCWA}(\exists nr : Tel(p, nr), Dept(p, CS))$$

The sort of local CWA as proposed by Motro [1989] was later used in other contexts as well. In Etzioni et al. [1997], the same notion of local CWA in the context of planning problems is used. The information contained in the planner consists of a consistent and correct set L of (positive or negative) literals, and a collection of completeness constraints. A polynomial approximate (sound but incomplete) algorithm is introduced for computing certain answers for queries in the form of conjunctions of atoms. It is in that paper where the term Closed World Information (CWI) is introduced. Note that Etzioni et al. [1997] consider a set L of literals rather than a database instance D as in Motro [1989]. Yet, L can be translated to a database instance D consisting of the positive literals, together with extending $\mathcal{C}$ by completeness constraints of the form $\{P(\bar{a}) \mid \neg P(\bar{a}) \in L\}$.

Another closely related approach to specify local completeness is presented in Gottlob and Zicari [1988]. The basic idea is the inverse of ours, namely to open portions of a closed database relation by means of a particular symbol, a so-called *open null* value which defines attributes, parts of a relation, or entire relations as *open*. In the context of our Examples 1 and 3, Gottlob and Zicari [1988] would add a tuple $Tel(\text{LD}, open)$ to express that Lien Desmet might have other numbers than those in the table. Similarly, it would have tuples $Tel(\text{DF}, open)$ and $Tel(\text{BD}, open)$. The same information can easily be conveyed by a LCWA, namely

$$\mathcal{L} = \{\mathcal{LCWA}(Tel(e, t), e \neq LD \wedge e \neq DF \wedge e \neq BD)\}.$$

On the other hand, it is not always straightforward to represent locally closed databases by open null values. Indeed, as shown, for example, in Example 8, a window of expertise may use different relations to express closure on a predicate object, while open null values can only express incompleteness over particular tuples of a single table.

Doherty et al. [2000] also study a generalized form of completeness constraints in planning systems. They extend the work of Etzioni et al. [1997] by considering arbitrary first-order queries in $\mathcal{C}$. They present a circumscriptive semantics [McCarthy 1990; Lifschitz 1994] for knowledge bases consisting of a set L of literals and a set $\mathcal{C}$ of completeness constraints. The meaning of such a knowledge base is given by DCA, UNA, the set L, and a circumscription axiom minimizing the answers to the elements in $\mathcal{C}$. It is shown that for constraints that are existentially quantified conjunctions of atoms, the circumscriptive semantics coincides with the semantics of Etzioni et al. [1997], explained before. Other results obtained in Doherty et al. [2000] are that for certain types of completeness constraints, in particular for semi-Horn clauses, the circumscriptive axiom can be translated in a fixpoint query, allowing to solve queries in polynomial time. This approach is then further extended to rough databases with integrity constraints.

Note 8. The earlier approach of defining the extended relational formula of a completeness constraint $CC(\Psi[\bar{x}])$ by a formula of the form

$$\forall \bar{x} : \left(\Psi[\bar{x}] \supset \bigvee_{\{\bar{d} | D \models \Psi[\bar{d}]\}} \bar{x} = \bar{d} \right)$$

generalizes to the case of full first-order formulas Ψ as well. It is easy to see that when Ψ is a positive formula, the first-order approach and the circumscriptive approach coincide. Indeed, the minimal answer to a positive query is obtained in the smallest model, and this is given by D (or by the structure given by the positive literals in L). However, for nonpositive queries, the two approaches strongly disagree. Consider, for example, the completeness constraint $CC(\neg P(x))$. The first-order semantics for this constraint is

$$\forall \bar{x} : \left(\neg P(\bar{x}) \supset \bigvee_{\{\bar{d} | D \models \neg P(\bar{d})\}} \bar{x} = \bar{d} \right).$$

The tuples $\bar{x}$ that falsify $\bigvee_{\{\bar{d} | D \models \neg P(\bar{d})\}} \bar{x} = \bar{d}$ are exactly those for which $P(\bar{x}) \in D$. It follows that, given the DCA, the previous formula is equivalent to

$$\forall \bar{x} : (P(\bar{x}) \in D \supset P(\bar{x})).$$

Surprisingly, this constraint restates that D is sound in P and is entailed by the soundness assumption. In the circumscription approach for this example, however, $\neg P(x)$ is minimized, which implies that P is true for all domain elements. This shows that in the approach of Doherty et al. [2000], circumscriptive semantics for general completeness constraints does *not* correctly implement the original intuition of Motro [1989] and Etzioni et al. [1997], which is that all answers of a completeness constraint in the real world can be computed from the database. While it does not mean that this approach has no virtues, this certainly pinpoints an unsolved semantical problem. In the electronic appendix, we show how local closed world assumptions can be represented by circumscriptive formulas. By Section 2.5, then, this alternative representation of local

closure in database systems by means of second-order formulas *is* faithful to Levy's [1996] original intuition on this matter.

Our approach is also tightly linked with Levesque's work on reasoning on incomplete knowledge-bases [Levesque 1982], where he introduces a framework for reasoning with first-order knowledge-bases (theory) KB and modal logic queries in the logic $KFOPCE$. In this framework, the query for certain answers for $\mathcal{Q}(\bar{x})$ is stated as $\mathbf{K}\mathcal{Q}(\bar{x})$ and for possible answers as $\neg\mathbf{K}\neg\mathcal{Q}(\bar{x})$. Such a modal formula is evaluated using S5 semantics with respect to a possible world structure $W(KB)$, consisting of all models of KB.

Levesque also studies the use of modal logic to assert completeness or incompleteness on the first-order knowledge base. For instance, the statement

$$\forall x : \forall y : (Works(x, CS) \supset (Tel(x, y) \supset \mathbf{K}Tel(x, y)))$$

is the way Levesque would express that the telephone numbers of people from the Computer Science department are known. He goes on by warning that one should not just extend KB with such formulas. The result of this would be a modal knowledge-base with some undesirable properties. The solution he proposes is to *reduce* such a modal formula to a first-order formula in the context of KB. In particular, given a modal formula $\alpha[\bar{x}]$, there is a first-order formula $\beta[\bar{x}]$ such that

$$KB \models \forall \bar{x} : (\alpha[\bar{x}] \leftrightarrow \beta[\bar{x}]),$$

where $\models$ means truth in $W(KB)$. Levesque also presents an algorithm to compute β from KB and α. Interestingly, if we assume that KB consists of $\mathsf{DCA} \wedge \mathsf{UNA} \wedge \bigwedge_{P(\bar{a})\in D} P(\bar{a})$, then by applying this reduction algorithm on a formula of the form

$$\forall x : \forall y : (Works(x, CS) \supset (Tel(x, y) \supset \mathbf{K}Tel(x, y)))$$

we obtain (after simplification using DCA) the formula

$$\forall x : \forall y : (Works(x, CS) \supset (Tel(x, y) \supset (Tel(x, y) \in D))),$$

which of course is the logical meaning of $\mathcal{LCWA}(Tel(x, y), Works(x, CS))$.

This observation is extendable to arbitrary LCWA's. Thus, our approach coincides with (a fragment of) Levesque's approach.

In the context of integration of distributed databases, local forms of closure and soundness axioms may be applied as well. Grahne and Mendelzon [1999] study integrated databases using *Local-as-View* (LAV). This approach involves a set of source databases and a set of conjunctive queries defining source predicates in terms of global database relations. Source predicates can be declared either as open (sound), closed (complete), or clopen (sound and complete). Grahne and Mendelzon [1999] present a coNP-algorithm for determining consistency of such databases and for computing certain and possible answers of global queries. Optimization of query answering in LAV databases using more general completeness constraints on sources is studied in Abiteboul and Duschka [1998].

Declaring a source predicate as complete (or sound), following Grahne and Mendelzon's approach, looks quite crude compared to our local closed world

assumptions. Yet, their algorithm can be accommodated to handle some more refined forms of LCWA, and in fact, in a more general form of databases. This algorithm accepts a generalized sort of database, called a *database template*, whose extended relational theory is given by

$$\mathsf{DCA}(\mathcal{C}(\sigma)) \wedge \mathsf{UNA}(\mathcal{C}(\sigma)) \bigwedge (D_1 \vee \cdots \vee D_n) \bigwedge (C_1 \wedge \ldots C_m),$$

where D_i are existentially quantified conjunctions of atoms and C_i are constraints of the form

$$\forall \bar{x} : \left(C_i' \supset (\bigwedge_{\bar{d} \in S_i} \bar{y} = \bar{d}) \right).$$

Here again, C_i' are conjunctions of atoms with free variables $\bar{x}$, $\bar{y}$ is a subset of $\bar{x}$, and S_i are sets of tuples of domain elements of the same arity as $\bar{y}$. Thus, a database template can be viewed as a disjunction of databases with null values (the variables), augmented with a set of closure constraints.

It is easy to see that a locally closed database $(D, \mathcal{L})$ can be represented as a database template if all windows of expertise are existentially quantified conjunctions of atoms. In this case, the extended relational formula $\mathcal{M}_D(\theta)$ of a LCWA $\theta \in \mathcal{L}$ has the form of a constraint C_i. The same holds for databases with completeness constraints in Motro's sense. Thus, Grahne and Mendelzon's algorithm can be used to compute certain and possible answers for conjunctive queries in locally closed databases with conjunctive formulas as windows of expertise. Note that conjunctive queries are positive formulas, and hence, its certain answers can be accurately obtained by querying the database D. To obtain possible answers, however, observe that such databases are hierarchically closed (Section 4.5) and hence, Algorithm 1 can be used, although it does not always return optimal answers. For example, the possible answer for $P \wedge Q$ with respect to $\mathfrak{D} = (\emptyset, \{\mathcal{LCWA}(P, Q)\})$ is $\mathbf{f}$, whereas the algorithm rewrites the query as $P \vee \neg Q$, which evaluates to $\mathbf{u}$ in $\mathcal{O}_{\mathfrak{D}}$. Thus, Grahne and Mendelzon's algorithm is more accurate for possible conjunctive queries at the cost of a higher complexity.

6.2 Deductive Databases and Logic Programming

The area of deductive databases is tightly connected with logic programming. In this area, several approaches have been proposed for weakening the CWA. Next, we discuss the most relevant ones.

6.2.1 *Answer Set Programming.* In Gelfond and Lifschitz [1991], the formalism of *general logic programming* is extended by dropping the closed world assumption and allowing *classical negation* to appear in heads and bodies of rules. One of the main reasons for this extension was the need in many applications to relax the closed world assumption in order to represent incomplete knowledge.[12] Thus, while CWA in general logic programming entails the falsity of all atoms that are not proved true, in answer set programming, falsity

[12]Personal communication with M. Gelfond.

of atoms needs to be represented explicitly by rules with a classical negation literal $\neg P(\bar{t})$ in the head. This allows the authors to represent local forms of closed world assumption in a way which is similar to ours.

Consider, for instance, the program rule $\neg P(\bar{x}) \leftarrow \texttt{not}P(\bar{x}), \varphi$. Under stable-models semantics, the truth of $\neg P(\bar{x})$ is established if there is no evidence that $P(\bar{x})$ is true (i.e., $P(\bar{x})$ is not in the database) and φ is provable. The relationship with the LCWA as defined in this article is clear: φ is the window of expertise of predicate P.

The semantics of an answer set program is given by stable models, which are (consistent) sets of literals. Such a set is clearly associated with a three-valued structure, and so answer set programs correspond to the fixpoint queries of Section 4.4 that express certainly true and certainly false atoms. Indeed, in the notations of Section 4.4, $P(\bar{t})$, $\neg P(\bar{t})$, $\texttt{not}P(\bar{t})$ and $\texttt{not}\neg P(\bar{t})$ correspond, respectively, to $P^c(\bar{t})$, $P^{c\neg}(\bar{t})$, $P^{p\neg}(\bar{t})$, and $P^p(\bar{t})$.

As a way of expressing LCWA's, answer set programming has the same limited accuracy as our approximate methods. This is demonstrated in the following example.

Example 22. The locally closed database $\mathfrak{D} = (\{\}, \{\mathcal{LCWA}(P, Q)\})$ may be expressed by the program $\mathcal{P} = \{\neg P \leftarrow \texttt{not}P,\ Q\}$. As we saw before, $\mathfrak{D}$ entails that $P \wedge Q$ is false, but as in our approximate method, the falsity of $P \wedge Q$ is not inferable from the unique answer set $\{\}$ of $\mathcal{P}$.

In comparison with our approach, a weakness of using answer set programming for expressing LCWA's is that the preceding sort of approximating programs need to be expressed manually, while in our case the fixpoint queries are automatically derived from the database.

It is interesting that Baral et al. [1993] introduced a method for computing an answer set program as an approximation (*interpolation* in terms of Baral et al. [1993]) of a collection of *general logic programs*. The techniques presented in that paper are likely to provide an extension of our approximate methods for locally closed *deductive databases*, that is, locally closed databases extended with view definitions.

6.2.2 *Other Generalized Closed World Assumptions.* Minker [1982] introduced disjunctive databases as an extension of deductive databases for representing incomplete knowledge by allowing clauses to have disjunctions in their heads. As noted originally by Reiter [1978], such expressions may be inconsistent with the CWA. For instance, since the disjunctive database $\{P(a) \vee Q(a)\}$ neither entails $P(a)$ nor $Q(a)$, the CWA would entail both $\neg P(a)$ and $\neg Q(a)$, which of course is inconsistent with the content of the database. Minker addresses this phenomenon by introducing the *Generalized Closed World Assumption* (GCWA) [Minker 1982], specifying that a negated atom is considered to be true if the atom does not appear in any minimal model of the database. Note that in case a database does not contain disjunctive clauses, GCWA and CWA coincide. This shows that disjunctive databases and locally closed databases are different ways of representing incomplete knowledge, and that the GCWA is a very different principle than LCWA's.

Another extension of the CWA in the context of logic programming is the *Any World Assumption*, introduced by Loyer and Straccia [2005]. They argue that due to the incompleteness of information, any semantics of logic programming has to rely on a default assumption on the missing information. In case of the stable semantics, this default is *false*, while in case of OWA, the default is *unknown*. Loyer and Straccia [2005] view these as two extremes and develop a semantics in which the default assumption can be any prespecified value in an arbitrary truth space. While this approach aims at somewhat similar goals as ours, context, techniques, and viewpoints are completely different.

7. DISCUSSION AND FUTURE WORK

In this section we comment on some LCWA-related issues that are outside the scope of the current article and on some directions for further work. In Section 6.1 we already discussed a generalized form of the LCWA that captures Motro's completeness constraints. There is also the need for a better understanding of the kind of queries that are useful in practical applications and how close to optimal the approximate answers are for those queries. Some other issues are explored more in depth in the following.

7.1 Domain-Independent and Safe Queries

The problem of domain independence of a query [Topor and Sonenberg 1988] for certain answers in standard (finite) databases is corecursively enumerable, and the same holds for locally closed databases, both for possible and certain answers.[13]

A more relevant practical question is the safety [Topor and Sonenberg 1988] of approximate query answering, as this question is linked to computing certain or possible answers using relational algebra. Let us call a query $\mathcal{Q}(\bar{x})$ *safe for certain (respectively, possible) answers* if the fixpoint query $[\mathbf{lfp}_{Q^{\mathcal{L}},\Delta^{c}_{\mathcal{Q},\mathcal{L}}}](\bar{x})$ (respectively, if $\neg[\mathbf{lfp}_{Q^{\neg},\Delta^{c}_{\mathcal{Q},\mathcal{L}}}](\bar{x})$) is safe. As we have seen in several examples before, queries to locally closed databases are often unsafe, especially positive queries for possible answers and negative queries for certain answers.

Example 23. Following are two examples for unsafe queries to locally closed databases.

(1) As shown in Example 11, the certain answers for $CarO(n, m, id)$ are given by $CarO(n, m, id)$. Clearly, this is a safe query. On the other hand, the possible answers for the same query are given by $CarO(n, m, id) \vee \neg Loc(n, Bx)$, which is domain dependent, since n is unconstrained in the second disjunct. Also, the query $\neg CarO(n, m, id) \wedge Loc(n, Bx)$ that computes the certainly false answers (or the certain answers for $\neg CarO(n, m, id)$), is unsafe as m and id are unconstrained.

[13]To see this, compute for arbitrary large finite extensions of the domain all models with that domain, and the possible or certain answers of the query. Check if these possible or certain answers for that domain are the same as those for the original domain. If not, then there is domain dependence.

(2) More generally, take a locally closed database in which all windows of expertise are positive formulas (hence, it is hierarchically closed). Then $R^p(\mathcal{Q}(\bar{x}))$ is obtained from $\mathcal{Q}(\bar{x})$ by substituting every database predicate atom $P(\bar{t})$ in $\mathcal{Q}(\bar{x})$ by $P(\bar{t}) \vee \neg\Psi_P[\bar{t}]$. Since the second disjunct is a negative formula, the resulting query will be unsafe unless it is variable-free.

Note that the opposite situations are also possible: unsafe queries in standard relational databases are not necessarily unsafe in locally closed databases.

Example 24. Consider the query $\mathcal{Q}(x) = \neg P(x)$ on a locally closed database $\mathfrak{D} = (D, \mathcal{L})$, where $D = \{R(a)\}$ and $\mathcal{L} = \{\mathcal{LCWA}(R(x), \mathbf{t}),\ \mathcal{LCWA}(P(x), R(x))\}$. Since there is complete knowledge on R, the set of certain answers for $\mathcal{Q}(x)$ is $\{a\}$. Note that $R^c(\neg P(x)) = \neg P(x) \wedge R(x)$, which is a safe query.

We believe that domain dependence for possible query answering is inherent to all formalisations of incomplete database, as such databases do not (or hardly not) characterize unnamed objects.

Some preliminary ideas on how to assure safety in locally closed databases are as follows.

—The database could maintain a unary predicate $U/1$ representing all elements of the domain. Replacing every literal $\neg P(\bar{x})$ that contains an unsafe variable x by the conjunction $U(x) \wedge \neg P(\bar{x})$ ensures safety.[14]

—One could develop symbolic query answering methods that return queries with constrained variables to correctly represent the answer in arbitrary domains. This is the case, for instance, with the set $\{(PS, m, id) \mid (m, id) \neq (M, Q1)\}$ that represents the possible answers for the query $CarO(n, m, id)$ in Example 11 for arbitrary domains. For example, techniques from constructive negation in logic programming could be useful [Chan 1988].

7.2 Locally Closed Deductive Databases with Integrity Constraints and Views

Another challenging problem is to extend the concepts and methods presented here to deductive databases with first-order integrity constraints [Reiter 1977; Ullman 1988] and with view definitions.

In Section 4.6, we showed how approximate query answering could be extended to handle functional dependency constraints and how such constraints allowed us to infer more information. The approximate method could also be extended in a straightforward way to handle locally closed databases with view definitions. Let us assume, without loss of generality, that a view predicate P is defined by unique rule

$$P(\bar{x}) \leftarrow \Psi[\bar{x}],$$

where Ψ is a first-order formula. To answer a query $\mathcal{Q}(\bar{y})$ possibly including view predicates, we can extend the method of Section 4.4, by adding the following

[14]This is the solution that is implemented in the prototype system in Cortés-Calabuig [2008].

rules to the fixpoint definition $\Delta^c_{\mathcal{Q},\mathcal{L}}$.

$$P^c(\bar{x}) \leftarrow (\Psi[\bar{x}])^c$$
$$P^{c\neg}(\bar{x}) \leftarrow (\Psi[\bar{x}])^{c\neg}$$

This raises many interesting open questions. For instance, in locally closed databases with view definitions, should local closed world assumptions be allowed only on database predicates or also on the view predicates? Interestingly, the latter would be very similar to the sort of generalized LCWA's as discussed in Section 6.1. For instance, the LCWA that at least one telephone number is known for a person of CS could be expressed by a combination of a view definition

$$HasTelephone(p) \leftarrow \exists nr : Tel(p, nr)$$

and the LCWA

$$\mathcal{LCWA}(HasTelephone(p), Dept(p, CS)).$$

A derived question then is how to extend the approximate methods for LCWA's on view predicates.

Further work needs to be done for analysing other kinds of integrity constraints and for extending the approximation method to handle them. For this purpose, we believe that the work of Wittocx et al. [2008] might be useful. It presents a generalization of our approximation algorithm that computes approximations of predicates and formulas of general first-order theories in polynomial time. Such a method could in fact be the basis for an approximate query answering method in the generalized setting of integrated locally closed (deductive) databases.

7.3 Towards Data Integration and Exchange in Locally Closed Databases

Local closed world assumptions on databases are specially suited to model data imported from external sources, since each exchange or integration rule may come with different assumptions on the completeness of the exchanged data. The following example illustrates how the aforementioned extension of locally closed deductive databases with views can be applied to a data integration setting.

Example 25. Consider two local sources with the following two relations: *DirMovie* stores directors and the movies they have filmed, and *MovieGenre* stores movies and their genre. These relations are integrated by means of a Global-as-View (GAV) mapping (see Lenzerini [2002]) into a global predicate *DirGenre* (that states the genre of movies a director has filmed) as follows.

$$DirGenre(d, g) \leftarrow \exists m : DirMovie(d, m) \wedge MovieGenre(m, g)$$

Suppose further that the relation *DirMovie* is complete with respect to Alfred Hitchcock (AH), and that the relation *DirGenre* is complete with respect to Science Fiction (SF) movies. This is represented, respectively, by the following LCWA's.

$$\mathcal{LCWA}(DirMovie(x, y), x = AH),\ \mathcal{LCWA}(MovieGenre(y, z), z = SF).$$

Using the method of Section 4.4 (applying the query transformation on the body of the rule), we obtain negative information about *DirGenre*.

$$DirGenre^{c\neg}(d,g) \leftarrow \forall m : DirMovie^{c\neg}(d,m) \vee MovieGenre^{c\neg}(m,g)$$
$$DirMovie^{c\neg}(d,m) \leftarrow \neg DirMovie(d,m) \wedge d = AH$$
$$MovieGenre^{c\neg}(m,g) \leftarrow \neg MovieGenre(m,g) \wedge g = SF$$

The disjunction expresses that d is certainly not the director of movies of genre g if, for each movie m, either AH is not the director of m or the genre of m is not SF. This, together with the source instances, allows us to derive, for instance, that Alfred Hitchcock is not the director of science fiction movies ($DirGenre^{c\neg}(AH, SF)$). It follows that local completeness at the source level allows us to derive negative answers at the mediator level. To the best of our knowledge, this is first attempt to retrieve negative answers from integrated databases.

Data exchange refers to finding an instance of a target schema given an instance of a source schema and a specification of a logical mapping between the schemas (see Libkin [2006]). This problem is thus closely related to data integration and incompleteness. Semantics for the mappings adopting an open or closed world assumption were proposed by Fagin et al. [2005] and by Libkin [2006], respectively. Libkin and Sirangelo [2008] prefer a combined approach with annotated values in the same spirit of the open null values proposed in Gottlob and Zicari [1988] and discussed in Section 6.1. As locally closed databases are closely related to the open null values in Gottlob and Zicari [1988], a refinement of the methods in Libkin and Sirangelo [2008] based on schema mappings of locally closed databases as exemplified earlier seems a promising application of local completeness.

7.4 Extending the Query Language with Modal Operators

It is quite natural to have queries that combine certain answers to one subquery with possible answers to another subquery.

Example 26. A company has a (partially complete) database about manufacturers and the products they produce. In this database, the relation *Type* classifies types of companies, and *Produces* specifies what kind of product a manufacturer produces (e.g., office articles). Both tables are partially complete. Suppose the company is interested in manufacturers of copy machines, well aware of the fact that such manufacturers belong to the category of office articles, and that the *Produces* relation is quite incomplete. A reasonable query would be to ask for companies that are certainly producers of office articles and possibly produce copy machines.

In order to handle such queries, one needs "modal" operators **P** for encapsulating possible answers to queries and **C** for encapsulating certain answers to queries. This allows one to express queries like

$$\mathbf{C}(Type(x, Office\_Articles)) \wedge \mathbf{P}(Produces(x, Copy\_Machines)).$$

A straightforward way to handle such a query is to rewrite it into

$$Type^c(x, Office\_Articles) \wedge Produces^p(x, Copy\_Machines),$$

where Q^c is derived from Q as in Definition 16 and Q^p from Q as in Definition 17. Another approach related to Levesque's modal logic *KFOPCE* [Levesque 1982; Reiter 1992] (see also our discussion in Section 6.1) is explored in Cortés-Calabuig [2008]. This topic requires further analysis.

7.5 LCWA for Semistructured Data

The interest in the semantic Web is continuously growing. A direct application of our methods in this context could be the specification of local completeness of different Web sources. For this, the notion of local completeness should be adapted to the data models used for the semistructured data present on the semantic Web [Decker et al. 2000] as in the line proposed by Abiteboul et al. [2006].

8. SUMMARY AND CONCLUSION

The need to weaken the CWA and the ability to efficiently reason with partially complete information has long been recognized as an issue of central importance in relational database systems. The current article wrapped up and considerably extended a series of publications [Cortés-Calabuig et al. 2005, 2006, 2007, 2008] in which a logical and computational framework was built for locally closed databases. The notion of local closed world assumption in the framework is the one of Levy [1996]. We generalized this notion to general first-order windows of expertise, and clarified the relation with Closed World Information known from Motro [1989], Levy [1996], Etzioni et al. [1997], and Doherty et al. [2000] in the context of knowledge-base agents and relational databases systems.

We compared our method to those and other existing approaches. We identified important cases in which LCWA induces CWI, generalized to first-order logic, explored the computational complexity of reasoning with locally closed databases, developed polynomial approximate methods to estimate query answers, and presented mappings of certain and possible queries to standard first-order queries and to fixpoint queries that allow us to compute the approximate answers using ordinary database methods. We also investigated the accuracy of the approximate methods by studying syntactical and semantical conditions that guarantee optimality of the approximate algorithms for producing certain query answers. Our results show that for quite broad classes of locally closed databases and queries, the approximate methods are optimal. The precision of these methods beyond these classes remains to be evaluated, but the variety and extent of these results suggest that in practice even the approximate methods may attain a high degree of accuracy.

Finally, we view the current work as a first step towards the more ambitious goal of providing a unifying framework for efficient approximative query answering in extended settings such as incomplete deductive databases in the presence of arbitrary integrity constraints and in settings involving distributed

data such as data integration and exchange and beyond, in the World Wide Web. Specially in the distributed settings, incompleteness is an unavoidable and prevalent phenomenon that has to be properly managed.

ELECTRONIC APPENDIX

An electronic appendix to this article is available in the ACM Digital Library.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for carefully reading this article and for many helpful comments.

REFERENCES

ABITEBOUL, S. AND DUSCHKA, O. 1998. Complexity of answering queries using materialized views. In *Proceedings of the 17th SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, 254–263.

ABITEBOUL, S. AND GRAHNE, G. 1985. Update semantics for incomplete databases. In *Proceedings of the 11th International Conference on Very Large Data Bases (VLDB)*. 1–12.

ABITEBOUL, S., HULL, R., AND VIANU, V. 1995. *Foundations of Databases*. Addison-Wesley.

ABITEBOUL, S., SEGOUFIN, L., AND VIANU, V. 2006. Representing and querying xml with incomplete information. *ACM Trans. Datab. Syst. 31*, 1, 208–254.

BANCILHON, F., MAIER, D., SAGIV, Y., AND ULLMAN, J. 1986. Magic sets and other strange ways to implement logic programs. In *Proceedings of the 5th SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS)*. ACM, 1–15.

BARAL, C., GELFOND, M., AND KOSHELEVA, O. 1993. Approximating general logic programs. In *Proceedings of the of the International Logic Programming Symposium*. 181–198.

CHAN, D. 1988. Constructive negation based on the completed database. In *Proceedings of the of the 5th International Conference on Logic Programming*, R. Kowalski and K. Bowen, Eds. MIT Press.

CHAUDHURI, S. 1993. Finding nonrecursive envelopes for datalog predicates. In *Proceedings of the 12th SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, 135–146.

CHAUDHURI, S. AND KOLAITIS, P. 1994. Can datalog be approximated? In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, 86–96.

CORTÉS-CALABUIG, A. 2008. Towards a logical reconstruction of a theory for locally complete databases. Ph.D. thesis, Department of Computer Science, K.U.Leuven.

CORTÉS-CALABUIG, A., DENECKER, M., ARIELI, O., AND BRUYNOOGHE, M. 2006. Representation of partial knowledge and query answering in locally complete databases. In *Proceedings of the 13th International Conference Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*. Lecture Notes in Computer Science, vol. 4246. Springer, 407–421.

CORTÉS-CALABUIG, A., DENECKER, M., ARIELI, O., AND BRUYNOOGHE, M. 2007. Approximate query answering in locally closed databases. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI)*. AAAI Press, 397–402.

CORTÉS-CALABUIG, A., DENECKER, M., ARIELI, O., AND BRUYNOOGHE, M. 2008. Accuracy and efficiency of fixpoint methods for approximate query answering in locally complete databases. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press, 81–91.

CORTÉS-CALABUIG, A., DENECKER, M., ARIELI, O., NUFFELEN, B. V., AND BRUYNOOGHE, M. 2005. On the local closed-world assumption of data-sources. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*. Lecture Notes in Computer Science, vol. 3662. Springer, 145–157.

DECKER, S., MELNIK, S., VAN HARMELEN, F., FENSEL, D., KLEIN, M., BROEKSTRA, J., ERDMANN, M., AND HORROCKS, I. 2000. The semantic web: The roles of XML and RDF. *Internet Comput. 4*, 5, 63–73.

DENECKER, M. AND TERNOVSKA, E. 2004. Inductive situation calculus. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. D. Dubois and C. Welty, Eds. 545–553.

DOHERTY, P., LUKASZEWICZ, W., AND SZALAS, A. 2000. Efficient reasoning using the local closed-world assumption. In *Proceedings of the of the 9th AIMSA International Conference on Artificial Intelligence: Methodology, Systems, and Applications*. Lecture Notes in Computer Science, vol. 2407, Springer, 49–58.

ETZIONI, O., GOLDEN, K., AND WELD, D. 1997. Sound and efficient closed-world reasoning for planning. *Artif. Intell. 89*, 1-2, 113–148.

FAGIN, R., KOLAITIS, P. G., AND POPA, L. 2005. Data exchange: Getting to the core. *ACM Trans. Datab. Syst. 30*, 1, 174–210.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Gener. Comput. 9*, 365–387.

GOTTLOB, G. AND ZICARI, R. 1988. Closed world databases opened through null values. In *Proceedings of the 14th International Conference on Very Large Data Bases (VLDB)*. 50–61.

GRAHNE, G. 1984. Dependency satisfaction in databases with incomplete information. In *Proceedings of the International Confeference on Very Large Data Bases (VLDB)*. 37–45.

GRAHNE, G. 1989. Horn tables—An efficient tool for handling incomplete information in databases. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. 75–82.

GRAHNE, G. 2002. Information integration and incomplete information. *IEEE Data Engin. Bull. 25*, 3, 46–52.

GRAHNE, G. AND MENDELZON, A. 1999. Tableau techniques for querying information sources through global schemas. In *Proceedings of the International Conference on Database Theory (ICDT)*. 332–347.

GRAHNE, G. AND THOMO, A. 2001. Approximate reasoning in semistructured data. In *Proceedings of the 8th International Workshop on Knowledge Representation meets Databases (KRDB)*.

IMIELINSKI, T. AND JR., W. L. 1981. On representing incomplete information in a relational data base. In *Proceedings of the International Confeference on Very Large Data Bases (VLDB)*. 388–397.

INTILLE, S. AND BOBICK, A. 1995. Closed-World tracking. In *Proceedings of the 5th International Conference on Computer Vision (ICCV)*. IEEE, 672–678.

KLEENE, S. 1952. *Introduction to Metamathematics*. Bibliotheca Mathematica, vol. 1. Van Nostrand & Wolters-Noordhoff/North-Holland, Princeton, NJ.

LENZERINI, M. 2002. Data integration: A theoretical perspective. In *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, 233–246.

LEVESQUE, H. 1982. The logic of incomplete knowledge bases. In *On Conceptual Modelling (Intervale)*. 165–189.

LEVY, A. 1996. Obtaining complete answers from incomplete databases. In *Proceedings of the 22th International Conference on Very Large Data Bases (VLDB)*. 402–412.

LEVY, A., MENDELZON, A., SAGIV, Y., AND SRIVASTAVA, D. 1995. Answering queries using views. In *Proceedings of the 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, 95–104.

LIBKIN, L. 1995. *Elements of Finite Model Theory*. Springer.

LIBKIN, L. 1998. Models of approximation in databases. *Theor. Comput. Sci. 190*, 2, 167–210.

LIBKIN, L. 2006. Data exchange and incomplete information. In *Proceedings of the 25th SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. ACM, 60–69.

LIBKIN, L. AND SIRANGELO, C. 2008. Data exchange and schema mappings in open and closed worlds. In *Proceedings of the 27th SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*. 139–148.

LIFSCHITZ, V. 1994. Circumscription. In *Handbook of Logic in AI and Logic Programming*, Vol. 3, Oxford University Press, 298–352.

LOYER, Y. AND STRACCIA, U. 2005. Any-World assumptions in logic programming. *Theore. Comput. Sci. 342*, 2-3, 351–381.

MCCARTHY, J. 1980. Circumscription—A form of nonmonotonic reasoning. *Artif. Intell. 13*, 27–39.

MCCARTHY, J. 1990. Applications of circumscription to formalizing common sense knowledge. In *Formalizing Common Sense: Papers by John McCarthy*, V. Lifschitz, Ed. Ablex Publishing, 198–225.

MINKER, J. 1982. On indefinite databases and the closed world assumption. In Lecture Notes in Computer Science, vol. 138. Springer-Verlag, 292–308.

MOTRO, A. 1989. Integrity = validity + completeness. *ACM Trans. Datab. Syst. 14*, 4, 480–502.

PAN, J. AND THOMAS, E. 2007. Approximating owl-dl ontologies. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*. 1434–1439.

REITER, R. 1977. Deductive question-answering on relational data bases. In *Logic and Data Bases*, 149–177.

REITER, R. 1978. On closed world data bases. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Plenum Press, New York, 55–76.

REITER, R. 1982. Towards a logical reconstruction of relational database theory. In *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages (Intervale)*, 191–233.

REITER, R. 1986. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *J. ACM 33*, 2, 349–370.

REITER, R. 1992. What should a database know? *J. Logic Program. 14*, 1–2, 127–153.

STUCKENSCHMIDT, H. AND VAN HARMELEN, F. 2002. Approximating terminological queries. In *Proceedings of the 5th International Conference Flexible Query Answering Systems*. 329–343.

SWIFT, T. 1999. Tabling for non-monotonic programming. *Ann. Math. Artif. Intell. 25*, 3-4, 201–240.

TOPOR, R. W. AND SONENBERG, L. 1988. On domain independent databases. In *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, 217–240.

TRAKHTENBROT, B. 1963. Impossibility of an algorithm for the decision problem in finite classes. *Amer. Math. Soc. Trans. 3*, 2, 1–5.

ULLMAN, J. D. 1988. *Principles of Database and Knowledge-Base Systems*, Vol. I. Computer Science Press.

VAN FRAASSEN, B. 1966. Singular terms, truth-value gaps, and free logic. *J. Philos. 63*, 17, 481–495.

VAN GELDER, A. 1993. The alternating fixpoint of logic programs with negation. *J. Comput. Syst. Sci. 47*, 1, 185–221.

VARDI, M. 1982. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing*. 137–145.

WITTOCX, J., MARIËN, M., AND DENECKER, M. 2008. Approximate reasoning in first-order logic theories. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*. AAAI Press.

Received March 2009; revised November 2009; accepted April 2010