Approximate Query Answering in Locally Closed Databases

Álvaro Cortés-Calabuig[†], **Marc Denecker**[†], **Ofer Arieli**[‡] and **Maurice Bruynooghe**[†]

[†] Department of Computer Science, Katholieke Universiteit Leuven, Belgium.
{alvaro.cortes,marc.denecker,maurice.bruynooghe}@cs.kuleuven.be

[‡] Department of Computer Science, The Academic College of Tel-Aviv, Israel.
oarieli@mta.ac.il

Abstract

The *Closed-World Assumption* (CWA) on databases expresses that an atom not in the database is false. A more appropriate assumption for databases that are sound but partially incomplete, is the *Local Closed-World Assumption* (LCWA), which is a local form of the CWA, expressing that the database is complete in a certain area, called the 'window of expertise'. Databases consisting of a standard database instance augmented with a collection of LCWA's are called *locally closed databases*. In this paper, we investigate the complexity of certain and possible query answering in such databases. As it turns out that these problems are intractable, we develop efficient approximate methods to underestimate certain answers and overestimate possible answers. We prove that under certain conditions, our methods produce complete answers.

Introduction

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

Databases, however, are not always complete. There are many reasons for this fact, including ignorance about the domain, lack of proper maintenance, incomplete migration, accidental deletion of tuples, the intrinsic nature of database mediator-based systems (Lenzerini 2002), and so forth. Unless properly handled, partial information in database systems might lead to erroneous conclusions.

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 (see Figure 1). Assume that in this case the database is complete with respect to all CS department members, but it is not complete regarding external collaborators. Thus, appropriate answers for $\mathsf{Telephone}(\text{Bart Delvaux}, 3962836)$ and $\mathsf{Telephone}(\text{Leen Desmet}, 3212445)$ are "no" and "unknown", respectively. If completeness of the database is taken for granted, then the answer for these queries is "no". For the query $\exists x \mathsf{Telephone}(\text{David Finner}, x)$, under the CWA the answer is "no", but as the database is complete only w.r.t. the CS department, one cannot exclude the possibility that David Finner has a phone number, so the correct answer should be "unknown".

Telephone

Name	Telephone
Leen Desmet	6531421
Leen Desmet	09-23314
Bart Delvaux	5985625
Tom Demans	5845213

Deparment

Name	Department
Bart Delvaux	Computer Sci.
Leen Desmet	Philosophy
Tom Demans	Computer Sci.
David Finner	Biology

Figure 1: A database of phone numbers for a CS department

This example illustrates a situation in which database information is only *locally* complete, and so applying the CWA is not correct, and may lead to wrong conclusions.

The other extreme approach is to make no closure assumption at all. This approach is known as the *Open-World Assumption* (OWA) (Abiteboul & Duschka 1998; Grahne 2002) and is often used in the context of integration of distributed databases, e.g. for mediator-based systems. In this approach, any tuple not in a database table represents an unknown fact, which can be true or false. Also this assumption is clearly not satisfied in the example, where it is known that 1234567 is not a telephone number of Bart Delvaux. Situations of this kind have provided the motivation to introduce relaxed forms of the closed world assumption. Such "local" forms of closed-world assumption have been formalized in different ways by Motro (1989), Levy (1996), Etzioni, Golden, & Weld (1997), Doherty, Lukaszewicz, & Szalas (2000), and Cortés-Calabuig *et al.* (2005).

This paper elaborates the work in (Cortés-Calabuig *et al.* 2005; 2006) that extends the formalism of Levy (1996) which uses *locally closed databases* for handling partially complete information. Such databases consist of a database instance augmented with *Local Closed-World Assumptions* (LCWAs) that are *"specifications of the areas in the real world in which a database contains all true tuples"*. In Example 1, for instance, the LCWA would state that for each telephone number held by a member of the CS department, there is a corresponding entry in the table of Telephone.

Query answering for locally closed databases is considered in (Cortés-Calabuig *et al.* 2006), but the algorithmic approach given there is impractical, since it is based on an *explicit* construction of a 3-valued interpretation that approximates all the (two-valued) models of the database. Moreover, this 3-valued interpretation should be recomputed each time that the database is updated. In this paper, we introduce a simple rewriting algorithm for query answering in locally closed database that avoids those shortcomings. More specifically, the contribution of this paper is twofold:

- We study the basic reasoning tasks in locally closed databases. In particular, we investigate the computational complexity of obtaining *certain* and *possible* answers to queries, and of deciding whether the database has *complete knowledge* on a query, i.e., if its possible and certain answers coincide.
- We present a simple yet efficient rewriting algorithm for *approximating* certain and possible answers in locally closed databases. We also define conditions on the LCWA expressions and on the queries that assure *exact* answers.

The outcome is a study on efficient ways of computing (all the) query answers from locally closed databases.

The Local Closed-World Assumption

In this section we recall the concepts of the LCWA as introduced in (Cortés-Calabuig *et al.* 2005; 2006).

We denote by Σ a finite first-order vocabulary, consisting of sets $\mathcal{R}(\Sigma)$ of predicate symbols and $\mathcal{C}(\Sigma)$ of constants. First-order formulas over Σ are constructed as usual. $\Psi[\bar{x}]$ denotes a formula with free variables that are a subset of $\bar{x}$. Interpretations for Σ (Σ-structures) are also defined as usual. In particular, a Herbrand interpretation has a domain $\mathcal{C}(\Sigma)$, such that each element of $\mathcal{C}(\Sigma)$ interprets itself.

A *database* D over Σ (alternatively, a database instance) consists of a finite set of ground facts with predicate symbols from $\mathcal{R}(\Sigma)$ and arguments from $\mathcal{C}(\Sigma)$ augmented with some infinite supply of constants.

Unlike the standard definitions of databases (see, e.g., Abiteboul, Hull, & Vianu 1995), a database in this paper is neither a structure nor a logical theory, but it is a 'container' of true atoms (or, equivalently, a set of relations), whose full meaning can only be understood by taking into consideration the relevant set of local closed-world assumptions, as defined below:

Definition 1 *A* local closed-world assumption *(LCWA) is an expression of the form*

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

where $P \in \mathcal{R}(\Sigma)$ is called the LCWA's object *and $\Psi[\bar{x}]$, called the LCWA's* window of expertise*, is a first-order formula over Σ.*

The intuitive reading of the expression in Definition 1 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 value of the variables $\bar{x}$ are constrained by Ψ. For this reason we call Ψ a *window of expertise* of the predicate P.

Definition 2 *A* locally closed database $\mathfrak{D}$ *over Σ is a pair $(D, \mathcal{L})$ of a database D over Σ and a finite set $\mathcal{L}$ of local closed-world assumptions over Σ.*

We denote by $dom(\mathfrak{D})$ the active domain of a locally closed database $\mathfrak{D} = (D, \mathcal{L})$. That is, $dom(\mathfrak{D})$ is the finite set consisting of $\mathcal{C}(\Sigma)$ and all constants that appear in D. We define $\Sigma_{\mathfrak{D}}$ as the extension of Σ such that $\mathcal{R}(\Sigma_{\mathfrak{D}}) = \mathcal{R}(\Sigma)$ and $\mathcal{C}(\Sigma_{\mathfrak{D}}) = dom(\mathfrak{D})$.

Example 2 The database of Example 1 consists of two relations. It can be abbreviated as follows:

$$D = \left\{ \begin{array}{l} \mathsf{Tel}(\mathsf{LD}, 6531421), \mathsf{Dept}(\mathsf{BD}, \mathsf{CS}), \\ \mathsf{Tel}(\mathsf{BD}, 5985625), \mathsf{Dept}(\mathsf{LD}, \mathsf{Phil}), \\ \mathsf{Tel}(\mathsf{TD}, 5845213), \mathsf{Dept}(\mathsf{TD}, \mathsf{CS}), \\ \mathsf{Tel}(\mathsf{LD}, \text{09-23314}), \mathsf{Dept}(\mathsf{DF}, \mathsf{Bio}) \end{array} \right\}$$

Some examples of local closed-world assumptions for this database are the following:

1. $\mathcal{LCWA}(\mathsf{Tel}(x, y), \mathsf{Dept}(x, \mathsf{CS}))$ states that all the telephone numbers of the CS department members are known and occur in the database. That is, for every x_0 in $\{x \mid \mathsf{Dept}(x, \mathsf{CS})\}$ (the window of expertise for Tel), all true atoms of the form $\mathsf{Tel}(x_0, y)$ are in the database.
2. $\mathcal{LCWA}(\mathsf{Dept}(x, y), y = \mathsf{CS})$ expresses that all the members of the CS department are known and are mentioned in the database.
3. $\mathcal{LCWA}(\mathsf{Tel}(x, y), x = \mathsf{LD})$ expresses that D contains all telephone numbers of Leen Desmet.

The Meaning of Local Closed-World Assumptions

The intuitive meaning behind the LCWA expressions of Definition 1 can be formally captured using first-order formulas. For this we first introduce the following notation.

Notation 1 *Let D be a database and let P be a predicate that appears in D. Denote by P^D the set of P-tuples in D. Given a tuple $\bar{t}$ of terms, we denote by $P(\bar{t}) \in D$ the formula $\vee_{\bar{a} \in P^D}(\bar{t} = \bar{a})$.*

Definition 3 *Let D be a database over Σ and let $\theta = \mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$ be an LCWA over Σ. The* meaning *of θ in D is given by the formula*

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

Example 3 Consider the database D of Example 2. The meaning of $\theta = \mathcal{LCWA}(\mathsf{Tel}(x, y), \mathsf{Dept}(x, \mathsf{CS}))$ is given by

$$\begin{aligned} \mathcal{M}_D(\theta) = \forall x \forall y \big(&\mathsf{Dept}(x, \mathsf{CS}) \supset \big(\mathsf{Tel}(x, y) \supset \\ &\big((x = \mathsf{LD} \wedge y = 6531421) \vee (x = \mathsf{LD} \wedge y = \text{09-23314}) \vee \\ &(x = \mathsf{BD} \wedge y = 5985625) \vee (x = \mathsf{TD} \wedge y = 5845213)\big)\big)\big). \end{aligned}$$

The two extreme cases of local closed-world assumptions are therefore the following:

- LCWA with a window of expertise that contains all tuples of the domain: $\mathcal{LCWA}(P(\bar{x}), \mathtt{t})$.

  This LCWA expresses that when $P(\bar{x})$ is true in the real world, then it belongs to the database. In other words, this expresses that D has complete knowledge on P.

- LCWA with an empty window of expertise $\mathcal{LCWA}(P(\bar{x}), \mathbf{f})$.

  This LCWA does not expresses any closure. In fact, $\mathcal{M}_D(\mathcal{LCWA}(P(\bar{x}), \mathbf{f}))$ is a tautology for every D.

A useful property of the local closed-world assumption is that any collection of LCWAs on the same predicate may be combined into one (disjunctive) LCWA, that is, the set of LCWA $\theta_i = \mathcal{LCWA}(P(\bar{x}), \Psi_i[\bar{x}_i])$, $i = 1, \ldots, n$, is equivalent to $\theta = \mathcal{LCWA}(P(\bar{x}), \bigvee_{i=1}^{n} \Psi_i[\bar{x}_i])$.

We therefore assume, without a loss of generality, that each predicate symbol P in $\mathcal{R}(\Sigma)$ is an object of exactly *one* LCWA expression, whose window of expertise is denoted Ψ_P.

The Meaning of Locally Closed Databases

The meaning of a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ is expressed by a first-order formula consisting of the conjunction of the database atoms, the meaning of the given local closed-word assumptions, and the following two axioms: Domain Closure and Unique Name axioms:

- $\mathsf{DCA}(dom(\mathfrak{D})) = \forall x(\bigvee_{i=1}^{n} x = C_i)$, (Domain Closure Axiom)
- $\mathsf{UNA}(dom(\mathfrak{D})) = \bigwedge_{1 \leqslant i < j \leqslant n} C_i \neq C_j$, (Unique Name Axiom)

Definition 4 *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database over Σ. The* meaning *of $\mathfrak{D}$ is the first-order sentence*

$$\mathcal{M}(\mathfrak{D}) = \begin{cases} \mathsf{UNA}(dom(\mathfrak{D})) \wedge \mathsf{DCA}(dom(\mathfrak{D})) \wedge \\ \bigwedge_{A \in D} A \wedge \\ \bigwedge_{\theta \in \mathcal{L}} \mathcal{M}_D(\theta). \end{cases}$$

The formula $\mathcal{M}(\mathfrak{D})$ expresses incomplete knowledge about the real world. Thus, in general, it has several models. A $\Sigma_{\mathfrak{D}}$-model M of $\mathcal{M}(\mathfrak{D})$ is also called a model of $\mathfrak{D}$, and this is denoted by $M \models \mathfrak{D}$. If every model of $\mathfrak{D}$ is also a model of a formula φ over $\Sigma_{\mathfrak{D}}$ we say that $\mathfrak{D}$ *entails* φ (or φ *follows* from $\mathfrak{D}$), and denote this by $\mathfrak{D} \models \varphi$.

Below are some important observations regarding the semantics of locally closed databases:

1. Imposing $\mathsf{UNA}(dom(\mathfrak{D})) \wedge \mathsf{DCA}(dom(\mathfrak{D}))$ is a drastic assumption for it implies that a database has complete knowledge on the domain (which often is not the case), and it complicates the use of infinite domains such as integers in the database. However, just like for standard databases, this assumption can be dropped by imposing domain independence on queries and windows of expertise. This topic is beyond the scope of this paper.
2. Since each model of $\mathfrak{D}$ is (isomorphic to) a Herbrand model of $\mathfrak{D}$ with domain $dom(\mathfrak{D})$, for checking such entailments we can restrict ourselves to the Herbrand models of $\mathfrak{D}$. It follows, then, that $\mathfrak{D}$ has a finite set of models (modulo isomorphism). As a consequence, a locally closed database $\mathfrak{D}$ is decidable, that is, there is an effective way of deciding whether any given first-order sentence is a theorem of $\mathfrak{D}$.
3. The database's semantics assumes the soundness of the database instance, since $\mathfrak{D}$ entails each atom in D.
4. The meaning of a locally closed database is always consistent, as the Herbrand interpretation corresponding to D is a model of $\mathcal{M}(\mathfrak{D})$. Moreover, for any other (Herbrand) model I of $\mathfrak{D}$, it holds that $D \subseteq I$.

Locally closed databases generalize both concepts of open world assumption (OWA) (Abiteboul & Duschka 1998; Grahne 2002) and closed-world assumption (CWA) (Reiter 1982). Indeed,

- A locally closed database $\mathfrak{D} = (D, \emptyset)$ corresponds to the database D under the OWA. This database can be represented by the locally closed database $(D, \{\mathcal{LCWA}(P(\bar{x}), \mathbf{f}) \mid P \in \mathcal{R}(\Sigma)\})$ which has an LCWA expression with an empty window of expertise for each predicate in $\mathcal{R}(\Sigma)$.
- The closed-world assumption on D can be represented as $\mathfrak{D} = (D, \{\mathcal{LCWA}(P(\bar{x}), \mathbf{t}) \mid P \in \mathcal{R}(\Sigma)\})$, i.e., by expressing unconditional complete knowledge for each predicate of the database.

Query Answering in Locally Closed Databases

In this section we investigate the computational complexity of querying locally closed databases. As this task turns out to be intractable, we define algorithms for approximating the query answers and show that the complexity of those algorithms is polynomial. Finally, conditions for getting exact answers are specified.

Definition 5 *Let $\mathfrak{D}$ be a locally closed database over Σ, $\mathcal{Q}[\bar{x}]$ a first-order query over Σ (whose free variables are in $\bar{x}$), and $\bar{t}$ a tuple of constants in $dom(\mathfrak{D})$.*

- *$\bar{t}$ is a* certain answer *in $\mathfrak{D}$ for $\mathcal{Q}[\bar{x}]$, if $\mathfrak{D} \models \mathcal{Q}[\bar{t}/\bar{x}]$.*
- *$\bar{t}$ is a* possible answer *in $\mathfrak{D}$ for $\mathcal{Q}[\bar{x}]$, if $\mathfrak{D} \cup \mathcal{Q}[\bar{t}/\bar{x}]$ is satisfiable (equivalently, if $\mathfrak{D} \not\models \neg\mathcal{Q}[\bar{t}/\bar{x}]$).*

In the sequel, given a locally closed database $\mathfrak{D} = (D, \mathcal{L})$, we denote by $Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$ the set of certain answers of $\mathcal{Q}[\bar{x}]$ in $\mathfrak{D}$ and by $Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$ the set of possible answers of $\mathcal{Q}[\bar{x}]$ in $\mathfrak{D}$.

Some Complexity Results

Following the usual measure of complexity in databases, the results below are specified in terms of data complexity, that is, in terms of the size $|D|$ of the database instance (assuming that all the rest is fixed). Accordingly, we consider the following decision problems:

$$\mathrm{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}]) = \{(D, \bar{t}) \mid \bar{t} \in Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])\},$$

$$\mathrm{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}]) = \{(D, \bar{t}) \mid \bar{t} \in Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])\}.$$

Proposition 1 *The decision problem $\mathrm{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ is in NP for all $\mathcal{L}$ and $\mathcal{Q}[\bar{x}]$ and is NP-hard for some of them. $\mathrm{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ is in coNP for each $\mathcal{L}$ and $\mathcal{Q}[\bar{x}]$ and is coNP-hard for some of them.*

Proof (Outline). 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 non-deterministically 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 $\mathrm{Poss}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ is in NP and $\mathrm{Cert}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ is in coNP. Hardness is shown by a reduction from the graph kernel problem as follows: let $\mathcal{R}(\Sigma) = \{Edge/2, Kernel/2, P/1\}$ and consider the following local closed-world assumptions $\mathcal{L}$:

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

where Φ is the following sentence:

$$\left(\begin{array}{l} \forall x \forall y (Kernel(x) \wedge Kernel(y) \supset \neg Edge(x,y)) \bigwedge \\ \forall x (\neg Kernel(x) \supset \exists y (Kernel(y) \wedge Edge(y,x))) \end{array} \right)$$

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 $Kernel^D = P^D = \emptyset$. 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. □

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 $\mathcal{Q}[\bar{x}]$. It can be defined as:

Definition 6 *A locally closed database $\mathfrak{D}$ over Σ has* complete information *on a query $\mathcal{Q}[\bar{x}]$ if for each tuple $\bar{t}$ of constants of $dom(\mathfrak{D})$, 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, i.e., $Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) = Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$. Such queries are of practical importance, since there is no uncertainty on their answers. The idea of complete information on queries is sometimes called *Closed-World Information* (CWI) on a query, and it was formalized by Levy (1996) in the context of incomplete databases. In (Etzioni, Golden, & Weld 1997), this notion was adapted to the logical agents setting.

Observe that the LCWA and CWI are related concepts that capture different phenomena. The LCWA expresses completeness of a set of atoms in a relational database, while the CWI identifies completeness of queries posed to the database. Frequently, LCWAs determine CWI on a query with respect to a given database.

Definition 7 *An expression $\mathcal{LCWA}(P(\bar{x}), \Psi[\bar{x}])$ is* primitive *iff $\Psi[\bar{x}]$ is a Boolean combination of $\mathbf{t}, \mathbf{f}$, and equality atoms. Likewise, a predicate P is primitive iff the LCWA with P as object is primitive.*

Primitive LCWAs induce CWI on appropriate subsets of their object predicates. Consider for instance a locally closed database $\mathfrak{D} = (D, \mathcal{L})$ such that $\mathcal{LCWA}(P(x), x = a) \in \mathcal{L}$. This is a primitive LCWA and it conveys CWI on $P(a)$. Thus $\mathfrak{D} \models P(a)$ or $\mathfrak{D} \models \neg P(a)$, no matter what D is.

For a given set $\mathcal{L}$ of LCWA's and query $\mathcal{Q}[\bar{x}]$, define $\mathrm{CWI}_{\mathcal{L}}(\mathcal{Q}[\bar{x}]) = \{D | (D, \mathcal{L})$ 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 2 *The decision problem $\mathrm{CWI}_{\mathcal{L}}(\mathcal{Q}[\bar{x}])$ is in coNP for each $\mathcal{L}$ and $\mathcal{Q}[\bar{x}]$, and is coNP-hard for some of them.*

The results in this section imply that query answering for locally complete databases is computationally unfeasible in the general case. Our goal in the next section is to develop a method for efficiently computing underestimations of certain answers for queries and overestimations of their possible answers. We also define particular cases in which those answers are exact, that is: the estimations are optimally precise.

Approximate Query Answering in Hierarchically Closed Databases

We introduce an algorithm for approximate query answering. First, we define the family of locally closed databases in the context of which this algorithm can be applied:

Definition 8 *The* LCWA dependency graph *determined by a set $\mathcal{L}$ of LCWAs is a directed graph whose nodes correspond to $\mathcal{R}(\Sigma)$ and there is a directed edge from Q to P iff there exists $\mathcal{LCWA}(P(\overline{x}), \Psi[\overline{x}]) \in \mathcal{L}$ in which Q occurs in Ψ.*

Definition 9 *A* hierarchically closed *database $\mathfrak{D}$, is a pair $(D, \mathcal{L})$, where D is a database instance and $\mathcal{L}$ is a set of LCWAs inducing a cycle-free dependency graph.*

Example 4 The local closed-world assumptions considered in Example 2 for the database instance of Examples 1 and 2, induce a hierarchically closed database.

The transitive closure of a cycle-free LCWA dependency graph is a well-founded strict order on $\mathcal{R}(\Sigma)$. The minimal predicates in this order are those that are the objects of primitive LCWAs. This property will turn out to be essential to warrant termination of the rewriting algorithm. Moreover, hierarchically closed databases cannot contain LCWA expressions in which the object predicates are mutually dependent, e.g., $\mathcal{LCWA}(P, Q)$ and $\mathcal{LCWA}(Q, P)$.

Now, given a hierarchically closed database $\mathfrak{D} = (D, \mathcal{L})$ and a query $\mathcal{Q}[\bar{x}]$ over Σ, we can use Algorithm 1 to rewrite the query in a way that depends on the kind of answers we are interested in. Implications and equivalences in queries are first converted to the language of $\neg, \vee, \wedge$.

Algorithm 1 : Query rewriting function R^+ (resp., R^-)

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 $R^+(\mathcal{Q}[\bar{x}])$ and $R^-(\mathcal{Q}[\bar{x}])$ by simultaneous induction on the structure of $\mathcal{Q}[\bar{x}]$ as follows:
- $R^+(P(\bar{t})) := R^-(P(\bar{t})) := P(\bar{t})$ if P is a primitive predicate $\mathbf{t}, \mathbf{f}$ or $=$.
- $R^+(P(\bar{t})) := P(\bar{t})$ if P in $\mathcal{R}(\Sigma)$.
- $R^-(P(\bar{t})) := (P(\bar{t}) \vee \neg R^+(\Psi_P(\bar{t}))$ if P in $\mathcal{R}(\Sigma)$.
- $R^+(\neg\varphi) := \neg R^-(\varphi)$.
- $R^-(\neg\varphi) := \neg R^+(\varphi)$.
- R^+ and R^- distribute over $\wedge, \vee, \exists, \forall$.

3: **Output:** a query $R^+(\mathcal{Q}[\bar{x}])$ (resp., $R^-(\mathcal{Q}[\bar{x}])$) over Σ.

The idea of the algorithm is as follows. The queries $R^+(\mathcal{Q}[\bar{x}])$ and $R^-(\mathcal{Q}[\bar{x}])$ compute the certain, respectively the possible answers of $\mathcal{Q}[\bar{x}]$. To compute an underestimate for $\mathcal{Q}[\bar{x}]$, positively occurring atoms in $\mathcal{Q}[\bar{x}]$ should be underestimated compared to each model of $\mathfrak{D}$ and negatively occurring atoms overestimated. To compute an overestimate, the inverse should be done. The optimal underestimate of a predicate P is the database table of P itself. Hence, R^+ preserves positively occurring atoms. An overestimate for P is provided by the database table for P plus all tuples which do not certainly belong to the window of expertise of P. Hence, R^+ transforms negatively occurring atoms into $(P(\bar{t}) \vee \neg R^+(\Psi_P(\bar{t}))$. R^- follows the opposite strategy. Notice that R^+ preserves positive queries and R^- negative ones.

The idea behind Algorithm 2 is then that certain (respectively possible) answers of $\mathcal{Q}[\bar{x}]$ w.r.t. $\mathfrak{D}$ can be obtained by directly posing the query $R^+(\mathcal{Q}[\bar{x}])$ (respectively $R^-(\mathcal{Q}[\bar{x}])$) against the database D, now interpreted as a first-order structure.

Algorithm 2 : Approximate certain (resp., possible) query answering

1: **Input:** a hierarchically closed database $\mathfrak{D} = (D, \mathcal{L})$ and a query $\mathcal{Q}[\bar{x}]$ over Σ.
2: Compute $\mathcal{Q}^D[\bar{x}] := R^+(\mathcal{Q}[\bar{x}])$ (resp $\mathcal{Q}^D[\bar{x}] := R^-(\mathcal{Q}[\bar{x}])$)
3: Evaluate $\mathcal{Q}^D[\bar{x}]$ with respect to the database D and active domain $dom(\mathfrak{D})$.
4: **Output:** A set $Cert_A(\mathcal{Q}[\bar{x}])$ (resp., $Poss_A(\mathcal{Q}[\bar{x}])$) of tuples for the evaluation in (3).

Example 5 *Let $\mathcal{R}(\Sigma) = \{P, Q\}$ and $\mathfrak{D} = (D, \mathcal{L})$ where $\mathcal{L} = \{\mathcal{LCWA}(P(x), Q(x)),\ \mathcal{LCWA}(Q(x), x = c)\}$ and $D = \{P(a), Q(c)\}$. $\mathfrak{D}$ is clearly hierarchically closed. Now, consider the query $\mathcal{Q}[x] = P(x)$.*

- *For certain query answers, we evaluate $R^+(P(x)) = P(x)$ with respect to D. Thus, $Cert_A(P(x)) = \{a\}$.*
- *For possible answers we start with $R^-(P(x)) = (P(x) \vee \neg R^+(Q(x))$. After evaluating $R^+(Q(x))$ we get $P(x) \vee \neg Q(x)$. Evaluating this expression with respect to D we have that $Poss_A(P(x)) = dom(\mathfrak{D}) \setminus \{c\}$.*

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

The following proposition shows that Algorithm 2 indeed approximates query answering on $\mathfrak{D}$ in the sense that it underestimates certain answers and overestimates possible answers:

Theorem 1 (Soundness) *Let $\mathfrak{D}$ be a hierarchically closed database. For a query $\mathcal{Q}$, $Cert_A(\mathcal{Q}[\bar{x}]) \subseteq Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}]) \subseteq Poss_A(\mathcal{Q}[\bar{x}])$.*

Proof (Outline). Assume that for each predicate $P \in \mathcal{R}(\Sigma)$, we have relations P_l, P_u such that for each model M of $\mathfrak{D}$, $P_l \subseteq P^M \subseteq P_u$. Define for a query $\mathcal{Q}[\bar{x}]$, the query $\mathcal{Q}_c[\bar{x}]$ obtained by substituting P_l for positive and P_u for negative occurrences of P. Likewise, define $\mathcal{Q}_p[\bar{x}]$ by substituting P_u for positive and P_l for negative occurrences. By a standard monotonicity argument, it can be shown that if $\bar{t}$ is an answer to $\mathcal{Q}_c[\bar{x}]$, then $\bar{t}$ is a certain answer to $\mathcal{Q}[\bar{x}]$, and if $\bar{t}$ is a possible answer to $\mathcal{Q}[\bar{x}]$, then it is an answer to $\mathcal{Q}_p[\bar{x}]$. If we look at $R^+(\mathcal{Q}[\bar{x}])$, this rewriting operation takes care that positive occurrences of P are evaluated by P^D and negative occurrences by the relation P_u consisting of all answers of $R^-(P(\bar{y})) \equiv (P(\bar{y}) \vee \neg R^+(\Psi_P(\bar{y}))$ in D. By the previous observation, it suffices to show that for each model M of $\mathfrak{D}$, $P^D \subseteq P^M \subseteq P_u$. The first of these inequalities is obvious. It can be shown that $P^M \subseteq P_u$ by an inductive argument on the dependency relation. On an intuitive level, this is plausible since P_u consists of all $\bar{d}$ in P^D plus all $\bar{d}$ which satisfy $\neg R^+(\Psi(\bar{d}))$, i.e., which do not certainly belong to the window of expertise of P. □

Next we show that our algorithm retains tractability.

Proposition 3 *Algorithm 1 always terminates and computes a query whose size (num. of atoms) is constant in $|D|$.*

Proof (Outline). The size of a rewritten query is obviously constant in $|D|$ since R^+ and R^- do not depend on D. Termination of the rewriting process implemented by R^+ and R^- follows from the fact that $\mathfrak{D}$ is hierarchically closed. □

A direct consequence of this proposition is the following complexity result.

Theorem 2 (Complexity) *Given a hierarchically closed database $\mathfrak{D}$ and a query $\mathcal{Q}$, the computation time of $Cert_A(\mathcal{Q}[\bar{x}])$ and $Poss_A(\mathcal{Q}[\bar{x}])$ by Algorithm 2 is polynomial in $|D|$.*

As degenerate cases can be designed in which all precision of Algorithm 2 is lost, it is unfeasible to provide bounds on the approximation in the general case. Instead, we show the optimality of the algorithm for broad classes of query–database pairs, defined by syntactical properties that are easily verified. As a trivial case, our method is optimal for all positive queries, since R^+ preserves such queries and they can be solved optimally in the least model D of $\mathfrak{D}$. Below, we consider two other classes.

Theorem 3 (Completeness) *Let $\mathfrak{D} = (D, \mathcal{L})$ be a hierarchically closed database such that every window of expertise in $\mathcal{L}$ is a conjunction of literals. If query $\mathcal{Q}[\bar{x}]$ is a conjunction of literals, then $Cert_A(\mathcal{Q}[\bar{x}]) = Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$. If $\mathcal{Q}[\bar{x}]$ is a disjunction of literals, then $Poss_A(\mathcal{Q}[\bar{x}]) = Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$.*

Proof (Outline). It can be shown that in case all windows of expertise are conjunctions of literals, then for each predicate P, the relation P_u consisting of the answers of $R^-(P(\bar{y}))$ w.r.t. D is optimally precise, in the sense that if $\bar{t} \in P_u$, then there is a model M of $\mathfrak{D}$ such that $M \models P(\bar{t})$. This can be proven by induction on the dependency graph.

Now, let $\mathcal{Q}[\bar{x}]$ be a conjunction of literals. Assume that $\bar{d}$ is not an answer to $R^+(\mathcal{Q}[\bar{x}])$ w.r.t. D. We need to show that there is a model M of $\mathfrak{D}$ such that $M \not\models \mathcal{Q}[\bar{d}]$. Since $R^+(\mathcal{Q}[\bar{d}])$ is false in D, there is a conjunct C of $\mathcal{Q}[\bar{d}]$ such that $R^+(C)$ is false in D. Either C is an atom in which case

$R^+(C) = C$, and C is false in D. In this case, we see that for $M = D$, $M \not\models \mathcal{Q}[\bar{d}]$. In the other case, C is a negative literal $\neg P(\bar{t})$, and $R^+(C) = \neg R^-(P(\bar{t}))$ which is false in D. It follows that $\bar{t} \in P_u$, which entails that for some model M of $\mathfrak{D}$, $M \models P(\bar{t})$ and hence $M \not\models \mathcal{Q}[\bar{d}]$. □

Theorem 4 *Let $\mathfrak{D} = (D, \mathcal{L})$ be a locally closed database. If $\mathcal{L}$ consists only of primitive local closed-world assumptions, then for each conjunction of literals $\mathcal{Q}[\bar{x}]$, $Cert_A(\mathcal{Q}[\bar{x}]) = Cert_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$, and for each disjunction of literals $\mathcal{Q}[\bar{x}]$, $Poss_A(\mathcal{Q}[\bar{x}]) = Poss_{\mathfrak{D}}(\mathcal{Q}[\bar{x}])$.*

Proof (Outline). If all LCWAs are primitive, then the database is hierarchically closed and Theorem 3 applies. □

For hierarchically closed databases with conjunctive windows of expertise (see, e.g., Examples 1 and 2) or with primitive LCWAs (like those in items 1 and 2 of Example 2) we therefore have a sound and complete algorithm for certain conjunctive query answering, which is polynomial in $|D|$.

Related Work

There are different methods to represent partial completeness in database systems and to determine whether query answers are complete even though the database is incomplete. In the approach of Motro (1989), for instance, partial completeness is specified as a set of *views* expressed as formulas in the database language, each of which can be solved completely by querying the database. In this approach, CWI on a given query holds if this query can be rewritten in terms of the views. The semantics of this method differs from ours, it is limited to conjunctive positive views and queries, and the issue of retrieving possible answers is not addressed.

In (Levy 1996), the notion of local completeness is semantically characterized in terms of *virtual* relations P and *available* relations P', representing, respectively, the facts that hold in the 'real world' and in the database instance. A *local completeness statement* then specifies that the available and virtual predicates coincide in some window of expertise represented by a conjunctive query. Our approach is a syntactic variant but extends it by allowing general windows of expertise. For instance, borrowing Levy's original example, the statement $LC(\mathsf{Movie}', \mathsf{Movie}, \mathsf{Year} \geqslant 1965)$ is equivalent to $\mathcal{LCWA}(Movie(x, year), year \geqslant 1995)$ in our approach. Levy's main contribution is a polynomial algorithm to determine whether a set of local completeness constraints implies CWI on a positive conjunctive query. The issue of retrieving certain and possible answers from general queries is again not addressed.

The idea of approximating query answers has been exploited in the context of information integration. In (Grahne & Mendelzon 1999), the authors present tableaux techniques to retrieve certain and/or possible answers to queries posed to the mediated schema. Since the data sources available to the mediator may not contain the precise information to answer mediator queries, the algorithm approximates answers. Although similar in spirit to the framework we present here, the methods developed in (Grahne & Mendelzon 1999) are specifically intended for mediator-based systems, and their application in the context of locally complete databases is yet to be investigated.

Conclusion

The need to weaken the CWA in database systems and the ability to efficiently reason with partially complete databases is a major goal whose importance should be obvious. In this paper, we improved the framework introduced in (Levy 1996) and afterwards in (Cortés-Calabuig *et al.* 2006), as a step towards a logical reconstruction of a theory for locally complete databases. In this respect, we explored the computational complexity of reasoning with locally closed databases, developed efficient approximate methods to estimate query answers, and defined a sound and under certain syntactical conditions also complete algorithm for producing certain and possible query answers. Future work includes lifting the domain closure assumption, incorporation of integrity constraints, and extending the current framework to deductive databases.

References

Abiteboul, S.; and Duschka, O. 1998. Complexity of answering queries using materialized views. In *Proc. 17th PODS*, 254–263.

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley Publishing Company.

Cortés-Calabuig, A.; Denecker, M.; Arieli, O.; Nuffelen, B. V.; and Bruynooghe, M. 2005. On the local closed-world assumption of data-sources. In *Proc. 8th LPNMR*, LNCS 3662, 145–157. Springer.

Cortés-Calabuig, A.; Denecker, M.; Arieli, O.; and Bruynooghe, M. 2006. Representation of partial knowledge and query answering in locally complete databases. In *Proc. 13th LPAR*, LNCS 4246, 407–421. Springer.

Doherty, P.; Lukaszewicz, W.; and Szalas, A. 2000. Efficient reasoning using the local closed-world assumption. In *Proc. 9th AIMSA*, LNCS 2407, 49–58. Springer.

Etzioni, O.; Golden, K.; and Weld, D. 1997. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence* 89(1-2):113–148.

Grahne, G., and Mendelzon, A. 1999. Tableau techniques for querying information sources through global schemas. In *Proc. 7th ICDT*, LNCS 1540, 332–347. Springer.

Grahne, G. 2002. Information integration and incomplete information. *IEEE Data Engineering Bulletin* 25(3):46–52.

Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proc. of the 21st PODS*, 233–246.

Levy, A. 1996. Obtaining complete answers from incomplete databases. In *Proc. 22nd VLDB*, 402–412.

Motro, A. 1989. Integrity = validity + completeness. *ACM Trans. Database Syst.* 14(4):480–502.

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