A Tractability Result for Reasoning with Incomplete First-Order Knowledge Bases

Yongmei Liu and **Hector J. Levesque**
Department of Computer Science
University of Toronto
Toronto, ON, Canada M5S 3G4
{yliu, hector}@cs.toronto.edu

Abstract

In previous work, Levesque proposed an extension to classical databases that would allow for a certain form of incomplete first-order knowledge. Since this extension was sufficient to make full logical deduction undecidable, he also proposed an alternative reasoning scheme with desirable logical properties. He also claimed (without proof) that this reasoning could be implemented efficiently using database techniques such as projections and joins. In this paper, we substantiate this claim and show how to adapt a bottom-up database query evaluation algorithm for this purpose, thus obtaining a tractability result comparable to those that exist for databases.

1 Introduction

As argued in [Levesque, 1998], there is only one deductive technique efficient enough to be feasible on knowledge bases (KBs) of the size seemingly required for common-sense reasoning: the deduction underlying classical database query evaluation. And yet, databases by themselves are too restricted to serve as the representational scheme for common-sense reasoning, since they require, among other things, complete knowledge of the domain. Levesque proposed a generalization of databases called *proper knowledge bases*, which allow for a limited form of incomplete knowledge. Despite the limitations, however, the deduction problem for proper KBs is no longer even decidable. Levesque proposed an alternative reasoning procedure V for proper KBs that was logically sound and, when the query was in a certain normal form, logically complete. Moreover, he argued that it should be possible to implement V for very large KBs using database techniques. However, no proof was given.

In this paper, we examine proper KBs and the V procedure more closely, and we prove a tractability result for this type of logical reasoning that are comparable to those that exist for classical database query evaluation. In particular, we adapt a bottom-up database retrieval algorithm to the case of proper KBs. Thus, what we show here is that in some cases it is indeed possible to reason efficiently with incomplete knowledge in a logically sound and complete way, or at least as efficiently as we can with a database.

The rest of the paper is organized as follows. In the next section, we review proper KBs and V, prove a new property of V, *i.e.* locality, and define answers to open queries. In Section 3, we review the complexity of database query evaluation, and present a polynomial time algorithm for evaluating k-guarded formulas. In Section 4, we show how to use this algorithm to evaluate queries wrt proper KBs and hence obtain a tractability result. In Section 5, we illustrate this query evaluation method for proper KBs with some example queries. Finally in Section 6, we describe some future work.

2 Proper KBs and V

2.1 Basic definitions

We use a standard first-order logical language $\mathcal{L}$ with a countably infinite set of constants $\mathcal{C} = \{c_1, c_2, \ldots\}$, no other function symbols, and a distinguished equality predicate. We have in mind that in any interpretation, $=$ should be understood as identity, that the constants will be unique names. This can be captured by a set of axioms $\mathcal{E}$, consisting of the axioms of equality and the set of formulas $\{c_i \neq c_j \mid i \neq j\}$.

In what follows, ρ ranges over atoms (excluding equality) whose arguments are distinct variables; e ranges over *ewffs*, *i.e.* quantifier-free formulas whose only predicate is equality. $\forall\alpha$ denotes the universal closure of formula α; α^x_d denotes α with all free occurrences of x replaced by constant d. The *width* of α is the maximal number of free variables in its subformulas. The *quantifier rank* $qr(\alpha)$ of α is the depth of nesting of quantifiers in α.

Definition 2.1 *A KB Σ is proper if $\mathcal{E} \cup \Sigma$ is consistent and Σ is a finite set of formulas of the form $\forall(e \supset \rho)$ or $\forall(e \supset \neg\rho)$.*

To illustrate the idea of a proper KB, imagine a scenario involving a robot security guard keeping track of the occupants of the rooms of a building. The robot can find out that someone has entered or left a room, and by going into a room, find out who the occupants are. We can express what the robot might know using a proper KB:[1]

- Alice is alone in Room 1.
  $\forall xy.\ x = \mathit{alice} \land y = \mathit{room1} \supset \mathit{In}(x, y)$
  $\forall xy.\ x = \mathit{alice} \land y \neq \mathit{room1} \supset \neg\mathit{In}(x, y)$
  $\forall xy.\ x \neq \mathit{alice} \land y = \mathit{room1} \supset \neg\mathit{In}(x, y)$

[1] We think of these as beliefs about some initial state of the world. It should be possible to generalize the notion of proper KB to deal with state change. See Section 6 for discussion.

- Bob and possibly others are in Room 2.
  $\forall xy.\, x = \textit{bob} \wedge y = \textit{room2} \supset \textit{In}(x,y)$
  $\forall xy.\, x = \textit{bob} \wedge y \neq \textit{room2} \supset \neg\textit{In}(x,y)$
- Carol is not in the building.
  $\forall xy.\, x = \textit{carol} \supset \neg\textit{In}(x,y)$
- Only Bob and Carol go into Room 3.
  $\forall xy.\, x \neq \textit{bob} \wedge x \neq \textit{carol} \wedge y = \textit{room3} \supset \neg\textit{In}(x,y)$

Note that this information cannot be expressed in a traditional database where, *e.g.* we cannot leave open the occupants of Room 2. On the other hand, facts requiring disjunction or existential quantification cannot be expressed as part of a proper KB, *e.g.* every room having at most three occupants.

It is not hard to see that the problem of determining whether a sentence is logically entailed by a proper KB is undecidable, since when the KB is empty, this reduces to classical validity.[2] Levesque [1998] proposed the reasoning procedure V instead. Given a proper KB and a query, V returns one of three values 0 (known false), 1 (known true), or $\frac{1}{2}$ (unknown) as follows:

1. $V[\Sigma, \rho\theta] = \begin{cases} 1 & \text{if there is a } \forall(e \supset \rho) \in \Sigma \\ & \text{ such that } V[\Sigma, e\theta] = 1 \\ 0 & \text{if there is a } \forall(e \supset \neg\rho) \in \Sigma \\ & \text{ such that } V[\Sigma, e\theta] = 1 \\ \frac{1}{2} & \text{otherwise} \end{cases}$
2. $V[\Sigma, t = t'] = 1$ if t is identical to t', and 0 otherwise;
3. $V[\Sigma, \neg\alpha] = 1 - V[\Sigma, \alpha]$;
4. $V[\Sigma, \alpha \vee \beta] = \max\{V[\Sigma, \alpha], V[\Sigma, \beta]\}$;
5. $V[\Sigma, \exists x\alpha] = \max\{V[\Sigma, \alpha^x_d] \mid d \in H^+(\Sigma \cup \{\alpha\})\}$, where $H^+(S)$ is the union of the constants in S, and a constant not in S, say the lexicographically smallest one.

This V procedure always terminates, and is logically sound (*i.e.* if it returns 1 (resp. 0), then the query (resp. its negation) is logically entailed by $\mathcal{E} \cup \Sigma$). Moreover, Levesque defined a certain normal form called $\mathcal{NF}$, and proved that for queries in $\mathcal{NF}$, the V procedure is also logically complete (so that, for instance, V returns 1 iff the query is logically entailed). Finally, Levesque showed that in the propositional case, every query could be equivalently expressed in $\mathcal{NF}$, and conjectured that this was also true in the first-order case.

2.2 Invariance under renaming

The first difficulty that arises when attempting to reason with proper KBs is the fact that unlike with databases, we cannot fix the domain in advance, or even put an upper bound on its size. For example, if Σ consists of just $\forall x(x \neq a \supset P(x))$, and α_n is the sentence of $\mathcal{L}$ that says that there are at least n distinct instances of P, then $V[\Sigma, \alpha_n] = 1$ for all n. However, we can take advantage of the fact that for this Σ, all constants other than a behave the same. This will allow us to define finite versions of answers to open queries, as shown below.

Notation Let α be a formula and S a set of formulas. Let $*$ be a bijection from $\mathcal{C}$ to $\mathcal{C}$. We use α^* to denote α with each c replaced by c^*, and use S^* to denote $\{\alpha^* \mid \alpha \in S\}$.

[2] The classical validity problem is undecidable even when there are no function symbols.

Lemma 2.2 *Let e be an ewff, and let $*$ be a bijection from $\mathcal{C}$ to $\mathcal{C}$. Then $\mathcal{E} \models e$ iff $\mathcal{E} \models e^*$.*

Theorem 2.3 *Let Σ be a proper KB and α a sentence. Let $*$ be a bijection from $\mathcal{C}$ to $\mathcal{C}$. Then $V[\Sigma, \alpha] = V[\Sigma^*, \alpha^*]$.*

Proof: The proof is by induction on the structure of α. The case for atoms uses Lemma 2.2. The cases for equalities, negations and disjunctions are trivial. Now consider $\exists x\alpha$. Let $d \in H^+(\Sigma \cup \{\alpha\})$. If d appears in Σ or α, by induction, $V[\Sigma, \alpha^x_d] = V[\Sigma^*, (\alpha^x_d)^*] = V[\Sigma^*, (\alpha^*)^x_{d^*}]$. Otherwise, d^* does not appear in Σ^* or α^*. Now let $c \in H^+(\Sigma^* \cup \{\alpha^*\})$ such that c does not appear in Σ^* or α^*. Let $\star$ be the composition of $*$ and the bijection that swaps c and d^* and leaves the rest constants unchanged. Then $d^\star = c$; $\Sigma^\star = \Sigma^*$, $\alpha^\star = \alpha^*$, since neither d^* nor c appears in Σ^* or α^*. By induction, $V[\Sigma, \alpha^x_d] = V[\Sigma^\star, (\alpha^\star)^x_{d^\star}] = V[\Sigma^*, (\alpha^*)^x_c]$. Thus $V[\Sigma, \exists x\alpha] = \max\{V[\Sigma, \alpha^x_d] \mid d \in H^+(\Sigma \cup \{\alpha\})\}$= $\max\{V[\Sigma^*, (\alpha^*)^x_d] \mid d \in H^+(\Sigma^* \cup \{\alpha^*\})\}$=$V[\Sigma^*, (\exists x\alpha)^*]$. ■

Corollary 2.4 *Let Σ be a proper KB and α a formula with a single free variable x. Let c and d be two constants that do not appear in Σ or α. Then $V[\Sigma, \alpha^x_c] = V[\Sigma, \alpha^x_d]$.*

Proof: Follows from the theorem by taking $*$ as the bijection that swaps c and d and leaves the rest constants unchanged. ■

2.3 Locality

A nice property of V is its locality, *i.e.*, for any proper KB Σ and query α, the value of $V[\Sigma, \alpha]$ only depends on those sentences of Σ that mention some predicate in α.

Definition 2.5 *Let Σ be a proper KB and α a query. The restriction of Σ wrt α, denoted $\Sigma|\alpha$, is the set $\{\forall(e \supset l) \in \Sigma \mid$ the predicate of l appears in $\alpha\}$.*

Theorem 2.6 $V[\Sigma, \alpha] = V[\Sigma|\alpha, \alpha]$.

Proof: The proof is by induction on the structure of α. Here we only prove the case for quantifications. $V[\Sigma, \exists x\alpha] =$ $\max\{V[\Sigma, \alpha^x_d] \mid d \in H^+(\Sigma \cup \{\alpha\})\}$ = (by induction) $\max\{V[\Sigma|\alpha, \alpha^x_d] \mid d \in H^+(\Sigma \cup \{\alpha\})\}$ = (by Corollary 2.4) $\max\{V[\Sigma|\alpha, \alpha^x_d] \mid d \in H^+(\Sigma|\alpha \cup \{\alpha\})\} = V[\Sigma|\alpha, \exists x\alpha]$. ■

In fact, we can go further than this: roughly speaking, we need only consider those sentences $\forall(e \supset l) \in \Sigma$ such that after substitutions dictated by e, the l would unify with some literal in the query α. For example, $\forall xy(y = b \supset P(x,y))$ is not needed to answer the query $\exists xP(x,a)$. For KBs that are large because they have facts about a large number of individuals, this optimization can reduce significantly the size of the KB fragment needed to answer the query correctly.

2.4 Answers to open queries

Definition 2.7 *Let Σ be a proper KB and $\alpha(\bar{x})$ a formula with free variables. The answer to α wrt Σ is the set $\alpha(\Sigma) = \{\bar{c} \mid V[\Sigma, \alpha(\bar{c})] = 1\}$.*

However, $\alpha(\Sigma)$ may well be infinite. Fortunately, we can find a finite representation for it.

Definition 2.8 *Let Σ be a proper KB and α a formula. Then $H(\Sigma, \alpha)$ is the finite set of constants consisting of those mentioned in Σ or α and the r next ones, lexicographically, where r is the quantifier rank of $\forall\alpha$.*

Definition 2.9 *Let Σ be a proper KB and $\alpha(\bar{x})$ a formula with free variables. The finite answer to α wrt Σ is the set $\alpha_f(\Sigma) = \{\bar{c} \in H(\Sigma|\alpha, \alpha) \mid V[\Sigma, \alpha(\bar{c})] = 1\}$.*

The following theorem justifies $\alpha_f(\Sigma)$ as a finite representation for $\alpha(\Sigma)$.

Theorem 2.10 *Let Σ be a proper KB and $\alpha(\bar{x})$ a formula. Then for any constants $\bar{c}$, $\bar{c} \in \alpha(\Sigma)$ iff $\bar{c}^\star \in \alpha_f(\Sigma)$, where $\bar{c}^\star$ is like $\bar{c}$ except that constants not in $\Sigma|\alpha \cup \{\alpha\}$ are replaced by unique representatives not in $\Sigma|\alpha \cup \{\alpha\}$ from $H(\Sigma|\alpha, \alpha)$.*

Proof: We can expand $\star$ to a bijection $*$ from $\mathcal{C}$ to $\mathcal{C}$ such that $c^* = c$ for any c that appears in $\Sigma|\alpha$ or α. Then $V[\Sigma, \alpha(\bar{c})]$= (by locality) $V[\Sigma|\alpha, \alpha(\bar{c})]$ = (by Theorem 2.3) $V[(\Sigma|\alpha)^*, (\alpha(\bar{c}))^*]$, *i.e.*$V[\Sigma|\alpha, \alpha(\bar{c}^\star)]$ =$V[\Sigma, \alpha(\bar{c}^\star)]$. ■

3 The complexity of database queries

3.1 An overview

The complexity of query evaluation has been one of the main pursuits of database theory. Traditionally, there are two complexity measures: combined complexity and data complexity [Vardi, 1982]. Combined complexity is measured in terms of the combined size of the database and the query. Chandra and Merlin [1977] proved that the combined complexity of conjunctive queries (queries expressed by first-order formulas that are of the form $\exists \bar{x}(\alpha_1 \wedge \ldots \wedge \alpha_n)$ where α_i are atoms) is NP-complete; Vardi [1982] proved that the combined complexity of first-order queries is PSPACE-complete. However, the main factor responsible for this high complexity is the size of the query and not the size of the database. This is contrary to the situation in practice where we normally evaluate small queries against large databases. Data complexity measures the complexity of query evaluation solely in terms of the size of the database and treats the size of the query as a constant. A folk result is that first-order queries can be evaluated in time $n^{O(l)}$, where n is the size of the database and l is the size of the query. Thus the data complexity of first-order queries is in PTIME. However, such a complexity can hardly qualify as tractable even if the exponent is small, say 5.

Yannakakis [1995] was the first to suggest that parameterized complexity [Downey and Fellows, 1995] might be an appropriate complexity measure for database query evaluation. Query evaluation is fixed-parameter tractable if there is a computable function $f : \mathbb{N} \to \mathbb{N}$ and a constant c such that the problem can be solved in time $f(l) \cdot n^c$, where l and n are the size of the query and the database, respectively. However, Papadimitriou and Yannakakis [1999] proved that the parameterized complexity of conjunctive queries is W[1]-complete and thus most likely not fixed-parameter tractable. We refer the reader to [Grohe, 2002] for a survey on parameterized complexity in database theory.

Therefore database query evaluation in general is hard with respect to both combined and parameterized complexity. And yet, database queries do work quite well in practice, even for very large databases. A careful examination of the hardness results show that they often depend on queries that are somewhat atypical in applications, *e.g.* databases representing graphs and queries asking for the existence of cliques.

Naturally, many research efforts have gone into finding classes of queries that can be proven tractable, even in the worst cases. The earliest result of this form, due to Yannakakis [1981], showed that acyclic conjunctive queries can be evaluated in polynomial time. This result has been extended in several ways. The first extension, due to Chekuri and Rajaraman [1997], showed that conjunctive queries with bounded treewidth are tractable. Later, Gottlob *et al.* [1999] introduced the notion of hypertree width and showed that conjunctive queries with bounded hypertree width are tractable; bounded hypertree width generalizes the notions of acyclicity and bounded treewidth. Recently, Flum *et al.* [2001] generalized the notions of acyclicity and bounded treewidth from conjunctive queries to nonrecursive stratified Datalog (NRSD), which is known to have the same expressive power as all of first-order logic, and showed that acyclic and bounded treewidth NRSD are tractable. Inspired by their work, Gottlob *et al.* [2001] extended the notion of hypertree width to NRSD, and obtained a nice logical characterization of hypertree width: they showed that the k-guarded fragment of first-order logic has the same expressive power as NRSD of hypertree width at most k. Thus k-guarded first-order logic turns out to be the largest tractable class of queries so far.

3.2 An evaluation algorithm

The guarded fragment GF was introduced by Andréka *et al.* [1996], and has recently received a lot of attention. The idea is that any existentially quantified subformula ϕ must be conjoined with a guard, *i.e.* an atom containing all free variables of ϕ. Gottlob *et al.* [2001] extended GF to the k-guarded fragment GF_k where up to k atoms may jointly act as a guard, and proved that GF_k-formulas can be evaluated in time $O(l^2 n^k)$ where l and n are as before. The proof is by transforming GF_k-formulas into k-guarded NRSD programs.

In this section, we introduce the k-guarded fragment of $\mathcal{L}$, and explicitly present a polynomial algorithm for evaluating k-guarded formulas against databases and analyze its complexity. We will use this algorithm to evaluate k-guarded formulas with respect to proper KBs.

Definition 3.1 *The k-guarded fragment GF_k of $\mathcal{L}$ is the smallest set of formulas such that*

1. *GF_k contains atoms and equalities;*
2. *GF_k is closed under Boolean operations;*
3. *GF_k contains $\exists \bar{x}(G_1 \wedge \ldots \wedge G_m \wedge \phi)$, provided that the G_i are atoms or equalities, $m \leq k$, $\phi \in GF_k$, and the free variables of ϕ appear among the G_i.*

Definition 3.2 *The strictly k-guarded fragment SGF_k are the formulas in GF_k of the form $\exists \bar{x}(G_1 \wedge \ldots \wedge G_m \wedge \phi)$. We allow the degenerate cases where $\bar{x}$ is empty or $m = 0$ or ϕ is* TRUE.

Note that any k-guarded sentence is strictly k-guarded.

The evaluation algorithm below will take a strictly k-guarded formula as the query. It turns out that any formula ϕ can be transformed in linear time into an equivalent SGF_k-formula for some k, using guards of the form $x = x$, if necessary. For example, note that $R(x, y, z) \wedge S(z, u, v)$ can be formulated in SGF_2, but $\neg R(x, y, z) \wedge \neg S(z, u, v)$ only in SGF_5.

The evaluation algorithm will have worst-case time complexity that is exponential only in the k. For fixed k, the algorithm is polynomial, but perhaps impractical when k is large. An upper bound on k is the width of ϕ. Obviously, if ϕ has at most k distinct variables, the width of ϕ is no more than k.

Before presenting the evaluation algorithm, we first recall some basic notions of relational database theory. A database instance is simply a logical structure (or interpretation) with a finite domain. Let $\mathcal{A}$ be such a structure with domain A. Let X and Y be sets of variables. An X-relation R over A, is a finite set of mappings γ from X into A, which we write as $\{\gamma : [X \to A]\}$. If $Y \subseteq X$, the Y-projection of an X-relation R, which we write $\pi_Y(R)$, is the set $\{\gamma|Y \mid \gamma \in R\}$, where $\gamma|Y$ is the mapping restricted to Y. Let R be an X-relation and S a Y-relation. The join of R and S is the $X \cup Y$-relation

$$R \bowtie S = \{\gamma : [X \cup Y \to A] \mid \gamma|X \in R,\ \gamma|Y \in S\}.$$

The join of R with the complement of S is the $X \cup Y$-relation

$$R \bowtie^c S = \{\gamma : [X \cup Y \to A] \mid \gamma|X \in R,\ \gamma|Y \notin S\}.$$

Let ϕ be a formula with free variables $X = \{x_1, \ldots, x_n\}$. The *answer* to ϕ wrt $\mathcal{A}$ is the X-relation

$$\phi(\mathcal{A}) = \{\gamma : [X \to A] \mid \mathcal{A} \models \phi(\gamma(x_1), \ldots, \gamma(x_n))\}.$$

Here is the algorithm for computing answers:

Procedure Eval$(\mathcal{A}, \phi)$
Input: A structure $\mathcal{A}$ such that the domain and all relations are sorted, and a formula $\phi \in$ SGF$_k$
Output: $\phi(\mathcal{A})$

suppose $\phi(\bar{x}) = \exists\bar{y}(G_1 \wedge \ldots \wedge G_m \wedge \psi)$
return $\pi_{\bar{x}}\{\text{Eval}(\mathcal{A}, G_1(\mathcal{A}) \bowtie \ldots \bowtie G_m(\mathcal{A}), \psi')\}$
where ψ' is the result of pushing $\neg$s in ψ so that they are in front of atoms, equalities, or existentials.

Procedure Eval$(\mathcal{A}, R, \phi)$ is defined recursively by:

1. If ϕ is an atom or an equality, then Eval$(\mathcal{A}, R, \phi) = R \bowtie \phi(\mathcal{A})$;
2. If ϕ is negation of an atom or an equality, then Eval$(\mathcal{A}, R, \phi) = R \bowtie^c \phi(\mathcal{A})$;
3. Eval$(\mathcal{A}, R, (\phi \wedge \psi)) =$ Eval$(\mathcal{A}, R, \phi) \cap$ Eval$(\mathcal{A}, R, \psi)$;
4. Eval$(\mathcal{A}, R, (\phi \vee \psi)) =$ Eval$(\mathcal{A}, R, \phi) \cup$ Eval$(\mathcal{A}, R, \psi)$;
5. Eval$(\mathcal{A}, R, \exists\bar{y}\phi) = R \bowtie$ Eval$(\mathcal{A}, \exists\bar{y}\phi)$;
6. Eval$(\mathcal{A}, R, \neg\exists\bar{y}\phi) = R \bowtie^c$ Eval$(\mathcal{A}, \exists\bar{y}\phi)$.

Theorem 3.3 *Given a sorted structure $\mathcal{A}$ and a formula $\phi \in$ SGF$_k$, $\phi(\mathcal{A})$ can be computed in time $O(ln^k)$, where l is the size of ϕ, and n is the size of the database.*

Proof: For each of the l literals and logical operators in the query, we perform either a join, a join with complement, an intersection, or a union. Note that when we perform a join or a join with complement within Eval$(\mathcal{A}, R, \phi)$, the attributes of the second argument is always a subset of those of the first, since for guarded formulas, the free variables in ϕ are a subset of those in R. By induction, Eval$(\mathcal{A}, R, \phi)$ always returns a relation that is a subset of R. Moreover, the R is obtained from joins of at most k guards, and so has size at most n^k. Assuming all relations start in sorted form, the results of all operations can be kept in sorted form, and computed in time $O(n^k)$. Thus, the time overall is $O(ln^k)$. ■

4 The complexity of V

In this section, we consider how hard it is to compute V. Not surprisingly, V is no easier than database query evaluation. What is more significant is that under reasonable assumptions, it is not much harder either.

4.1 Intractability results

Theorem 4.1 *The combined complexity of V is NP-hard for conjunctive queries, and PSPACE-hard for first-order queries. The parameterized complexity of V is W[1]-hard for conjunctive queries.*

Proof: The proof is essentially the same as that for database query evaluation. For conjunctive queries, the reduction is from the clique problem. For first-order queries, the reduction is from QBF (Quantified Boolean Formula). ■

4.2 A tractability result

We now present a tractability result for V. The basic idea is to reduce V to database query evaluation and use the algorithm Eval from Section 3. To do this efficiently, we would like to use database operations as much as possible, and only reason with the formulas e in $\forall(e \supset l)$ when necessary. For each predicate P appearing in a query α, we will consider storing two relations P^+ and P^- depending on the form of Σ.

Definition 4.2 *The ewff defining a predicate P denoted e_P, is the disjunction of all e such that $\forall\bar{x}(e(\bar{x}) \supset P(\bar{x})) \in \Sigma$. The ewff defining $\neg P$ denoted $e_{\neg P}$ is analogous.*

Definition 4.3 *We distinguish among three cases for an ewff:*

1. *e is relational if it is of the form $\bar{x} = \bar{c}_1 \vee \ldots \vee \bar{x} = \bar{c}_n$ where $n \geq 0$.*
2. *e is co-relational if it is of the form $\bar{x} \neq \bar{c}_1 \wedge \ldots \wedge \bar{x} \neq \bar{c}_n$ where $n > 0$.*
3. *e is unrestricted otherwise.*

Definition 4.4 *The e-size of Σ is the maximum size of an unrestricted e_P or $e_{\neg P}$ in Σ.*

For each predicate P, we store P^+ as a sorted relation when e_P is relational or co-relational, and analogously for P^- using $e_{\neg P}$. We let *DB*(Σ, C) (where C is a set of constants that includes all those in Σ) denote the database whose domain is C and whose relations are P^+ and P^- as above.

Note that in the case of a predicate P in Σ corresponding to a normal database-style relation, e_P will be relational and $e_{\neg P}$ will be co-relational and the two relations P^+ and P^- will be identical. A simple optimization would merge them.

Next, given a formula α we construct a database query by replacing every occurrence of a predicate P as follows: a positive[3] occurrence of P is replaced by P^+ when e_P is relational, $\neg P^+$ when co-relational, and by e_P itself otherwise; a negative occurrence of P is replaced by $\neg P^-$ when $e_{\neg P}$ is relational, P^- when co-relational, and by $\neg e_{\neg P}$ itself otherwise. We let $Q(\Sigma, \alpha)$ denote the resulting database query with double negation removed.

We present examples of this transformation in the next section. Here we note the following correctness result:

[3]A positive (resp. negative) occurrence of P in α is one within the scope of an even (resp. odd) number of negations.

Theorem 4.5 *Let Σ be a proper KB and α a formula of $\mathcal{L}$. Let $C = H(\Sigma|\alpha, \alpha)$, $\mathcal{A} = DB(\Sigma|\alpha, C)$, and $\phi = Q(\Sigma, \alpha)$. Then $\alpha_f(\Sigma) = \phi(\mathcal{A})$.*

This theorem says that the (finite version of) answers to open queries in $\mathcal{L}$ correspond exactly to the answers we get for the database constructed as above. This is a consequence of locality and the following lemma:

Lemma 4.6 *Let Σ be a proper KB and α a sentence of $\mathcal{L}$. Let C be any finite set of constants consisting of those in $\Sigma \cup \{\alpha\}$ and k extra ones for some $k \geq qr(\alpha)$. Let $\mathcal{A} = DB(\Sigma|\alpha, C)$. Then $V[\Sigma, \alpha] = 1$ iff $\mathcal{A} \models Q(\Sigma, \alpha)$, and $V[\Sigma, \alpha] = 0$ iff $\mathcal{A} \models Q(\Sigma, \neg\alpha)$.*

Proof: The proof is by induction on the structure of α. Here we choose to prove the case for $V[\Sigma, \alpha] = 0$ which is trickier than the case for $V[\Sigma, \alpha] = 1$.

$V[\Sigma, P(\bar{c})] = 0$ iff $\mathcal{E} \models e_{\neg P}(\bar{c})$ iff $\mathcal{A} \models Q(\Sigma, \neg P(\bar{c}))$ since

$$\mathcal{E} \models e_{\neg P}(\bar{c}) \text{ iff } \begin{cases} \mathcal{A} \models P^-(\bar{c}) & \text{if } e_{\neg P} \text{ is relational} \\ \mathcal{A} \models \neg P^-(\bar{c}) & \text{if } e_{\neg P} \text{ is co-relational} \\ \mathcal{A} \models e_{\neg P}(\bar{c}) & \text{otherwise} \end{cases}$$

The case for $t = t'$ is trivial. $V[\Sigma, \neg\alpha] = 0$ iff $V[\Sigma, \alpha] = 1$ iff (by induction) $\mathcal{A} \models Q(\Sigma, \alpha)$, *i.e.* $Q(\Sigma, \neg\neg\alpha)$.

Consider $\alpha \vee \beta$. Let $Q(\Sigma, \neg\alpha) = \neg\phi$ and $Q(\Sigma, \neg\beta) = \neg\psi$. Then $Q(\Sigma, \neg(\alpha \vee \beta)) = \neg(\phi \vee \psi)$. $V[\Sigma, \alpha \vee \beta] = 0$ iff $V[\Sigma, \alpha] = 0$ and $V[\Sigma, \beta] = 0$ iff (by induction) $\mathcal{A} \models Q(\Sigma, \neg\alpha)$ and $\mathcal{A} \models Q(\Sigma, \neg\beta)$ iff $\mathcal{A} \models \neg\phi \wedge \neg\psi$ iff $\mathcal{A} \models \neg(\phi \vee \psi)$, *i.e.* $Q(\Sigma, \neg(\alpha \vee \beta))$.

Consider $\exists x\alpha$. Let $Q(\Sigma, \neg\alpha) = \neg\phi$. Then $Q(\Sigma, \neg\exists x\alpha) = \neg\exists x\phi$. $V[\Sigma, \exists x\alpha] = 0$ iff $V[\Sigma, \alpha^x_d] = 0$ for any $d \in H^+(\Sigma \cup \{\alpha\})$ iff (by Corollary 2.4 since C contains at least one constant not in $\Sigma \cup \{\alpha\}$) $V[\Sigma, \alpha^x_d] = 0$ for any $d \in C$ iff (by induction since C consists of those constants in $\Sigma \cup \{\alpha^x_d\}$ and j extra ones for some $j \geq qr(\alpha)$) $\mathcal{A} \models Q(\Sigma, \neg\alpha^x_d)$, *i.e.* $\neg\phi^x_d$ for any $d \in C$ iff $\mathcal{A} \models \forall x\neg\phi$ iff $\mathcal{A} \models \neg\exists x\phi$, *i.e.* $Q(\Sigma, \neg\exists x\alpha)$. ■

To obtain a tractability result for open queries over proper KBs, we need only ensure that the database queries we construct are k-guarded then use the Eval procedure.

Definition 4.7 *Let Σ be a proper KB. The strictly k-guarded fragment $SG_k(\Sigma)$ of $\mathcal{L}$ for Σ is the same as SGF_k except that when $\exists\bar{x}(G_1 \wedge \ldots \wedge G_m \wedge \phi)$ is within the scope of an even (resp. odd) number of $\neg$s, the G_i must be equalities or atoms $P(\bar{t})$ s.t. e_P is relational (resp. $e_{\neg P}$ is co-relational) or literals $\neg P(\bar{t})$ s.t. $e_{\neg P}$ is relational (resp. e_P is co-relational).*

Clearly, for any α, if $\alpha \in \mathrm{SG}_k(\Sigma)$, then $Q(\Sigma, \alpha) \in \mathrm{SGF}_k$. Note that every formula α with width m can be transformed into an equivalent $\mathrm{SG}_k(\Sigma)$-formula for some $k \leq m$, using equalities as guards if necessary.

Now we get our main complexity result:

Theorem 4.8 *Let Σ be a proper KB and $\alpha \in SG_k(\Sigma)$. Then $\alpha_f(\Sigma)$ can be computed in time $O(lwn^k)$, where l is the size of α, n is the size of $\Sigma|\alpha$, and w is the e-size of $\Sigma|\alpha$.*

Proof: Let $C = H(\Sigma|\alpha, \alpha)$, $\mathcal{A} = DB(\Sigma|\alpha, C)$, and $\phi = Q(\Sigma, \alpha)$. We sort C. By Theorem 4.5, $\alpha_f(\Sigma) = \mathrm{Eval}(\mathcal{A}, \phi)$. The size of $\mathcal{A}$ is $O(n)$, and the length of ϕ is $O(lw)$. By Theorem 3.3, $\alpha_f(\Sigma)$ can be computed in time $O(lwn^k)$. ■

As a special case of the theorem, we have:

Corollary 4.9 *Let Σ be a proper KB and α a sentence in $SG_k(\Sigma)$. Then whether $V[\Sigma, \alpha] = 1$ can be decided in time $O(lwn^k)$, where l, w and n are as above.*

Corollary 4.10 *Let Σ be a proper KB and $\alpha \in \mathcal{NF} \cap SG_k(\Sigma)$. Then whether $\mathcal{E} \cup \Sigma \models \alpha$ can be decided in time $O(lwn^k)$, where l, w and n are as above.*

Proof: Follows from previous corollary and the fact that V is logically sound and complete for queries in $\mathcal{NF}$. ■

Note that these bounds are identical to their database counterparts modulo the w factor. In most cases of interest, the w will be small since even in a large KB, we only expect to see a large e_P or $e_{\neg P}$ in database-like cases, where the e is relational or co-relational. In the case where none of the e are relational or co-relational, we get the following:

Corollary 4.11 *Let Σ be a proper KB and α a formula of width k with all literals defined by an unrestricted ewff. Then $\alpha_f(\Sigma)$ can be evaluated in time $O((lw)^{k+1})$, where l is the size of α, and w is the e-size of $\Sigma|\alpha$.*

Proof: We transform α into an equivalent $\mathrm{SG}_k(\Sigma)$-formula, using equalities as guards. The size of $\Sigma|\alpha$ is $O(lw)$. ■

5 An example

To illustrate how query evaluation would work with proper KBs, we return to the example robotic scenario mentioned in the first section, involving the predicate $In(person, room)$. We can see from the example that e_{In} is given in relational form, while $e_{\neg In}$ is given in unrestricted form. To make things interesting, we assume two more predicates: One is $Mgr(person_1, person_2)$ saying that the first person is a manager and that the second is one of his or her employees; we assume that *Mgr* is given as a closed database predicate. The other is $Cmp(person_1, person_2)$ saying that the two persons are compatible. We assume that the robot knows that any two people are compatible except $m(> 2)$ pairs; among these m pairs, the robot only knows that two pairs are not compatible. Hence e_{Cmp} is given in co-relational form, and $e_{\neg Cmp}$ is given in relational form. Thus the corresponding database has the following relations: In^+, Mgr^+, Mgr^-, Cmp^+, and Cmp^-, where $Mgr^+ = Mgr^-$, and $Cmp^- \subseteq Cmp^+$.

Here are some example queries,with the guards underlined.

1. What rooms are unoccupied?
   $\underline{r = r} \wedge \neg\exists x(\underline{x = x \wedge r = r} \wedge In(x, r))$
2. Are there rooms with more than one manager in them?
   $\exists r \exists m \exists m'(\underline{In(m, r) \wedge In(m', r)} \wedge m \neq m' \wedge \exists x \underline{Mgr(m, x)} \wedge \exists x' \underline{Mgr(m', x')})$
3. Are there any managers whose employees are all in the same room?
   $\exists m \exists r(\underline{m = m \wedge r = r} \wedge \neg\exists x(\underline{Mgr(m, x) \wedge r = r} \wedge \neg In(x, r)))$
4. Is it true that any two people who are not compatible are in different rooms?
   $\neg\exists x \exists x'(\underline{\neg Cmp(x, x')} \wedge \neg\exists r \exists r'(\underline{In(x, r) \wedge In(x, r')} \wedge r \neq r'))$

Observe that in all cases, the queries are strictly 2-guarded wrt the proper KB. Also, because no query contains a literal and a unifiable literal of opposite polarity, from results in [Levesque, 1998], all the queries are in $\mathcal{NF}$.

The corresponding database queries are as follows:

1. $\underline{r = r} \wedge \neg\exists x(\underline{x = x \wedge r = r} \wedge \neg e_{\neg \mathit{In}}(x, r))$
2. $\exists r \exists m \exists m'(\underline{\mathit{In}^+(m, r) \wedge \mathit{In}^+(m', r)} \wedge m \neq m' \wedge \exists x \underline{\mathit{Mgr}^+(m, x)} \wedge \exists x' \underline{\mathit{Mgr}^+(m', x')})$
3. $\exists m \exists r(\underline{m = m \wedge r = r} \wedge \neg\exists x(\underline{\mathit{Mgr}^-(m, x) \wedge r = r} \wedge \neg \mathit{In}^+(x, r)))$
4. $\neg\exists x \exists x'(\underline{\mathit{Cmp}^+(x, x')} \wedge \neg\exists r \exists r'(\underline{\mathit{In}^+(x, r) \wedge \mathit{In}^+(x, r')} \wedge r \neq r'))$

6 Conclusions

In this paper, we have shown how a bottom-up query evaluation procedure for databases can be used to answer queries for KBs with a certain form of incomplete knowledge. Although this procedure can be impractical for k-guarded queries where k is large, they would be impractical for databases too.

A number of questions remain to be addressed. First of all, Lakemeyer and Levesque [2002] have proposed an extension to proper KBs that allow disjunctions in the KB. It would be interesting to see how much of the database retrieval mechanism could be preserved in this case. We can also imagine other extensions to proper KBs, such as relaxing the unique name assumption over constants, or allowing a limited use of function symbols. Also, as suggested in the first section, we can imagine a dynamic scenario where at any given point what a robot or agent knows about the current situation is expressible as a proper KB. It would then be useful to amalgamate regression-based techniques for reasoning about change from [Reiter, 2001] with the database techniques considered here. Among other things, this would require determining those cases where the successor state axioms guarantee that a proper KB remains proper after an action has been performed, perhaps along the lines of [Petrick and Levesque, 2002]. It would also be interesting to investigate the relationship between proper KBs and other subsets of logic to see if the complexity results presented here can be further generalized. Two immediate candidates are datalog programs and stratified logic programs that include some form of classical negation. Finally, we note that additional optimizations can be made to our query evaluation procedure that do not change the worst-case performance, but would improve its behaviour in practice.

Acknowledgments

We would like to thank Leonid Libkin for pointing us to the relevant database literature. Financial support was gratefully received from the Natural Sciences and Engineering Research Council of Canada.

References

[Andréka *et al.*, 1996] H. Andréka, I. Hodkinson, and I. Németi. Modal languages and bounded fragments of predicate logic. ILLC Research Report ML-96-03, University of Amsterdam, 1996.

[Chandra and Merlin, 1977] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Conference record of the 9th annual ACM Symp. on Theory of Computing*, pages 77–90, 1977.

[Chekuri and Rajaraman, 1997] C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. In *Proc. 6th Int. Conf. on Database Theory*, pages 56–70, 1997.

[Downey and Fellows, 1995] R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal on Computing*, 24(4):873–921, 1995.

[Flum *et al.*, 2001] J. Flum, M. Frick, and M. Grohe. Query evaluation via tree-decompositions. In *Proc. of the 8th Int. Conf. on Database Theory* , pages 22–38, 2001.

[Gottlob *et al.*, 1999] G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. In *Proc. 18th ACM Symp. on Principles of Database Systems*, 1999.

[Gottlob *et al.*, 2001] G. Gottlob, N. Leone, and F. Scarcello. Robbers, marshals, and guards: Game theoretic and logical characterizations of hypertree width. In *Proc. of the 20th ACM Symp. on Principles of Database Systems*, 2001.

[Grohe, 2002] M. Grohe. Parameterized complexity for the database theorist. *SIGMOD Record*, 31(4), 2002.

[Lakemeyer and Levesque, 2002] G. Lakemeyer and H.J. Levesque. Evaluation-based reasoning with disjunctive information in first-order knowledge bases. In *Proc. of KR-02*, pages 73–81, 2002.

[Levesque, 1998] H.J. Levesque. A completeness result for reasoning with incomplete first-order knowledge bases. In *Proc. of KR-98*, pages 14–23, 1998.

[Papadimitriou and Yannakakis, 1999] C.H. Papadimitriou and M. Yannakakis. On the complexity of database queries. *Journal of Computer and System Sciences*, 58(3):407–427, 1999.

[Petrick and Levesque, 2002] R. Petrick and H.J. Levesque. Knowledge equivalence in combined action theories. In *Proc. of KR-02*, pages 303–314, 2002.

[Reiter, 2001] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, 2001.

[Vardi, 1982] M.Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th Annual ACM Symp. on Theory of Computing*, pages 137–146, 1982.

[Yannakakis, 1981] M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of the 7th Int. Conf. on Very Large Data Bases*, pages 82–94, 1981.

[Yannakakis, 1995] M. Yannakakis. Perspectives on database theory. In *Proc. of the 36th Annual Symp. on Foundations of Computer Science*, pages 224–246, 1995.