Efficient Reasoning in Proper Knowledge Bases with Unknown Individuals

Giuseppe De Giacomo
Sapienza Università di Roma
Rome, Italy
degiacomo@dis.uniroma1.it

Yves Lespérance
York University
Toronto, Canada
lesperan@cse.yorku.ca

Hector J. Levesque
University of Toronto
Toronto, Canada
hector@cs.toronto.edu

Abstract

This work develops an approach to efficient reasoning in first-order knowledge bases with incomplete information. We build on Levesque's proper knowledge bases approach, which supports limited incomplete knowledge in the form of a possibly infinite set of positive or negative ground facts. We propose a generalization which allows these facts to involve unknown individuals, as in the work on labeled null values in databases. Dealing with such unknown individuals has been shown to be a key feature in the database literature on data integration and data exchange. In this way, we obtain one of the most expressive first-order open-world settings for which reasoning can still be done efficiently by evaluation, as in relational databases. We show the soundness of the reasoning procedure and its completeness for queries in a certain normal form.

1 Introduction

As argued by Levesque [1998], there is to date really only one logically correct deductive reasoning technique that appears to be efficient enough to be feasible for large knowledge bases (KBs) containing (say) millions of facts: *database retrieval*. (SAT solving does show promise too, but not for deduction and not for first-order reasoning.) Viewed in logical terms, however, a database requires complete information: every atomic formula must be known to be true or known to be false. In particular, databases (typically) make a *closed-world* assumption: anything not explicitly recorded as true can be assumed to be false. Yet many AI applications must survive in an *open-world* setting, with incomplete information of their domain, often acquired incrementally over time (e.g. a robot that must sense its environment to determine if there are boxes nearby, or a travel-agent softbot that gathers information about available flights over the web).

Levesque [1998] raises the question: for what sorts of logical theories can we retain the deductive efficiency of database retrieval without requiring complete knowledge? To answer it, he proposes the notion of a *proper knowledge base*, where incomplete knowledge amounts to a possibly infinite set of positive or negative ground facts. For this kind of KB he devises a reasoning procedure based on *formula evaluation*, which he shows is logically sound and sometimes also complete. Most interestingly, the procedure is essentially as efficient as query evaluation in databases with complete information [Liu and Levesque, 2003; Liu, 2006]. However, one serious limiting element of this proposal is that a proper KB cannot include knowledge about individuals whose identity is unknown.

Incomplete information has also been studied in databases [Imielinski and Lipski, 1984; Reiter, 1986; Vardi, 1986; van der Meyden, 1998], and recently it is acquiring new importance in the context of uncertain databases, connected with probabilistic approaches [Agrawal *et al.*, 2010; Antova *et al.*, 2009], XML semistructured data [Abiteboul *et al.*, 2006; Barceló *et al.*, 2010], and most prominently in data integration and data exchange [Lenzerini, 2002; Kolaitis, 2005]. Interestingly, in much of this work, incomplete information comes in the form of unknown individuals (or *labeled null values*), i.e., we know that there exists an object with certain properties, but we cannot really identify it (though we may still use it in joins, and more generally, in formulas). Incomplete information of this form is tightly linked with the problem of query containment, and more specifically with query containment involving conjunctive queries [Chandra and Merlin, 1977; Ullman, 1997; Lausen and Wei, 2003]. Indeed Chandra and Merlin showed that a conjunctive query can be seen as a database with labeled nulls and conversely, that a database with labeled nulls can be seen as a conjunctive query. This observation has led to a rich theory which is at the base of modern approaches to data integration and data exchange, including the work on ontology-based data access and integration proposed within the AI community [Poggi *et al.*, 2008].

In this paper we combine ideas from these two lines of research. We obtain a generalization of Levesque's proper KBs that handles both missing facts (like the original version) and information about unknown individuals (as in the database literature). In this way, we obtain one of the most expressive open-world settings for which reasoning can still be done by evaluation as in closed-world databases. Specifically we devise a query answering procedure and show that it is always logically sound and is also logically complete when the query is in a certain normal form. We analyze its computational complexity, showing that as long as the number of unknown individuals remains small (logarithmic in the size of the KB), the reasoning can indeed be implemented efficiently.

The rest of the paper is organized as follows. In Section 2, we review databases as knowledge bases with a built-in closed-world assumption, and proper KBs as open-world knowledge bases. In Section 3, we show how to handle unknown individuals (null values) in proper KBs, while retaining the same sort of efficiency and soundness and completeness properties. Finally in Section 4, we draw some conclusion and discuss future work.

2 Proper Knowledge Bases

We use an ordinary first-order logical language $\mathcal{L}$ with an infinite supply of predicate symbols (including $=$), an infinite supply of constants we call *standard names* (which we write as $^{\#}1, ^{\#}2, ^{\#}3, \ldots$), and no other function or constant symbols. We use the notation α^x_n to mean the result of replacing every free occurrence of variable x in formula α by standard name n. In this paper, we will be using the normal Tarski semantics for $\mathcal{L}$, with $\models$ understood as normal logical entailment.

However, we wish to make the *unique name assumption* for the standard names. What this means is that for any knowledge base KB $\subseteq \mathcal{L}$ and any query $\alpha \in \mathcal{L}$, instead of asking if KB $\models \alpha$, we will always be asking if $\mathcal{E} \cup \text{KB} \models \alpha$ where $\mathcal{E}$ is the usual axioms of equality (reflexitivity, symmetry, transitivity, and substitution of equals for equals) together with

$$\{\, \neg(n = n') \mid n \text{ and } n' \text{ are distinct standard names} \,\}.$$

Using $\mathcal{E}$ in this way has a number of interesting properties. In particular, it can be shown that whenever the KB is finite (as it will always be here), the set $(\mathcal{E} \cup \text{KB})$ has a model iff it has a *standard* model, that is, one where where $=$ is interpreted as identity and the domain is isomorphic to the set of standard names (see [Levesque, 1998], Theorem 2). So this gives us what amounts to an infinitary version of a *domain closure assumption*. Another property we will use is this:

Theorem 1: *[Levesque, 1998] Suppose that* KB *is finite and H contains all the standard names that appear in* KB *or α and at least one other standard name. Then*

$$\mathcal{E} \cup \text{KB} \models \forall x\alpha \;\; \textit{iff} \;\; \mathcal{E} \cup \text{KB} \models \alpha^x_n \; \textit{for every } n \in H.$$

This means that we can determine whether $\forall x\alpha$ is entailed by checking whether a finite set of instances of α are entailed.

Now let us consider databases under the closed-world assumption. Following Reiter [Reiter, 1986], we can think of a database as a logical theory formed by a finite set of ground atoms A, those explicitly in the database, augmented with a set of negated atoms obtained by applying the closed-world assumption, i.e., we can take the KB to be $A \cup \{\neg P(\vec{n}) \mid \vec{n}$ is a vector of standard names and $P(\vec{n}) \notin A\}$. Then for any query α, we can decide if $\mathcal{E} \cup \text{KB} \models \alpha$ using the following procedure:

1. $V[\text{KB}, P(\vec{n})] = \begin{cases} 1 & \text{if } P(\vec{n}) \in A \\ 0 & \text{otherwise} \end{cases}$
2. $V[\text{KB}, (n = n')] =$
$\begin{cases} 1 & \text{if } n \text{ and } n' \text{ are the same standard name} \\ 0 & \text{otherwise} \end{cases}$
3. $V[\text{KB}, \neg\alpha] \;=\; 1 - V[\text{KB}, \alpha]$
4. $V[\text{KB}, (\alpha \vee \beta)] \;=\; \max\{V[\text{KB}, \alpha],\, V[\text{KB}, \beta]\}$
5. $V[\text{KB}, \exists x\,\alpha] \;=\; \max_{n \in H} V[\text{KB}, \alpha^x_n]$

(We can treat $(\alpha \wedge \beta)$, $(\alpha \supset \beta)$, $(\alpha \equiv \beta)$ and $\forall x\,\alpha$ as abbreviations.) V simply evaluates the truth of α in the model corresponding to $\mathcal{E} \cup \text{KB}$ in the obvious way.

Proper KBs generalize databases to support open-world reasoning, where we do not require every formula or its negation to be known [Levesque, 1998]. Specifically a *proper knowledge base* KB is a finite collection of sentences of $\mathcal{L}$ of the form $\forall\vec{x}\,(e \supset \rho)$, where

- e is an *ewff*, i.e. a quantifier-free formula whose only predicate is equality, and
- ρ is $P(\vec{x})$ or $\neg P(\vec{x})$

and such that $\mathcal{E} \cup \text{KB}$ is consistent.

It can be shown that a proper KB is a finite representation for a possibly infinite consistent set of ground literals:

$$\{\, \rho\theta \mid \mathcal{E} \cup \text{KB} \models \rho\theta \,\}.$$

In this paper, θ is a substitution of free variables by standard names, and $\alpha\theta$ denotes α after the substitution.

Proper KBs have many interesting properties. Unlike databases, we can leave the status of some literals *open*.

Example 1: Let KB stand for : $\{\, \forall x(x = {}^{\#}1 \supset P(x))$, $\forall x(x \neq {}^{\#}1 \wedge x \neq {}^{\#}2 \supset \neg P(x)) \,\}$. Then

$$\mathcal{E} \cup \text{KB} \models P(^{\#}1),\ \neg P(^{\#}3),\ \neg P(^{\#}4),\ \neg P(^{\#}5),\ \ldots$$

but $\mathcal{E} \cup \text{KB} \not\models P(^{\#}2)$ and $\mathcal{E} \cup \text{KB} \not\models \neg P(^{\#}2)$.

Also, we can make the closed-world assumption for *some* of the predicates by including both $\forall(e \supset \rho)$ and $\forall(\neg e \supset \overline{\rho})$, for some e and ρ.

Example 2: Let KB stand for

$$\{\forall x\,(x = {}^{\#}2 \vee x = {}^{\#}3 \vee x = {}^{\#}5 \vee x = {}^{\#}9 \supset P(x)),$$
$$\forall x\,(x \neq {}^{\#}2 \wedge x \neq {}^{\#}3 \wedge x \neq {}^{\#}5 \wedge x \neq {}^{\#}9 \supset \neg P(x))\}$$

This amounts to making the closed-world assumption on P.

Any *finite* set of ground literals can be reformulated as a proper KB, by simply rewriting any ground literal $\rho\theta$ in the set as $\forall\vec{x}\,(\vec{x} = \vec{x}\theta \;\supset\; \rho)$. Some *infinite* sets of literals can also be represented, *e.g.*, $\forall x, y, z(x \neq z \wedge y = {}^{\#}3 \supset R(x, y, z))$.

Reasoning with proper KBs. Although proper KBs are quite restricted, it remains *undecidable* to determine whether or not $\mathcal{E} \cup \text{KB} \models \alpha$. (Consider the case KB $= \{\,\}$, where we still need to determine whether an arbitrary first order formula is valid.) As an alternative, Levesque [1998] proposes a *limited reasoning procedure* $V[\text{KB}, \alpha]$, based on the evaluation function used for databases under the closed-world assumption, that returns 1 (known to be true), 0 (known to be false), or $\frac{1}{2}$ (unknown), defined as follows:

1. $V[\text{KB}, \rho\theta] =$
$\begin{cases} 1 & \text{if there is a } \forall(e \supset \rho) \in \text{KB s.t. } \mathcal{E} \models e\theta \\ 0 & \text{if there is a } \forall(e \supset \overline{\rho}) \in \text{KB s.t. } \mathcal{E} \models e\theta \\ \frac{1}{2} & \text{otherwise} \end{cases}$

2. $V[\mathrm{KB}, (n = n')] = \begin{cases} 1 & \text{if } n \text{ and } n' \text{ are the same standard name} \\ 0 & \text{otherwise} \end{cases}$
3. $V[\mathrm{KB}, \neg\alpha] = 1 - V[\mathrm{KB}, \alpha]$
4. $V[\mathrm{KB}, (\alpha \vee \beta)] = \max\{V[\mathrm{KB}, \alpha], V[\mathrm{KB}, \beta]\}$
5. $V[\mathrm{KB}, \exists x\, \alpha] = \max_{n \in H} V[\mathrm{KB}, \alpha^x_n]$

The procedure V has been shown to have several interesting properties. First of all V *is tractable* in a very strong sense.

1. If e is a ground ewff, then $\mathcal{E} \models e$ iff $V[\{\}, e] = 1$ and can be determined in time linear in $|e|$.
2. $V[\mathrm{KB}, \rho\theta]$ can be determined in time linear in $|\mathrm{KB}|$: scan KB for $\forall(e \supset \rho)$ or $\forall(e \supset \overline{\rho})$ and check if $\mathcal{E} \models e\theta$.
3. Overall, computing $V[\mathrm{KB}, \alpha]$ can be made as efficient as database retrieval [Liu and Levesque, 2003].

Also V *is logically sound*:

Theorem 2: *[Levesque, 1998]*

- *If* $V[\mathrm{KB}, \alpha] = 1$, *then* $\mathcal{E} \cup \mathrm{KB} \models \alpha$.
- *If* $V[\mathrm{KB}, \alpha] = 0$, *then* $\mathcal{E} \cup \mathrm{KB} \models \neg\alpha$.

However V *is not (and cannot be) logically complete* in general. For example:

$$\mathcal{E} \cup \{\} \models (p \vee \neg p), \text{ but } V[\{\}, (p \vee \neg p)] = \frac{1}{2}.$$

Completeness for queries in the normal form $\mathcal{NF}$. To get completeness one has to consider suitable sublanguages of $\mathcal{L}$. One interesting sublanguage is $\mathcal{NF}$ [Levesque, 1998].

First, let us say that a set of sentences S is *logically separable* iff for every consistent set of literals L, if $L \cup S$ has no standard interpretation, then for some $\alpha \in S$, $L \cup \{\alpha\}$ has no standard interpretation. Then the normal form $\mathcal{NF} \subseteq \mathcal{L}$ is defined as the least set such that:

1. If α is a ground atom or ewff, then $\alpha \in \mathcal{NF}$.
2. If $\alpha \in \mathcal{NF}$ then $\neg\alpha \in \mathcal{NF}$.
3. If $S \subseteq \mathcal{NF}$, S is logically separable, and S is finite, then $\bigwedge S \in \mathcal{NF}$.
4. If $S \subseteq \mathcal{NF}$ S is logically separable, and $S = \{\alpha^x_n \mid n \text{ is a standard name}\}$, then $\forall x\, \alpha \in \mathcal{NF}$.

V is complete (as well as sound) for queries in $\mathcal{NF}$:

Theorem 3: *[Levesque, 1998] For any proper* KB *and any* $\alpha \in \mathcal{NF}$ *we have:*

- *If* $\mathcal{E} \cup \mathrm{KB} \models \alpha$, *then* $V[\mathrm{KB}, \alpha] = 1$.
- *If* $\mathcal{E} \cup \mathrm{KB} \models \neg\alpha$, *then* $V[\mathrm{KB}, \alpha] = 0$.

What sorts of queries can we express in $\mathcal{NF}$? In the propositional case, for any quantifier-free formula α, there is an equivalent formula α' that is in $\mathcal{NF}$. [Liu and Lakemeyer, 2008] have shown that this does not hold for arbitrary formulas with quantifiers. A counterexample is $\forall x\, y\, z\, (R(x, y) \wedge R(y, z) \supset R(x, z))$. However, some significant classes of $\mathcal{L}$ queries belong to $\mathcal{NF}$.

Let's say that two literals are *conflict-free* iff either they have the same polarity, or they use different predicates, or they use different standard names at some argument position.

Theorem 4: *[Levesque, 1998] If all pairs of literals in* α *are conflict-free, then* $\alpha \in \mathcal{NF}$.

Notice that this theorem applies for example to *conjunctive queries* from databases.

3 Dealing with Unknown Individuals

Proper KBs can represent some open-world knowledge, but only for individuals whose identity is *known*. There is no way for a proper KB to say that something has a certain property, without saying what the standard name of that thing is. In fact, for $\alpha \in \mathcal{NF}$ we have the following: If $\mathcal{E} \cup \mathrm{KB} \models \exists x \alpha$, then for some standard name n, $\mathcal{E} \cup \mathrm{KB} \models \alpha^x_n$. This is (in part) what allows the V procedure to work correctly.

In this section, we consider how to deal with open-world knowledge that involves individuals whose identity is not known. We start by extending our first-order logical language $\mathcal{L}$, so that in addition to the standard names, we include a disjoint countably infinite supply of constants called (labeled) *null values* (which we write $a_1, a_2, \ldots$). So now there are two distinct types of constants in $\mathcal{L}$. As constants, null values behave exactly like standard names, except that they are not mentioned by name in $\mathcal{E}$ and so the unique name assumption does not apply to them. From now on, let us assume that a proper KB is a a set of sentences as defined earlier, but using this richer language $\mathcal{L}$ (i.e., the ewffs in its assertions may contain null values). For example, if our KB is

$$\{\forall x.(x = {}^{\#}3 \supset P(x)),\ \forall x.(x = a \supset Q(x))\},$$

then we know the individual that is P (that is, $^{\#}3$), but we do not know which individual is Q; all we know is that *something* (which may or may not be $^{\#}3$) has property Q. In this sense, null values behave just like *Skolem constants*.

Reasoning with null values. Null values in proper KBs pose a problem for reasoning, because, unlike the original proper KBs, they allow certain types of *disjunctive knowledge* to be expressed. Consider the following examples:

Example 3: Let KB be $\{\forall x(x \neq {}^{\#}1 \wedge x \neq {}^{\#}2 \supset \neg P(x)),$ $\forall x(x = a \supset P(x))\}$. Then $\mathcal{E} \cup \mathrm{KB} \models (a = {}^{\#}1 \vee a = {}^{\#}2)$.

Example 4: Let KB be $\{\forall x(x = a \supset P(x)),$ $\forall x(x \neq a \supset Q(x))\}$. Then $\mathcal{E} \cup \mathrm{KB} \models (P({}^{\#}5) \vee Q({}^{\#}5))$.

Example 5: Let KB be $\{\forall x(x = a \wedge x = a' \supset P(x))\}$. Then $\mathcal{E} \cup \mathrm{KB} \models (a \neq a' \vee \exists x P(x))$.

Obviously, the V procedure, which handles $V[\mathrm{KB}, (\alpha \vee \beta)]$ by taking the maximum of $V[\mathrm{KB}, \alpha]$ and $V[\mathrm{KB}, \beta]$, will no longer work. Similar problems arise with existentials.

To see how we can avoid these problems, let us suppose from now on that the KB uses k null values $\vec{a} = \langle a_1, \ldots a_k\rangle$, and for simplicity, that any null value that appears in a query α also appears in KB.[1] We can in fact eliminate the null values by repeatedly replacing them by standard names:

[1] Dropping this assumption would complicate the technical treatment to follow, though all results still hold. In effect, a null value that appears *only* in a query can be replaced by a quantified variable.

Theorem 5: *Let $\vec{H}$ be a set of tuples of standard names $\langle n_1, \ldots, n_k \rangle$ such that n_1 ranges over the standard names in* KB *and α plus one more, and for each $1 \leq i < k$, n_{i+1} ranges over the names that n_i ranges over plus one more. Then*

$$\mathcal{E} \cup \mathrm{KB} \models \alpha$$
$$\textit{iff}$$
$$\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}} \models \alpha^{\vec{a}}_{\vec{n}} \ \textit{ for all names } \vec{n} \in \vec{H}$$

The proof of this uses Theorem 1, the Deduction Theorem, and the Skolemization Theorem (that a set of sentences containing an existential sentence is satisfiable iff the set with the existential variable replaced by a new constant is satisfiable). Thus, with a *caveat to follow*, we are indeed able to reduce the evaluation of a query $V[\mathrm{KB}, \alpha]$ possibly involving null values to a finite set of query evaluations $V[\mathrm{KB}^{\vec{a}}_{\vec{n}}, \alpha^{\vec{a}}_{\vec{n}}]$ that do not involve any null values. Note however that these involve looking at different instances of the KB.

The *caveat* is the following. Before we can use V (as previously defined) over $\mathrm{KB}^{\vec{a}}_{\vec{n}}$ and $\alpha^{\vec{a}}_{\vec{n}}$, we must ensure that $\mathrm{KB}^{\vec{a}}_{\vec{n}}$ (that is, the KB with its null values $\vec{a}$ replaced by the standard names $\vec{n}$) is a *proper* KB. $\mathrm{KB}^{\vec{a}}_{\vec{n}}$ will certainly have the right form, i.e., a finite set of $\forall(e \supset \rho)$ not containing null values. However, we must also ensure that $\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}}$ is consistent! (When it is not, $\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}}$ would then entail every query.)

Checking whether $\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}}$ is consistent is non-trivial. Indeed, even though $\mathcal{E} \cup \mathrm{KB}$ may be consistent, there can still be values of $\vec{n}$ for which $\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}}$ is not consistent.

Example 6: Let $\mathrm{KB} = \{\, \forall x(x \neq {}^{\#}1 \wedge x \neq {}^{\#}2 \supset \neg P(x)),$ $\forall x(x = a \supset P(x)) \,\}$. Then $\mathcal{E} \cup \mathrm{KB}^{a}_{{}^{\#}5}$ is inconsistent.

One possible solution is to restrict the original KB so that this discrepancy between the consistency of the original KB and that of the instantiated KBs never occurs. For instance, we could insist that if a null value occurs in a $\forall(e \supset \rho)$ in the KB, then the KB must not contain a sentence for $\overline{\rho}$.

Here, we focus on a more robust approach: detect when $\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}}$ is inconsistent and make sure that we return 1 for any query in this case. As it turns out, this can be done by using V.

Theorem 6: *Suppose* KB *is in the proper form and contains no null values. Then $\mathcal{E} \cup \mathrm{KB}$ is inconsistent iff there is a predicate P in the* KB *such that*

$$V[\{\,\},\, \exists \vec{x}\, (e^T_P \wedge e^F_P)] \;=\; 1$$

where e^T_P is the disjunction of all e s.t. $\forall(e \supset P(\vec{x})) \in \mathrm{KB}$, and e^F_P is the disjunction of all e s.t. $\forall(e \supset \neg P(\vec{x})) \in \mathrm{KB}$.

Using this result, we can immediately define a query-answering procedure for proper KBs with null values. However, the resulting procedure would not be very good. While the V can be executed efficiently, we would need to go over the *entire* KB to look for an inconsistency with some predicate P. Part of what made the query procedure presented in [Liu and Levesque, 2003] *practical* is that it only had to deal with the part of the KB involving predicates mentioned in the query. Furthermore, to use the above theorem, we would need to examine the entire KB after every substitution of null values by standard names from the KB and from the query.

Handling inconsistency offline. To make the query-answering procedure practical, we now show that it is possible to do all of the inconsistency checking in advance of seeing any queries. Our procedure works as follows:

- We construct a set of tuples $\vec{H_0}$ as in Theorem 5, but for the empty query.
- We construct a *table* T of those tuples from $\vec{H_0}$ such that $\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}}$ is inconsistent, applying Theorem 6.

After all this offline work, with the table T in hand, we are ready to handle any query α, as follows:

- We construct a set of tuples $\vec{H}$ as in Theorem 5, this time using the given α. (Some standard names may not have been considered in $\vec{H_0}$, as α may include new ones.)
- If some $\vec{n} \in \vec{H}$ *matches* (see below) an element of T, then we return 1. Otherwise, we return $V[\mathrm{KB}^{\vec{a}}_{\vec{n}}, \alpha^{\vec{a}}_{\vec{n}}]$.

A tuple of names $\langle n_1, \ldots, n_k \rangle$ *matches* another $\langle n^*_1, \ldots, n^*_k \rangle$ iff the following conditions hold:

1. $n_i = n_j$ iff $n^*_i = n^*_j$;
2. if n_i is in KB, then $n_i = n^*_i$;
3. if n_i is not in KB then n^*_i is not in KB.

We can compute this matching between two vectors of standard names using a unification-like procedure.

To see why such a procedure works, consider that when we constructed the table T, we considered the standard names in the KB and k new ones. But the actual new ones we used are not special in any sense, since the following theorem holds:

Theorem 7: *[Levesque, 1998] Suppose that β is a formula with one free variable x, and that n and m are two standard names that do not appear in β. Then $\mathcal{E} \models \beta^x_n$ iff $\mathcal{E} \models \beta^x_m$.*

Exploiting such a result, we can show the following:

Theorem 8: *For any standard names $\vec{n}$, $\mathcal{E} \cup \mathrm{KB}^{\vec{a}}_{\vec{n}}$ is inconsistent iff there is an $\vec{m} \in T$ such that $\vec{n}$ matches $\vec{m}$.*

So in the end, after computing the table T the final reasoning procedure for a proper KB that uses null values $\vec{a}$ is as follows: call a vector $\vec{n}$ of standard names *consistent* if it does not match any element of the table T; then we define:

$$V[\mathrm{KB}, \alpha] = \begin{cases} 1, \text{ if for all } \textit{consistent } \vec{n} \in \vec{H},\ V[\mathrm{KB}^{\vec{a}}_{\vec{n}}, \alpha^{\vec{a}}_{\vec{n}}] = 1 \\ 0, \text{ if for all } \textit{consistent } \vec{n} \in \vec{H},\ V[\mathrm{KB}^{\vec{a}}_{\vec{n}}, \alpha^{\vec{a}}_{\vec{n}}] = 0 \\ \frac{1}{2}, \text{ otherwise} \end{cases}$$

The fact that $\mathcal{E} \cup \mathrm{KB}$ is consistent guarantees that this V is well-defined. Indeed, it can be shown to be sound:

Theorem 9: *[Soundness] For any proper* KB *and query α, each of which may contain null values,*

- *if $V[\mathrm{KB}, \alpha] = 1$, then $\mathcal{E} \cup \mathrm{KB} \models \alpha$;*
- *if $V[\mathrm{KB}, \alpha] = 0$, then $\mathcal{E} \cup \mathrm{KB} \models \neg\alpha$.*

As before, the new V procedure is only complete for restricted queries. When a query α contains null values $\vec{a}$, we can let $\alpha \in \mathcal{NF}$ mean that $\alpha^{\vec{a}}_{\vec{n}} \in \mathcal{NF}$ (as previously defined) for every vector of standard names $\vec{n}$.[2] Then we get the following theorem:

[2] Other definitions of $\mathcal{NF}$ with null values are possible.

Theorem 10: *[Completeness] For any proper* KB *and query* $\alpha \in \mathcal{NF}$*, each of which may contain null values,*

- *if* $\mathcal{E} \cup \mathrm{KB} \models \alpha$*, then* $V[\mathrm{KB}, \alpha] = 1$*;*
- *if* $\mathcal{E} \cup \mathrm{KB} \models \neg\alpha$*, then* $V[\mathrm{KB}, \alpha] = 0$*.*

Let us now look at some examples.

Example 7: Returning to Example 4, we let the KB be $\{\, \forall x(x = a \supset P(x)), \forall x(x \neq a \supset Q(x)) \,\}$. First let the query α be $(P(^{\#}5) \vee Q(^{\#}5))$. (Thus $\mathcal{E} \cup \mathrm{KB} \models \alpha$.) Then $V[\mathrm{KB}, \alpha]$ will consider two names, $^{\#}5$ and say $^{\#}7$.

- For $n = {}^{\#}5$, $V[\mathrm{KB}^a_n, P(^{\#}5)] = 1$. So $V[\mathrm{KB}^a_n, \alpha] = 1$.
- For $n = {}^{\#}7$, $V[\mathrm{KB}^a_n, Q(^{\#}5)] = 1$. So $V[\mathrm{KB}^a_n, \alpha] = 1$.

So we get the value 1, as desired.

Now let the query α be $P(^{\#}5)$. (Thus $\mathcal{E} \cup \mathrm{KB} \not\models \alpha$.) Then $V[\mathrm{KB}, \alpha]$ will consider two new names, $^{\#}5$ and say $^{\#}7$.

- For $n = {}^{\#}5$, $V[\mathrm{KB}^a_n, P(^{\#}5)] = 1$.
- For $n = {}^{\#}7$, $V[\mathrm{KB}^a_n, P(^{\#}5)] = \frac{1}{2}$.

So we get the value $\frac{1}{2}$, as desired.

Example 8: Returning to Example 3, we let the KB be $\{\forall x(x = a \supset P(x)), \forall x(x \neq {}^{\#}1 \wedge x \neq {}^{\#}2 \supset \neg P(x))\}$. First, let the query α be $(a = {}^{\#}1 \vee a = {}^{\#}2)$. (So $\mathcal{E} \cup \mathrm{KB} \models \alpha$.) Then $V[\mathrm{KB}, \alpha]$ will need to consider three names.

- For $n = {}^{\#}1$ and $n = {}^{\#}2$, α^a_n will be a trivially true ewff. So $V[\mathrm{KB}^a_n, \alpha^a_n] = 1$.
- Any other n is inconsistent.

So we get the value 1, as desired.

Next, let the query α be $(a = {}^{\#}4)$. (So $\mathcal{E} \cup \mathrm{KB} \models \neg\alpha$.) Then $V[\mathrm{KB}, \alpha]$ will need to consider four names.

- For $n = {}^{\#}1$ and $n = {}^{\#}2$, α^a_n will be a trivially false ewff. So $V[\mathrm{KB}^a_n, \alpha^a_n] = 0$.
- Any other n is inconsistent.

So we get the value 0, as desired.

Finally, let the query α be $(a = {}^{\#}1)$. (Thus $\mathcal{E} \cup \mathrm{KB} \not\models \alpha$ and $\mathcal{E} \cup \mathrm{KB} \not\models \neg\alpha$.) $V[\mathrm{KB}, \alpha]$ will again consider three names.

- For $n = {}^{\#}1$, α^a_n will be a trivially true ewff. So $V[\mathrm{KB}^a_n, \alpha^a_n] = 1$.
- For $n = {}^{\#}2$, α^a_n will be a trivially false ewff. So $V[\mathrm{KB}^a_n, \alpha^a_n] = 0$.
- Any other n is inconsistent.

So we get the value $\frac{1}{2}$, as desired.

Computational complexity analysis. We conclude this section with a computational complexity analysis of the proposed technique. We denote by KB/α the knowledge base obtained from KB by removing all assertions that do not involve predicates occurring in α. Let's call the *width* of a formula the maximal number of free variables occurring in its subformulas [Liu and Levesque, 2003].

First observe that computing $V(\mathrm{KB}, \alpha)$ in the case of a KB that is a database amounts to the usual top down first-order query evaluation, see e.g., [Vardi, 1995], which can be done in $O(|\alpha| \cdot |\mathrm{KB}/\alpha|^w)$ where, w is the width of α. (Notice that no variables appear in the KB in this case.) If KB is a proper KB without null values, then computing $V(\mathrm{KB}, \alpha)$ can be done in time bounded by $O(|\alpha| \cdot |\mathrm{KB}/\alpha|^{w+1})$, where w is the width of α [Liu, 2006; Liu and Levesque, 2003].

Since our technique works by substituting null values with the standard names occurring in KB, by applying Liu's result, we get that:

Theorem 11: *Given the table T, computing $V(\mathrm{KB}, \alpha)$ can be done in time $O((h + k)^k \cdot |\alpha| \cdot |\mathrm{KB}/\alpha|^{w+1})$ where h is the number of standard names in* KB*, k is the number of null values in* KB*, w is the width of α.*[3]

In addition, we have that:

Theorem 12: *The table T can be computed in time $O((h + k)^k \cdot \ell^v \cdot |E|)$ where h is the number of standard names in* KB*, k is the number of null values appearing in* KB*, E is the largest formula of the $\exists\vec{x}\,(e^T_P \wedge e^F_P)$ in Theorem 6, ℓ is the number of standard names and null values appearing in E, and v is the maximum number of variables occurring in one such formula.*[4]

Let us now consider the various parameters separately. We call: *data complexity* the complexity obtained by considering as the only parameter the number of standard names mentioned in the KB and in α; *null value complexity* the complexity obtained by considering as the only parameter the number of null values in the KB; *KB complexity* the complexity obtained considering as the only parameter the size of the KB; finally, *combined complexity* the complexity obtained considering everything as a parameter.

We observe that checking for inconsistency can be done in NP: guess an assignment for the null values, guess predicate P, guess an assignment for existential values in the formula $\exists\vec{x}\,(e^T_P \wedge e^F_P)$ in Theorem 6, and evaluate equalities. Exploiting this observation, we ge that:

Theorem 13: *Computing $V(\mathrm{KB}, \alpha)$ is in:*

- *LOGSPACE (or better AC_0) in data complexity;*
- *coNP in null value complexity;*
- *coNP in* KB *complexity;*
- *PSPACE in combined complexity.*

We notice that the data complexity and the combined complexity are the same as for the evaluation of a first-order query in databases. Also, the above analysis tells us that to be efficient we really need to have a number of null values that is logarithmic in the size of the KB and similarly for the width of the query. The latter is already true for proper KBs without null values, and for databases.

A simple extension. One of the kinds of things we might want to do with null values is to include *equality information* about their unknown values in the KB itself. For example, consider a KB that contains the following sentences:

$$\begin{array}{l} (a_1 = {}^{\#}1 \;\vee\; a_1 = {}^{\#}2 \;\vee\; a_1 = {}^{\#}5) \\ (a_2 = a_3 \;\vee\; a_2 \neq {}^{\#}7) \\ (a_4 \neq a_5) \end{array}$$

[3] Notice that, $(h + k)^k \leq |\mathrm{KB}|^k$.

[4] Notice that, $O((h + k)^k \cdot \ell^v \cdot |E|) \leq O(|\mathrm{KB}|^{k+v+1})$.

This is clearly disjunctive information, but it can now be handled as part of the KB without causing any additional computational problems:

1. Let e be the conjunction of the equality sentences we want to include in the KB.
2. As we are constructing the inconsistency table T, we evaluate $V[\{\,\}, e^{\vec{a}}_{\vec{n}}]$.
3. If the answer is 0, then $\mathcal{E} \models \neg e^{\vec{a}}_{\vec{n}}$, so we add $\vec{n}$ to T.
4. Then we ignore this e when we evaluate $V[\mathrm{KB}^{\vec{a}}_{\vec{n}}, \alpha^{\vec{a}}_{\vec{n}}]$.

Example 9: Let KB be $\{\,\} \cup \{\, (a = {}^{\#}1 \vee a = {}^{\#}2) \,\}$. (So the table T might be $\{\langle {}^{\#}3 \rangle\}$, for example.) Let α be the query $(a = {}^{\#}4)$. Then $V[\mathrm{KB}, \alpha]$ needs to consider four names:

- For $n = {}^{\#}1$ and $n = {}^{\#}2$, $V[\mathrm{KB}^{a}_{n}, \alpha^{a}_{n}] = 0$.
- For $n = {}^{\#}4$, and $n = {}^{\#}6$, n is inconsistent.

So we get the value 0, as desired.

Now, let α be $(a = {}^{\#}1 \vee a = {}^{\#}2 \vee a = {}^{\#}5)$. $V[\mathrm{KB}, \alpha]$ again considers four names:

- For $n = {}^{\#}1$ and $n = {}^{\#}2$, $V[\mathrm{KB}^{a}_{n}, \alpha^{a}_{n}] = 1$.
- For $n = {}^{\#}5$, and $n = {}^{\#}7$, n is inconsistent.

So we get the value 1, as desired.

Finally, let α be $(a \neq {}^{\#}1)$. This time, $V[\mathrm{KB}, \alpha]$ considers three names:

- For $n = {}^{\#}1$, $V[\mathrm{KB}^{a}_{n}, \alpha^{a}_{n}] = 0$.
- For $n = {}^{\#}2$, $V[\mathrm{KB}^{a}_{n}, \alpha^{a}_{n}] = 1$.
- For $n = {}^{\#}5$, n is inconsistent.

So we get the value $\frac{1}{2}$, as desired.

4 Conclusion

First-order logic offers the promise of being able to reason with knowledge that is incomplete, as required in many AI applications. But so far, only formalisms that make a closed-world assumption, such as databases, appear to scale well. However, some simple forms of open-world knowledge can also be dealt with effectively: proper knowledge bases. As we have shown here, these techniques can be extended to deal with unknown individuals, provided that there are not too many of them. The same techniques can also deal with equality information about the unknown individuals. Note that we did not consider how to deal efficiently with KB updates in this paper. This is an issue for further study.

A further interesting extension would be to handle unknown individuals whose identity depends that of other individuals, for example, *bestFriend(george)*, that is, handling *Skolem functions* rather than simply Skolem constants as we have done here. Handling Skolem functions in the general case is perhaps asking for too much: Suppose e uses Skolem functions but no predicate other than equality. It is now undecidable to determine if $\mathcal{E} \models e$. But it would be interesting to know if there are some useful special cases that can be handled. For example, imagine that Skolem functions cannot appear in a query, and a variant of a proper KB where a Skolem function is only allowed to appear in the head ρ: e.g. $\forall x\, y\, (e \supset P(x, f(x, y)))$. We leave this issue for further investigation.

References

[Abiteboul *et al.*, 2006] Serge Abiteboul, Luc Segoufin, and Victor Vianu. Representing and querying XML with incomplete information. *ACM Trans. Database Syst.*, 31(1):208–254, 2006.

[Agrawal *et al.*, 2010] Parag Agrawal, Anish Das Sarma, Jeffrey D. Ullman, and Jennifer Widom. Foundations of uncertain-data integration. *PVLDB*, 3(1):1080–1090, 2010.

[Antova *et al.*, 2009] Lyublena Antova, Christoph Koch, and Dan Olteanu. 10^{10^6} worlds and beyond: efficient representation and processing of incomplete information. *VLDB J.*, 18(5):1021–1040, 2009.

[Barceló *et al.*, 2010] Pablo Barceló, Leonid Libkin, Antonella Poggi, and Cristina Sirangelo. XML with incomplete information. *J. ACM*, 58(1):4, 2010.

[Chandra and Merlin, 1977] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, pages 77–90, 1977.

[Imielinski and Lipski, 1984] Tomasz Imielinski and Witold Lipski. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[Kolaitis, 2005] Phokion G. Kolaitis. Schema mappings, data exchange, and metadata management. In *PODS*, pages 61–75, 2005.

[Lausen and Wei, 2003] Georg Lausen and Fang Wei. On the containment of conjunctive queries. In *Computer Science in Perspective*, pages 231–244, 2003.

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.

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

[Liu and Lakemeyer, 2008] Yongmei Liu and Gerhard Lakemeyer. On the expressiveness of Levesque's normal form. *J. Artif. Intell. Res. (JAIR)*, 31:259–272, 2008.

[Liu and Levesque, 2003] Yongmei Liu and Hector J. Levesque. A tractability result for reasoning with incomplete first-order knowledge bases. In *IJCAI*, pages 83–88, 2003.

[Liu, 2006] Yongmei Liu. *Tractable Reasonng in Incomplete First-Order Knowledge Bases*. PhD thesis, Department of Computer Science, University of Toronto, 2006.

[Poggi *et al.*, 2008] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

[Reiter, 1986] Raymond Reiter. A sound and sometimes complete query evaluation algorithm for relational databases with null values. *J. ACM*, 33(2):349–370, 1986.

[Ullman, 1997] Jeffrey D. Ullman. Information integration using logical views. In *ICDT*, pages 19–40, 1997.

[van der Meyden, 1998] Ron van der Meyden. Logical approaches to incomplete information: A survey. In *Logics for Databases and Information Systems*, pages 307–356, 1998.

[Vardi, 1986] Moshe Y. Vardi. Querying logical databases. *J. Comput. Syst. Sci.*, 33(2):142–160, 1986.

[Vardi, 1995] Moshe Y. Vardi. On the complexity of bounded-variable queries. In *PODS*, pages 266–276, 1995.