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

Hector Levesque
Dept. of Computer Science
University of Toronto
Toronto, Ont. M5S 3H5
hector@cs.toronto.edu

Abstract

Perhaps the simplest and most common method of dealing with incomplete knowledge in AI systems involves working with a third truth value standing for "unknown." By extending the ordinary two-valued truth tables in the obvious way, what is known about complex logical formulas can be efficiently calculated as a function of what is known about the atomic ones. While this form of reasoning can be shown to be sound, it is notoriously incomplete in classical logic. In this paper, we extend this reasoning method to include quantifiers and equality, and show that efficiency and soundness are preserved. We also show that for a wide class of first-order formulas in a certain normal form, the method is also complete. Finally, we prove that in the propositional case, every formula can be converted to a logically equivalent one in this normal form, and conjecture that this remains true in the first-order case.

1 INTRODUCTION

From the very beginnings of AI, the dream of getting a machine to exhibit common sense was linked to deductive reasoning:

> *We shall therefore say that a program has common sense if it automatically deduces for itself a sufficiently wide class of immediate consequences of anything it is told and what it already knows.*
> — John McCarthy in [14]

Since then, the enthusiasm for deduction has been tempered somewhat by what has been discovered about its computational difficulty. Regardless of how one feels about the relevance of complexity and computability theory to AI, for knowledge bases (KBs) large enough to hold what is presumed to be necessary for human-level common sense, deduction would have to be *extremely* efficient. Recent local search based methods like GSAT [17] do show some promise on large KBs, but so far (1) they are restricted to constraint satisfaction tasks not deductive ones, and (2) they work only on problems that can be formulated in a propositional language.[1]

To the best of our knowledge, there is so far only one logically correct (sound and complete) deductive technique efficient enough to be feasible on KBs of this size: the deduction underlying database retrieval. In KR terms, this amounts to what was called *vivid reasoning* in [11]. In logical terms, the requirements for this form of reasoning are clear: every relevant atomic formula must be known to be true or known to be false. That is, the KB must be equivalent to a maximally consistent set of literals. In addition, this set of literals must be readily computable. In the propositional case, one obvious way of ensuring this is to store the positive ones in a database and infer the negative ones using negation as failure. With every atom known true or known false, it then follows that every formula can be "efficiently" (in a sense to be discussed later) determined to be true or to be false by *evaluating* it, that is, by calculating its truth value as a function of the truth values of its constituent atoms.

But this requirement for complete knowledge is very strict. It would certainly be desirable to allow some atomic formulas to be *unknown*, with the understanding that other formulas would need to be unknown as well. Allowing arbitrary disjunctions (or existential quantifications) in the KB would obviously require a very different method of reasoning. A less radical move, which still allows incomplete knowledge, is to consider a KB that is equivalent to a finite consistent set of literals, not necessarily maximal. Unfortunately, although this is a trivial extension to the above, we can already see that it will not work: for the special case of a KB equivalent to the empty set of literals, the formulas

[1] Converting first-order reasoning problems into propositional ones remains a possibility (as done in [8], for example), but consider that for KBs with (say) 10^5 unique names, even a single binary predicate would generate far too many atomic propositions.

that would need to be known are precisely the *valid* ones. Computing these is co-NP hard in the propositional case, and even if we accept the argument that it may still be feasible in practice (perhaps because the query will always be small, or for reasons like those discussed in [7]), there is no escaping the fact that it would be undecidable in the first-order case.

So it appears that even a seemingly insignificant increase in expressive power, allowing for the most basic form of incompleteness in the KB, already makes deduction too hard. Despite this, it is precisely this form of incomplete knowledge that we will attempt to handle in this paper, suitably generalized to deal with quantifiers and equality. What we will prove is this:

1. when an evaluation-based reasoning procedure is applied to incomplete KBs of the above sort, it remains efficient and always provides logically sound answers;
2. for a wide class of queries in a special normal form ($\mathcal{NF}$), it also provides logically complete answers (and hence is logically correct);
3. we conjecture (and prove in the propositional case) that this special normal form entails no loss of expressive power, in the sense that every query can be equivalently expressed as a formula in $\mathcal{NF}$.

We need to clear about one thing at the outset: there is no free lunch. If the evaluation-based reasoning procedure is logically correct and efficient, then *converting a formula into $\mathcal{NF}$ will be computationally intractable.* In the first-order case, if my conjecture is correct, it must be undecidable!

So the conversion to normal form is not something we would want to do online. The application we have in mind is this: assume we have a knowledge-based program (problem solver, planner, whatever) that must use a very large external KB for some task. Embedded within the program are a number of queries to the KB, that is, a number of places where the program needs to know whether or not something is true. In an offline manner, we ensure these queries are in $\mathcal{NF}$ (perhaps by hand) before even approaching the KB. Once this is done, we can run the program that uses our deductive procedure with confidence, since we know it will efficiently generate answers that are logically sound and complete.

One caveat: even if we perform this conversion to $\mathcal{NF}$ by hand, it might still be the case that the $\mathcal{NF}$ is exponentially longer than the original query. We do not see this as a major problem. In the intended application, like with databases, we expect the query to be so small compared to the size of the KB that a worst-case exponential blowup (and the work this entails) is less of a concern. We will also end up assuming that within a first-order query, the depth of nesting of quantifiers is small relative to the size of the query. Note that these assumptions need not hold in a more mathematical (or "puzzle-mode") setting. There, the query (the theorem to be proved) might be about the same size as the KB (the given axioms), and the depth of quantifiers might be significant. For our application, think something like: depth of quantifiers < 4, length of query < 18 terms, and length of KB $< 10^6$ terms.

In the next section we discuss evaluation-based reasoning in general terms. In Section 3, we proceed with the development in detail for a first-order logic with equality, and state the main results. Finally, in Section 4 we draw some conclusions and describe some future work.

2 EVALUATION-BASED REASONING

The reasoning procedure we have in mind (for KBs with both complete and incomplete knowledge) is one that decides whether a formula is true or false, by evaluating it, reducing knowledge of complex formulas to knowledge of the ground atomic formulas, $\mathcal{A}$. Throughout, we will use 0 to mean "known to be false," 1 to mean "known to be true," and $\frac{1}{2}$ to mean "unknown."[2]

Given an assignment $V \in [\mathcal{A} \to \{0, 1, \frac{1}{2}\}]$ telling us which atoms are known, we extend the assignment to all boolean formulas in the obvious way:

1. $V[\neg\alpha] = 1 - V[\alpha]$.
2. $V[\alpha \wedge \beta] = \min\{V[\alpha], V[\beta]\}$.

This is merely a compact way of expressing the 3-valued truth tables first presented by Kleene [6]:

		$p \wedge q$			$\neg p$
	q	T	U	F	
	T	T	U	F	F
p	U	U	U	F	U
	F	F	F	F	T

Disjunctions, implications, and equivalences can be handled as abbreviations. We will sometimes also use the logical constant TRUE, with $V[\text{TRUE}] = 1$.

To handle quantification, assume we are given a finite set H of constants (intuitively, those names mentioned in some KB), and we define

3. $V[\forall x.\alpha] = \min_{c \in H^+} \{V[\alpha_c^x]\}$

Here α_c^x is the result of replacing free x by c in α, and H^+ is the union of the constants in H, those mentioned in α, and one new one outside of H and not mentioned in α. Thus,

[2] If we were to allow for inconsistent KBs as well, we would have a *fourth* truth value, as in [1, 3, 4, 5, 9, 10, 16], among many others. From an efficiency point of view, nothing is gained by this move, so we forego it for simplicity.

to evaluate $\forall x.\alpha$, we evaluate a finite set of its instances where the x ranges over the constants in the given H, over the constants mentioned in α, and over one new constant that is neither in H nor in α. We handle existentials as abbreviations.

Finally to handle equality formulas, we use the simplest possible scheme (for ground atomic ones):

4. $V[t = t'] = 1$ if t is identical to t', and 0 otherwise.

So all that is left to completely determine a V function is the set H and the value of V on atomic formulas. We will show how to get these from a given KB in Section3.3. Then, using these four rules, we can evaluate any closed formula, that is, how to compute what is known about the formula as a function of what is known about instances of its atoms.

Of course it remains to be seen whether this 3-valued evaluation scheme is any good. This is what is addressed in Sections 3.4 and Sections 3.5.

We should be clear about what we mean by correctness. We will want to talk about making deductions from a set of formulas S (the KB), and getting the correct answer (0, 1, or $\frac{1}{2}$) for a class of formulas T (the potential queries):

Definition 1: Let $S, T \subseteq \mathcal{L}$, and let $f \in [\mathcal{L} \rightarrow \{0, 1, \frac{1}{2}\}]$. Then

- f is logically *sound* wrt S for T iff for every $\alpha \in T$, if $f[\alpha] = 1$ then $S \models \alpha$, and if $f[\alpha] = 0$ then $S \models \neg\alpha$;
- f is logically *complete* wrt S for T iff for every $\alpha \in T$, if $S \models \alpha$ then $f[\alpha] = 1$, and if $S \models \neg\alpha$ then $f[\alpha] = 0$;
- f is logically *correct* wrt S for T iff it is both sound and complete.

We will see below (after we establish some properties of quantifiers and equality) that whenever we begin with an evaluation function that is logically sound for atomic formulas, it will end up logically sound for all formulas. But this will not be the case for logical completeness: it is a well known property of multi-valued logics [18] that classically correct answers for atoms do not guarantee correctness for all formulas.

Observe, for example, that we would want $V[p \vee \neg p]$ to be 1 even when $V[p] = \frac{1}{2}$, contrary to what we have above. This has suggested to some authors that perhaps tautologies and their negations need to be filtered out separately in the evaluation (as in [20] and in supervaluations [19]).

But the problem is not merely with tautologies. Suppose we have that $V[p] = \frac{1}{2}$, $V[q] = 1$ and $V[r] = 0$ (where *e.g.* $KB = \{q, \neg r\}$). Let α be the formula

$$(q \wedge (\neg r \wedge p)) \vee (\neg p \wedge (\neg r \wedge q)).$$

Then, we get $V[\alpha] = \frac{1}{2}$, whereas completeness requires a value of 1 (since $KB \models \alpha$). There is, however, a tautology hidden here: if we convert α to CNF, we get

$$[q \wedge \neg r \wedge (p \vee \neg p)],$$

which gives a value of 1, after we filter out the tautologous clause.

But consider the dual of α: $[(\neg q \vee r \vee p) \wedge (\neg q \vee r \vee \neg p)]$. For logical completeness, this should get value 0, although again V returns $\frac{1}{2}$. Moreover, the formula here is in CNF, and there are no hidden tautologous clauses to remove.[3] However, observe that the clause $(\neg q \vee r)$ is derivable from these two by Resolution, and if we were to conjoin this new clause to the formula, logical equivalence would be preserved and V would now return the correct answer, 0. This is the idea behind the normal form we will introduce later.

Notice that $(p \vee \neg p)$ (or any other valid formula) is an unusual query to appear in a knowledge-based program. We couldn't possibly want our program to do one thing when it was true, and another when it was false, for instance. The formula ought to be known, arguably, but only for logical reasons, not because of anything in the KB. Similarly, the formula α above ought to be known, but its truth is hidden in a logical puzzle. Our conversion to normal form insists on unpacking these logical puzzles within the query, and making explicit what we are asking regarding the KB. One way of saying this is that a disjunction in a query must not be entailed unless one of the disjuncts in the query is, or equivalently, the conjuncts in a query must not together entail anything that is not already entailed by one of them. This is what is behind our notion of "logically separable" below.

A few words on the efficiency of the above treatment of knowledge. If the query does not use quantifiers, V will ask for the value of an atom a linear (in the size of the query) number of times. So non-quantified queries are handled efficiently, assuming atoms are. But for quantified queries, the situation is less clear. Consider one like $\exists x_1 \cdots \exists x_n(\rho_1 \wedge \cdots \wedge \rho_m)$, where the ρ_j are atoms whose arguments are among the x_i. Even if we imagine a KB that is a simple database (a finite set of ground atoms) that uses k constants, the obvious way of handling this requires looking at all k^n vectors of constants, clearly infeasible for the sort of large k we are considering.[4] In actual database systems, queries like this can be formulated, but they are handled in practice using a number of optimizations such as sort restrictions on variables (so that not all constants need be considered for every variable), and bottom-up relational operations (like merge, join, and project). These types of optimizations will be available to us as well, and coupled with our assumption that n is very small, we take it that quanti-

[3] We could convert the formula to DNF and remove the complement of tautologous clauses, and that would work here, but not in the first-order case. See below.

[4] The worst case complexity of this problem does not look good, but is not an issue here.

fied queries can be handled efficiently (or as efficiently as can be expected), assuming again that atomic queries are.

3 FIRST-ORDER KBS AND QUERIES

We start with a standard first-order language $\mathcal{L}$ with no function symbols other than constants and a distinguished equality predicate. We assume a countably infinite set of constants $\mathcal{C} = \{c_1, c_2, \ldots\}$ for which we will be making a unique-name assumption.

3.1 QUANTIFIERS AND EQUALITY

Because we will be considering KBs and queries that use equality, we will end up wanting to compute the entailments not just of the KB, but of $\mathcal{E} \cup KB$, where we have:[5]

Definition 2: The set $\mathcal{E}$ is the axioms of equality (reflexitivity, symmetry, transitivity, substitution of equals for equals) and the (infinite) set of formulas $\{(c_i \neq c_j) \mid i \neq j\}$.

Note that because we are making a unique-name assumption for infinitely many constants, we will not be able to finitely "propositionalize" first-order KBs, despite the lack of function symbols. We will use θ to range over substitutions of all variables by constants, and write $\alpha\theta$ as the result of applying the substitutions to α. We will let ρ range over atoms whose arguments are distinct variables, so that $\rho\theta$ ranges over ground atoms. We will use $\forall\alpha$ to mean the universal closure of α. When S is finite, $\wedge S$ stands for the conjunction of its elements (and the logical constant TRUE, when S is empty). Finally, we will use e to range over *ewffs*, by which we mean quantifier-free formulas whose only predicate is equality.

Before discussing KBs and queries, we need to establish how the quantifiers and substitution behave. First we define the notion of a standard interpretation:

Definition 3: A *standard* interpretation of $\mathcal{L}$ is one where = is interpreted as identity, and the denotation relation between $\mathcal{C}$ and the domain of discourse is bijective.

The following is easy to show:

Lemma 1: *A standard interpretation M has the following properties:*

- $M \models \mathcal{E}$;
- $M \models e\theta$ *iff* $\mathcal{E} \models e\theta$;
- $M \models \forall x.\alpha$ *iff for every* $c \in \mathcal{C}$, $M \models \alpha^x_c$.

We get the following theorem:

Theorem 2: *Suppose S is any set of closed wffs, and that that there is an infinite set of constants that do not appear in S. Then $\mathcal{E} \cup S$ is satisfiable iff it has a standard model.*

Proof: Suppose we are given a model M of $\mathcal{E} \cup S$. We will show how to construct a model M' that is standard. We assume without loss of generality that the domain of M is countable, and because of $\mathcal{E}$, infinite. We begin by partitioning this domain into equivalence classes relative to the interpretation of = in M. The domain of M' will be these classes, the interpretation of = in M' will be the identity relation, and for any predicate Q, the interpretation of Q in M' is formed by taking the interpretation of Q in M and moving to the corresponding tuples of equivalence classes. Finally, to interpret a constant in M', we take the equivalence class of its denotation when the constant appears in S; otherwise, for the remaining countably infinite set of constants, we assign them in M' systematically to the remaining countably infinite set of equivalence classes, in such a way that each class is denoted by some constant. Then, it only remains to be shown that $M' \models S$. More generally, we can show that if μ is a mapping from variables to the domain of M, and μ' is the corresponding assignment to equivalence classes for M', then for any formula α all of whose constants appear in S, we have that $M, \mu \models \alpha$ iff $M', \mu' \models \alpha$. This is done by induction on the length of α. ∎

This is like Herbrand's Theorem (with $\mathcal{C}$ being like the Herbrand Universe) except that S is not required to be in prenex form, can contain arbitrary alternations of quantifiers (which would otherwise introduce Skolem functions) *etc.* Note that this is not simply the Skolem-Lowenheim Theorem either, since the theorem is false when S mentions *every* constant, as in the set $\{\exists x.P(x)\} \cup \{\neg P(c) \mid c \in \mathcal{C}\}$. This is an example of a satisfiable set that has no standard model. As a trivial consequence of the theorem, we get

Corollary 3: *If S is finite and $\mathcal{E} \cup S \models \alpha^x_c$ for every $c \in \mathcal{C}$, then $\mathcal{E} \cup S \models \forall x.\alpha$.*

Proof: If $\mathcal{E} \cup S \not\models \forall x.\alpha$, then $\mathcal{E} \cup S \cup \{\neg\forall x.\alpha\}$ is satisfiable, and since S is finite, by the Theorem, has a standard model M. Then, by Lemma 1, for some c, $M \models \neg\alpha^x_c$, and so $\mathcal{E} \cup S \not\models \alpha^x_c$. ∎

The second theorem concerns substitutions by constants:

Theorem 4: *Let S be a set of closed wffs, let α be a wff with a single free variable x, and let H^+ be a set of constants containing those in S, those in α, and at least one constant in neither. Then for every constant $d \in \mathcal{C}$, there is a constant $c \in H^+$ such that $\mathcal{E} \cup S \models \alpha^x_d$ iff $\mathcal{E} \cup S \models \alpha^x_c$.*

[5] To make the presentation here general and self-contained, we use ordinary first-order logic, with additional axioms for equality. But perhaps a better approach is to use a logical language like that presented in [9, 13] where equality is a special primitive, and the language includes "standard names" (in addition to constants and function symbols), for which the unique-name assumption is inherent in the semantics.

Proof: It is sufficient to show that if c and d are two constants that do not appear in S or α, then

$$\mathcal{E} \cup S \models \alpha^x_d \quad \text{iff} \quad \mathcal{E} \cup S \models \alpha^x_c.$$

To prove this, first let $*$ be any bijection from $\mathcal{C}$ to $\mathcal{C}$. Let α^* mean α with c replaced by c^*. Let S^* mean $\{\alpha^* \mid \alpha \in S\}$. Finally, for any interpretation M, let M^* mean the interpretation exactly like M except that the denotation of c in M^* is changed to that of c^* in M. Then, we can prove by induction that for any α, and any assignment to variables μ,

$$M^*, \mu \models \alpha \quad \text{iff} \quad M, \mu \models \alpha^*.$$

Now to prove the theorem, suppose that $\mathcal{E} \cup S \models \alpha^x_d$, and suppose that M is any interpretation such that $M \models \mathcal{E} \cup S$. We will show that $M \models \alpha^x_c$. To do so, let $*$ be the bijection that swaps c and d and leaves all other constants unchanged. Then $(\mathcal{E} \cup S)^* = (\mathcal{E} \cup S)$, and so $M \models (\mathcal{E} \cup S)^*$. By the above, we get that $M^* \models \mathcal{E} \cup S$ and so $M^* \models \alpha^x_d$. Applying the above again, we get that $M \models (\alpha^x_d)^*$, and so $M \models \alpha^x_c$, which completes the proof. ∎

It is this theorem that will allow us to restrict our attention a finite set of constants in H^+ when we do substitutions, as we will show below. Note that the theorem is false if H^+ contains just the constants in S and α. For example, let α be $P(x)$, and S be $\{\forall z(z \neq a \supset P(z))\}$. In this case, the only constant in S or α is a, and $\mathcal{E} \cup S \not\models \alpha^x_a$, but $\mathcal{E} \cup S \models \alpha^x_b$. The theorem is also false if H^+ does not contain the constants in α. For example, let α be $R(x, b)$, and S be $\{\forall y.\forall z.(y = z) \supset R(y, z)\}$. Here, $\mathcal{E} \cup S \models \alpha^x_b$, but for every other constant c, $\mathcal{E} \cup S \not\models \alpha^x_c$.

3.2 KNOWLEDGE BASES

Since we are considering a KB containing equality, variables, and universal quantifiers, we will not be able to do simple retrieval to find out what is known about the atoms. For example, let β be the formula

$$\forall x \forall y \forall z.(x \neq y \wedge z = a) \supset R(x, z, y).$$

If *KB* contains β then we want $R(b, a, a)$ to be known. So we must first be clear about the form of KB we will be using:

Definition 4: We call a set S of formulas *proper* if $\mathcal{E} \cup S$ is consistent and S is a finite set of formulas of the form $\forall(e \supset \rho)$ or $\forall(e \supset \neg\rho)$.

We will be interested in KBs that are proper. Observe that as a special case, we can represent any finite consistent set of literals as a proper KB: simply replace $\rho\theta$ (or its complement) by $\forall(e \supset \rho)$ where e is of the form $\wedge(x_i = c_i)$. We can also represent a variety of infinite sets of literals, as the formula β does above. We are free to characterize some of the positive instances of ρ by using $\forall(e \supset \rho)$, and leave the status of the rest open. We can do the same for negative instances. We can also make a closed world assumption about a predicate if we so choose, by using both $\forall(e \supset \rho)$ and $\forall(\neg e \supset \neg\rho)$, for some e and ρ.

It might appear that proper KBs are overly restrictive, and ought to be easy to reason with. It is worth remembering that deciding whether a proper KB entails a formula is recursively unsolvable, unless the formula is restricted in some way, as we intend to do.

Although proper sets are not the same as sets of literals, they can be used to represent them in the following way:

Definition 5: Let S be any finite set of $\forall(e \supset \alpha)$ formulas as above, but not necessarily consistent. Define

$$\textit{Lits(S)} = \{\alpha\theta \mid \forall(e \supset \alpha) \in S,\ \mathcal{E} \models e\theta\}.$$

Then we get the following:

Theorem 5: *Let S be a finite set of formulas of the above form, and let M be any standard interpretation. Then*

$$M \models S \quad \textit{iff} \quad M \models \textit{Lits(S)}$$

Proof: For the only-if direction, observe that $\mathcal{E} \cup S$ entails every element of *Lits(S)*. Thus, because $M \models S$ and by Lemma 1, $M \models \mathcal{E}$, it follows that $M \models$ *Lits(S)*.

For the if-direction, assume that $M \models$ *Lits(S)*, and suppose that $\forall(e \supset \alpha) \in S$, where α is either ρ or $\neg\rho$. Assume that for some θ, $M \models e\theta$. Then by Lemma 1, $\mathcal{E} \models e\theta$, so $\alpha\theta \in$ *Lits(S)*, and $M \models \alpha\theta$. Since this works for any θ, and the model is standard, we get by Lemma 1 that $M \models \forall(e \supset \alpha)$, and so M satisfies S. ∎

So S and *Lits(S)* are satisfied by the same standard interpretations (although there will be non-standard interpretations where they diverge). As corollary, we get

Corollary 6: *Let S be as above. Then $\mathcal{E} \cup S$ has a standard model iff Lits(S) is consistent.*

Proof: The only-if direction is immediate.

For the if-direction, let M be the standard interpretation whose domain is the constants $\mathcal{C}$, where each constant is interpreted as itself, where = is interpreted as identity, and where any predicate Q is interpreted as the set of tuples $\{\vec{c} \mid Q(\vec{c}) \in \textit{Lits(S)}\}$. Since, *Lits(S)* is assumed to be consistent, we get that $M \models$ *Lits(S)*, and so $M \models \mathcal{E} \cup S$ by the Theorem. ∎

3.3 ATOMIC QUERIES

Now we want to define how atomic queries will be handled for proper KBs. We will use the fact that V has already been defined for closed ewffs, and (by a simple induction argument) satisfies the following:

Lemma 7: $V[e\theta] = 1$ *iff* $\mathcal{E} \models e\theta$.

This establishes that V is logically correct for ewffs.

Definition 6: For any proper *KB*, the atomic *evaluation* associated with *KB* is the function V where the H (for handling quantifiers) is the set of constants mentioned in *KB*, and such that for any ground atom $\rho\theta$

$$V[\rho\theta] = \begin{cases} 1 & \text{if there is a } \forall(e \supset \rho) \in KB \\ & \quad \text{such that } V[e\theta] = 1 \\ 0 & \text{if there is a } \forall(e \supset \neg\rho) \in KB \\ & \quad \text{such that } V[e\theta] = 1 \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

This function is well-defined, in that, if there were formulas $\forall(e_1 \supset \rho), \forall(e_2 \supset \neg\rho) \in KB$ such that $V[e_1\theta] = V[e_2\theta] = 1$, we would have by Lemma 7 that $\mathcal{E} \models e_1\theta \wedge e_2\theta$, and so $\mathcal{E} \cup KB \models \rho\theta \wedge \neg\rho\theta$, violating the consistency of $\mathcal{E} \cup KB$.

Furthermore, the function (as a procedure) runs in time that is no worse than linear in the size of the *KB*. Given the considerations discussed in the previous section, this settles the efficiency question as far as we are concerned: using the evaluation V associated with a KB, arbitrary closed queries can be answered efficiently.

We now turn to the correctness of V.

3.4 SOUNDNESS OF QUERY EVALUATION

We begin by showing that the evaluation associated with a *KB* always returns logically correct answers for atomic queries.

Theorem 8: *For any proper KB, the evaluation associated with KB is logically correct for ground atomic queries wrt* $\mathcal{E} \cup KB$.

Proof: For soundness, assume that $V[\rho\theta] = 1$. Then by Lemma 7, we have that $\mathcal{E} \models e\theta$, where $\forall(e \supset \rho) \in KB$, from which it follows that $\mathcal{E} \cup KB \models \rho\theta$. The case when $V[\rho\theta] = 0$ is analogous. For completeness, assume that $\mathcal{E} \cup KB \models \rho\theta$. It follows that

$$\mathcal{E} \cup KB \cup \{\forall.\wedge(x_i = c_i) \supset \neg\rho\}$$

is inconsistent, for the appropriate variables and constants. This set therefore has no standard models, and by Corollary 6, *Lits(KB)* $\cup \{\neg\rho\theta\}$ is inconsistent. Thus, $\rho\theta \in$ *Lits(KB)*, from which it follows that $V[\rho\theta] = 1$. The case when $\mathcal{E} \cup KB \models \neg\rho\theta$ is analogous. ∎

Next we show that the evaluation associated with a *KB* always returns logically sound answers for any query:

Theorem 9: *Suppose KB is proper. Then the evaluation associated with KB is logically sound for any closed formula wrt* $\mathcal{E} \cup KB$.

Proof: The proof is by induction on the length of the query. The only tricky case is when the query is $\forall x.\alpha$. If $V[\forall x.\alpha] = 0$, then for some $c \in H^+$, $V[\alpha^x_c] = 0$, and so by induction, $\mathcal{E} \cup KB \models \neg\alpha^x_c$, and so $\mathcal{E} \cup KB \models \neg\forall x.\alpha$. If $V[\forall x.\alpha] = 1$ then for every $c \in H^+$, $V[\alpha^x_c] = 1$, and so by induction, for every $c \in H^+$, $\mathcal{E} \cup KB \models \alpha^x_c$. Then, by Theorem 4, we have that for every $c \in \mathcal{C}$, $\mathcal{E} \cup KB \models \alpha^x_c$, and by Corollary 3, $\mathcal{E} \cup KB \models \forall x.\alpha$. ∎

As a simple consequence of this soundness, we also have that when the *KB* happens to have complete knowledge of some set of predicates, the evaluation is logically complete:

Corollary 10: *Suppose KB is proper, and that for every* θ *and every* ρ *taken from some set of predicates,* $\mathcal{E} \cup KB \models \rho\theta$ *or* $\mathcal{E} \cup KB \models \neg\rho\theta$. *Then the evaluation associated with KB is logically complete wrt KB for any closed formula that uses just those predicates.*

Proof: First observe that $V[\alpha] \neq \frac{1}{2}$ for any query using just those predicates (by induction on the length of the query). If $\mathcal{E} \cup KB \models \alpha$, then since *KB* is proper, $\mathcal{E} \cup KB \not\models \neg\alpha$, and so by soundness, $V[\alpha] \neq 0$ and so $V[\alpha] = 1$. The case where $\mathcal{E} \cup KB \models \neg\alpha$ is analogous. ∎

This theorem shows that for complete knowledge represented as above, evaluation provides an efficient and logically correct deductive reasoning procedure.

However, as we already argued, we cannot expect to have logical correctness when knowledge is incomplete. In the next section, we show that we do get it for the special case of queries in normal form.

3.5 NORMAL FORM

This is the normal form we will be using:

Definition 7: A set S of closed formulas is logically *separable* iff for every consistent set of ground literals L, if $L \cup \{\alpha\}$ is consistent for every $\alpha \in S$, then $L \cup S$ has a standard model.

Definition 8: The normal form formulas $\mathcal{NF}$ is 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 $\wedge S \in \mathcal{NF}$;
4. if $S \subseteq \mathcal{NF}$, S is logically separable, and for some α, $S = \{\alpha^x_c \mid c \in \mathcal{C}\}$, then $\forall x.\alpha \in \mathcal{NF}$.

Before explaining how the definition works, we prove the main theorem:

Theorem 11: *Suppose KB is proper. Then the evaluation associated with KB is logically complete for any normal form formula wrt $\mathcal{E} \cup KB$.*

Proof: The proof is by induction on the length of the query. For atoms, ewffs, and negations, the argument is clear.

For conjunctions (other than ewffs), if $\mathcal{E} \cup KB \models (\alpha \wedge \beta)$ then $\mathcal{E} \cup KB \models \alpha$ and $\mathcal{E} \cup KB \models \beta$, and so by induction $V[\alpha] = 1$ and $V[\beta] = 1$, meaning $V[\alpha \wedge \beta] = 1$. If on the other hand, $\mathcal{E} \cup KB \models \neg(\alpha \wedge \beta)$ then $\mathcal{E} \cup KB \cup \{\alpha, \beta\}$ has no standard models. By Theorem 5, *Lits(KB)* $\cup \{\alpha, \beta\}$ has no standard models either. Since $(\alpha \wedge \beta) \in \mathcal{NF}$, either *Lits(KB)* $\cup \{\alpha\}$ or *Lits(KB)* $\cup \{\beta\}$ is inconsistent, and has no standard models. By Theorem 5, either $\mathcal{E} \cup KB \cup \{\alpha\}$ or $\mathcal{E} \cup KB \cup \{\beta\}$ has no standard models. By Theorem 2, $\mathcal{E} \cup KB \models \neg\alpha$ or $\mathcal{E} \cup KB \models \neg\beta$, and so by induction, $V[\alpha] = 0$ or $V[\beta] = 0$, implying $V[\alpha \wedge \beta] = 0$.

For quantifications, if $\mathcal{E} \cup KB \models \forall x.\alpha$ then for every c, $\mathcal{E} \cup KB \models \alpha^x_c$, and so by induction, $V[\alpha^x_c] = 1$, and then $V[\forall x.\alpha] = 1$. If on the other hand, $\mathcal{E} \cup KB \models \neg\forall x.\alpha$ then $\mathcal{E} \cup KB \cup \{\forall x.\alpha\}$ has no standard models. By Theorem 5, *Lits(KB)* $\cup \{\forall x.\alpha\}$ has no standard models either. Since $\forall x.\alpha \in \mathcal{NF}$, for some $c \in \mathcal{C}$, *Lits(KB)* $\cup \{\alpha^x_c\}$ is inconsistent, and has no standard models. By Theorem 5, for some $c \in \mathcal{C}$, $\mathcal{E} \cup KB \cup \{\alpha^x_c\}$ has no standard models. By Theorem 2, for some $c \in \mathcal{C}$, $\mathcal{E} \cup KB \models \neg\alpha^x_c$, and therefore by Theorem 4, for some $c \in H^+$, $\mathcal{E} \cup KB \models \neg\alpha^x_c$. Then, by induction, we have that for some $c \in H^+$, $V[\alpha^x_c] = 0$ and so $V[\forall x.\alpha] = 0$. ∎

This theorem shows that as long as the query is in normal form, we have an efficient deductive reasoning procedure for first-order KBs with incomplete knowledge that is guaranteed to be logically correct. In other words, we can evaluate a query to determine if it or its negation is entailed, and always get answers that are logically correct.

3.5.1 Examples of normal form

So see how the definitions of logically separable and normal form work, it is best to start with the propositional case (and ignore "standard" in the definition). Notice that any literal will be in normal form. If L and S are both consistent sets of literals and $L \cup S$ is inconsistent, then they contain complementary literals, and so $L \cup \{\rho\}$ is inconsistent for some $\rho \in S$. It follows that S is logically separable, according to the definition. So if S is a finite consistent set of literals (including the empty set), $\wedge S \in \mathcal{NF}$ and $\neg\wedge S \in \mathcal{NF}$. Thus $\mathcal{NF}$ contains all non-tautologous clauses and their complements. On the other hand, $\{p, \neg p\}$ is not logically separable since $\{q, p, \neg p\}$ is inconsistent, but $\{q, p\}$ and $\{q, \neg p\}$ are both consistent. So $\mathcal{NF}$ does not contain tautologous clauses and their complements.

Now the next question is which conjunctions of non-tautologous clauses will be in normal form. Consider the set $S = \{(p \vee \neg q), (\neg p \vee r)\}$. Let L be $\{q, \neg r\}$. Then $L \cup S$ is inconsistent, but both $L \cup \{(p \vee \neg q)\}$ and $L \cup \{(\neg p \vee r)\}$ are consistent, so S is not logically separable. But consider $S' = S \cup \{(\neg q \vee r)\}$. In this case, it can be shown that any consistent set of literals that is inconsistent with S' will also be inconsistent with one of the clauses in S'. The result is that $\wedge S' \in \mathcal{NF}$ and $\neg\wedge S' \in \mathcal{NF}$. As we will see more generally below, to guarantee separability in the propositional case, it is sufficient that a set of non-tautologous clauses be closed under Resolution.

To see what work this constraint will do, observe that for any *KB* and any α and β, we have that

$$\text{if } KB \models (\alpha \wedge \beta), \text{ then } KB \models \alpha \text{ and } KB \models \beta.$$

This suggests that if V gets the correct value for α and β, it will get the correct value for $(\alpha \wedge \beta)$. But the following does *not* hold in general:

$$\text{if } KB \models \neg(\alpha \wedge \beta), \text{ then } KB \models \neg\alpha \text{ or } KB \models \neg\beta,$$

for example, as above when $KB = \{q, \neg r\}$, $\alpha = (p \vee \neg q)$, and $\beta = (\neg p \vee r)$. However, the above property *does* hold when the query is in $\mathcal{NF}$, and this will give us completeness for all normal form queries.

In the first order case, the considerations are similar, with universal quantification behaving somewhat like infinite conjunction. For any *KB* and any α we have that

$$\text{if } KB \models \forall x.\alpha, \text{ then } KB \models \alpha^x_c, \text{ for every } c.$$

But the following does not hold in general:

$$\text{if } KB \models \neg\forall x.\alpha, \text{ then } KB \models \neg\alpha^x_c \text{ for some } c.$$

A simple example is when α is the formula

$$(\neg P(x) \wedge \exists y.P(y)),$$

or expressed in terms of existentials, when the query is $\exists x(P(x) \vee \forall y.\neg P(y))$. This existential is entailed by the empty KB, but no substitution instances are (nor, for that matter, are disjunctions of substitution instances). The implication is that even if V gets the correct value for every α^x_c, it need not get the correct value for $\forall x.\alpha$. The definition of $\mathcal{NF}$ rules out such formulas as queries.

One subtlety in the definition is the requirement for the model of $L \cup S$ to be *standard*. When S is finite, as in the case of conjunction, this imposes no additional constraint. But its importance can be seen from the example of α above. Here $S = \{\alpha^x_c \mid c \in \mathcal{C}\}$, and when $L \cup \{\alpha^x_c\}$ is consistent for every c, $L \cup S$ does unfortunately have a model: it is a non-standard one where $P(c)$ comes out false for every c, but $\exists x.P(x)$ comes out true. However, S is not logically separable, since $L \cup S$ cannot have a *standard* model, as S itself does not have one. Consequently, neither $\forall x.\alpha$ nor its negation are in normal form, as desired.

3.5.2 The expressive range of $\mathcal{NF}$

As a final topic, we consider the expressive range of $\mathcal{NF}$. We can prove that in the propositional sublanguage, the restriction to normal form is without loss of expressive power:

Lemma 12: *Suppose S is a finite set of ground clauses that is closed under Resolution, in the sense that*

> *If $\alpha \in S$, $\beta \in S$, and α and β resolve to γ, then either γ is tautologous or there is a $\delta \in S$ such that $\delta \subseteq \gamma$.*

Then S is logically separable.

Proof: Suppose that $L \cup S$ is inconsistent for some consistent set of literals L. We will show that for some $\alpha \in S$, $L \cup \{\alpha\}$ is inconsistent. Let $\{d_1, d_2, \ldots, d_n\}$ be a Resolution refutation of $L \cup S$, with no tautologies, and with all uses of L moved to the end. Because L is consistent, there must be at least one clause of S used. So there will be some k, $1 \leq k \leq n$, where d_k is derivable using only the clauses in S, and d_k uses just the literals in L to get to the empty clause. Thus, $L \cup \{d_k\}$ is inconsistent. Since S is closed under Resolution, there is an $\alpha \in S$, such that $\alpha \subseteq d_k$, and so $L \cup \{\alpha\}$ is inconsistent too. ∎

Theorem 13: *In the propositional sublanguage, for every $\alpha \in \mathcal{L}$, there is $\alpha' \in \mathcal{NF}$ such that $\models (\alpha \equiv \alpha')$.*

Proof: Consider the following operation on α: convert α to CNF, and starting with this set of clauses, run Resolution repeatedly on the resulting clauses, deleting any tautologous or subsumed ones until no new clauses are generated. Call the (finitely many) resulting clauses A. Since each element of A is non-tautologous, as noted above, $A \subseteq \mathcal{NF}$. Further, A is closed under Resolution, and so by the Lemma, A is logically separable. Now let α' be $\wedge A$. Then, $\alpha' \in \mathcal{NF}$, and $\models (\alpha \equiv \alpha')$, which completes the proof. ∎

The formula $\alpha' \in \mathcal{NF}$ used in the proof of this theorem is in what is called *Blake Canonical Form* (BCF) [2]. Using later terminology (due to Quine), it is the conjunction of the non-tautologous prime implicates of α. Note, however, that while $\mathcal{NF}$ includes BCF, it goes beyond it, in that it is closed under negation, and has formulas of arbitrary alternations of $\wedge$ and $\vee$. As a very simple example, suppose that α and β are in BCF and share no atoms. Then it is easy to show that $\{\neg\alpha, \neg\beta\}$ is logically separable, and so $(\alpha \vee \beta) \in \mathcal{NF}$.

I have as yet been unable to prove or disprove that the above theorem generalizes to the first-order case. To see some of the complications, consider, for example, the formula $\exists x . \neg R(a, x) \wedge R(x, b)$ as a query. This is in prenex form, has a matrix that is both in CNF and in DNF, and does not appear to involve hidden valid formulas or their negations. This seemingly innocuous formula is not in $\mathcal{NF}$, however. To see why it presents difficulties, suppose we have a *KB* where *Lits(KB)* is $\{\neg R(a, a), R(b, b)\}$. Although this KB does not entail any instance of the query, by reasoning by cases with $R(a, b)$, we can see that it entails the existential.[6] Clearly we would not be able to perform the deduction in this case by finding an appropriate substitution, and so our evaluation-based reasoning method fails on this formula.

It is not, however, a counterexample to the first-order version of the above theorem because there is a formula equivalent to it that is in $\mathcal{NF}$. It is easiest to start with the dual, $\forall x . R(a, x) \vee \neg R(x, b)$. Observe that this formula is logically equivalent to

$$\forall x . (R(a, x) \vee \neg R(x, b)) \wedge (R(a, a) \vee \neg R(b, b))$$

which, with some effort, can be shown to be in $\mathcal{NF}$. So the negation of this formula is an existential that is equivalent to the query and is in $\mathcal{NF}$.[7]

Although I cannot prove that every first-order formula has an equivalent normal form variant, it is useful to consider some special cases guaranteed to be in normal form. For example, we have

Theorem 14: *If S is proper, then $\wedge S \in \mathcal{NF}$.*

Proof: It is not hard to see that $S \subseteq \mathcal{NF}$. To see that S is logically separable, suppose that for some consistent set of literals L, $L \cup S$ has no standard model. Then by Theorem 5, $L \cup$ *Lits(S)* has no standard model. Thus, there is $\alpha\theta \in$ *Lits(S)* whose complement is L. This implies that $L \cup \{\forall(e \supset \alpha)\}$ is inconsistent, for the appropriate e. ∎

Another special case is as follows:

Definition 9: Two literals are *conflict-free* iff either they have the same polarity, or they use different predicates, or they use different constants at some argument position.

Theorem 15: *If all the literals in α are conflict-free, then $\alpha \in \mathcal{NF}$.*[8]

Proof: (sketch) Put α into a prenex CNF form. It suffices to show that if we have clauses $\gamma_1, \ldots, \gamma_n$, all of whose literals are conflict-free, then the set of instances of $\{\gamma_1, \ldots, \gamma_n\}$ is logically separable. ∎

---

[6] This is variant of the "3 block problem" discussed in [15]. In our case, from the given KB, we know that the x in question must be a or b, but we cannot say which.

[7] Observe that the conversion to $\mathcal{NF}$ required conjoining a non-subsumed entailed clause to the universal formula, as we do with the BCF. To prove the theorem, it would suffice to show that only finitely many such clauses, perhaps with new variables and equalities, are ever required.

[8] A literal appears in α if the corresponding atom appears within the scope of an appropriate number (odd or even) of negation operators.

As a further special case of this theorem, if we have a query where every predicate letter appears only positively or only negatively, we are guaranteed to be in normal form, and so to get logically correct answers.

4 CONCLUSION

In this paper, we have *not* proposed a new way of doing deductive reasoning. Rather, we have proved some properties of an existing and well known method (in the propositional case, at least) for efficiently dealing with incomplete knowledge: to determine whether or not a formula is known, we evaluate it, reducing the question to what is known about (instances of) its atomic components. Of course, other more general methods do exist for handling incomplete knowledge efficiently, and our results say nothing about the sort of incomplete knowledge that arises from disjunctions or existentials in a KB. For these, one needs to use a more complex deductive procedure to guarantee efficiency, such as the one described in [9, 16]. I suspect that similar completeness theorems could be proved about that system as well.

The results here can be thought of as one possible approach to the expressiveness / tractability tradeoff [12], especially for large KBs: we allow a very limited (but arguably, quite useful) form of incompleteness in the KB, but to preserve tractability, we insist that the query be in a certain normal form.

In my opinion, it is *not* useful to think of the results here as suggesting some sort of two-phased approach to reasoning with incomplete knowledge: first, perform query optimization by putting the query into $\mathcal{NF}$, and second, perform evaluation-based reasoning. The first phase is too difficult to do in an online automated way. A good analogy is with programs that terminate properly. We would not expect a system that *executes* programs to first filter out those programs that run forever. Rather, we make it *our* responsibility to write programs that terminate properly, and where necessary prove (by hand, usually) that they terminate before even submitting them for execution.

As to future work, I believe that it would not be overly difficult to generalize our KB to include predicates that are *defined* in terms of more basic ones (in a stratified way), preserving both correctness and efficiency. Incorporating Skolem constants (null values) in the KB is another possible extension. It would also be useful to characterize other easy-to-check special cases of normal form queries. But the more pressing open question, perhaps, is the expressive range of these queries: does every closed formula have an equivalent normal form? Should the normal form defined here be shown to be less than fully general, the obvious next step would be to look for a more expressive one that remains tractable.

Acknowledgments

The research reported here was done while the author was visiting at Simon Fraser University and York University. I most gratefully acknowledge my Engine Room cronies Jim Delgrande and Pat Dymond for technical discussions, computer support, and fun and games. I would also like to thank David Mitchell, Gerhard Lakemeyer, and Ray Reiter for helpful comments, and John Funge for interesting conversations about real-valued fluents and intervals, which originally inspired the work here. Finally, I acknowledge the Natural Sciences and Engineering Research Council of Canada for the moolah.

References

[1] Belnap, N., "A useful four-valued logic", in *Modern uses of multiple-valued logic*, J. Dunn and G. Epstein, eds., Reidel Publishing Co., 1977, pp. 8–37.

[2] Blake, A., *Canonical expressions in Boolean algebra*, Ph. D. thesis, University of Chicago, 1938.

[3] Cadoli. M. and Schaerf, M., "Approximate reasoning and non-omniscient agents", in *Proc. TARK 92*, Los Altos, CA, 1992, Morgan Kaufmann Publishers, Inc., pp. 169–183.

[4] Dunn, M., "Intuitive semantics for first-degree entailments and 'coupled trees"', *Philosophical Studies*, **29**, pp. 149–168, 1976.

[5] Ginsberg. M. "Multivalued logics: a uniform approach to reasoning in Artificial Intelligence", *Computational Intelligence*, **4**, pp. 265–316, 1988.

[6] Kleene, S. "On a notation for ordinal numbers", *Journal of symbolic logic*, **3**, pp. 150-155, 1938.

[7] Hogg, T., Huberman, B., Williams, C. (eds.), *Frontiers in problem solving: phase transitions and complexity*, special issue of *Artificial Intelligence*, **81**, March 1996.

[8] Kautz, H., McAllester, D., Selman, B. "Encoding plans in propositional logic", in *Proc. of KR-96*, Cambridge, MA, 1996.

[9] Lakemeyer, G., *Models of belief for decidable reasoning in incomplete knowledge bases*, Ph. D. thesis, Dept. of Computer Science, University of Toronto, Toronto, Ont., 1990.

[10] Levesque, H., "A logic of implicit and explicit belief", in *Proc. AAAI-84*, pp. 198–202, Austin, TX, 1984.

[11] Levesque, H. "Making believers out of computers," *Artificial Intelligence*, **30**, 1986, pp. 81–108.

[12] Levesque, H., Brachman, R. "Expressiveness and tractability in knowledge representation and reasoning," *Computational Intelligence*, **3**, 2, May 1987, pp. 78–93.

[13] Levesque, H., Lakemeyer, G. *The Logic of Knowledge Bases: Foundations of a Functional Approach to Knowledge Representation*, technical monograph, coming soon.

[14] McCarthy, J, "Programs with common sense," in *Semantic Information Processing*. M. Minsky, *ed.* Cambridge, MA: The MIT Press, 1968, pp. 403–418. Reprinted in *Readings in Knowledge Representation.* R. J. Brachman and H. J. Levesque, *eds.* San Mateo, CA: Morgan Kaufmann Publishers, Inc., 1985, pp. 299–307.

[15] Moore, R., "The role of logic in knowledge representation and commonsense reasoning," in *Proceedings of AAAI-82*, Pittsburgh, PA, pp. 428–433, 1982.

[16] Patel-Schneider, P., "A decidable first-order logic for knowledge representation," *Proceedings of IJCAI-85*, Los Angeles, CA, pp. 455–458, 1985.

[17] Selman, B., Levesque, H., and Mitchell, D. "A new method for solving hard instances of satisfiability," in *Proc. of AAAI-92*, San Jose, Aug. 1992, pp. 440–446.

[18] Urquhart, A., "Many-valued logic", in *Handbook of philosophical logic, vol. III*, D. Gabbay and F. Guenthner (eds.), Reidel Publishing Company, 1986.

[19] Van Fraasen, B., "Singular terms, truth-value gaps, and free logic", *Journal of philosophical logic*, **63**, pp. 481–495, 1966.

[20] Vassiliou, Y. *A formal treatment of incomplete information in database management*, Ph. D. thesis, Dept. of Computer Science, University of Toronto, 1980.