Chapter 1

TWO APPROACHES TO EFFICIENT OPEN-WORLD REASONING

Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Rome, Italy
degiacomo@dis.uniroma1.it

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

Abstract We show how a simple but efficient evaluation procedure that is logically correct only for closed-world knowledge bases can nonetheless be used in certain contexts with open-world ones. We discuss two cases, one based on restricting queries to be in a certain normal form, and the other, arising in reasoning about actions, based on having sensing information at the right time so as to dynamically reduce open-word reasoning to closed-word reasoning.

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 (McCarthy, 1968)

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 (Selman et al., 1992) 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 query answering. In KR terms, this amounts to what was called *vivid reasoning* in (Levesque, 1986). 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 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 (Hogg et al., 1996)), 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 consider in this paper, suitably generalized to deal with quantifiers and equality. We refer to the sort of reasoning required as *open-world reasoning*, to distinguish it from closed-world reasoning where every formula is known to be true or known to be false on the one hand, and fully general reasoning, which allows for the presence of disjunctions, existentials *etc.*, on the other.[2]

What we will argue is that open-world reasoning is a middle ground that can be dealt with effectively (sometimes) using two complementary approaches:

- by restricting the class of queries to a special normal form ($\mathcal{NF}$), a simple evaluation procedure provides inference that is both logically sound and complete;
- by assuming that we have sensing information, *i.e.,* information coming from outside the system, available at the right time, we can reduce an otherwise open-world reasoning problem to a closed-world one, and again use the simple evaluation procedure.

Here we describe the two approaches and state the main theorems without proof. Further technical details on the two approaches can be found in (Levesque, 1998) and (De Giacomo and Levesque, 1999) respectively.

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, $\mathcal{A}$.[3] 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."[4]

Given an assignment $V \in [\mathcal{A} \rightarrow \{0, 1, \frac{1}{2}\}]$ 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. $V[\alpha \wedge \beta] = \min\{V[\alpha], V[\beta]\}$.

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

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

3. $V[\forall x.\alpha] = \min\limits_{c \in H^+}\{V[\alpha^x_c]\}$

Here α^x_c 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, 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 Section 3.3. Then, using these four rules, we can evaluate any closed formula, that is, 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 in what contexts this 3-valued evaluation scheme can be used. This is what is addressed in Sections 3.4 and 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 <u>sound</u> 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 <u>complete</u> 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 <u>correct</u> 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 (Urquhart, 1986) 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 (Vassiliou, 1980) and in supervaluations (Van Fraasen, 1966)).

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.[5] 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.

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.[6] 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 sophisticated implementations of relational operations (e.g, join, selections) and careful subgoal (join) ordering and selection placement. These types of optimizations will be available to us as well, and coupled with an assumption that n is very small, we take it that quantified queries can be handled efficiently (or as efficiently as can be expected), assuming again that atomic queries are.

3. A FIRST APPROACH

The first approach which will allow us to use the above evaluation procedure requires queries to be in a certain normal form. But first we must be clear about the sorts of KBs we will be using. For the purposes of this section, 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:

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 use ρ to range over atoms (other than equalities) 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.*

We get the following theorem:

Theorem 1 *Suppose S is any set of closed wffs, and 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.*

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 a variant of the Skolem-Lowenheim Theorem either, since our theorem does not hold 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.

The second theorem concerns substitutions by constants:

Theorem 2 *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 $\mathcal{E} \cup S \models \alpha^x_c$.*

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

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

If a 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 <u>proper</u> if $\mathcal{E} \cup S$ is consistent and S is a finite set of formulas of the form $\forall(e \supset \rho)$ or $\forall(e \supset \neg\rho)$, where e is an ewff, and ρ is an atom as above.*

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*

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

Then we get the following:

Theorem 3 *Let S be a finite set of formulas of the above form, and let M be any standard interpretation. Then*

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

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

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 4 *For any ewff e, $V[e\theta] = 1$ iff $\mathcal{E} \models e\theta$.*

This establishes that V is logically correct for ewffs.

Definition 6 *For any proper KB, the atomic <u>evaluation</u> 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 $\rho\theta$*

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

This function is well-defined: 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 4 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 5 *For any proper KB, the evaluation associated with KB is logically correct for ground atomic queries wrt $\mathcal{E} \cup KB$.*

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

Theorem 6 *Suppose KB is proper. Then the evaluation associated with KB is logically sound for any closed formula wrt $\mathcal{E} \cup KB$.*

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 state the main theorem:

Theorem 7 *Suppose KB is proper. Then the evaluation associated with KB is logically complete for any normal form formula wrt $\mathcal{E} \cup KB$.*

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.

Moreover, we can prove that in the propositional sublanguage, the restriction to normal form is without loss of expressive power:

Theorem 8 *In the propositional sublanguage, for every $\alpha \in \mathcal{L}$, there is $\alpha' \in \mathcal{NF}$ such that $\models (\alpha \equiv \alpha')$.*

This is not suggest that a good general query procedure would be to first convert a formula into normal form, and then apply the evaluation procedure; such an α' could be exponentially larger than the original α. The formula $\alpha' \in \mathcal{NF}$ used in the proof of this theorem is in what is called *Blake Canonical Form* (BCF) (Blake, 1938). 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 $(\alpha \vee \beta) \in \mathcal{NF}$.

We have as yet been unable to prove or disprove that every first-order formula has an equivalent normal form variant. However, it is useful to consider some special cases guaranteed to be in normal form. For example, we have

Theorem 9 *If S is proper, then $\wedge S \in \mathcal{NF}$.*

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 10 *If all the literals in α are conflict-free, then $\alpha \in \mathcal{NF}$.*[7]

Roughly speaking, this means that if we have a query where nothing can be inferred using the query alone (because none of its literals conflict), then we can use the evaluation procedure. As a further special case, 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. A SECOND APPROACH

The second approach to open-world reasoning which will allow us to use the evaluation procedure of Section 2 requires sensing, *i.e.,* getting knowledge from outside the system, to fill in details about otherwise unknown atoms. This approach is most meaningful in a context where we are reasoning about actions and their effects.

4.1 PROJECTION

One of the most fundamental tasks concerned with reasoning about actions is the *projection task*: determining whether a fluent[8] does or does not hold after performing a sequence of actions. In the usual formulation, we are given a characterization of the initial state of the world and some sort of specification of what each action does. The projection task requires us to determine the cumulative effects (and non-effects) of sequences of actions.

Projection is clearly a prerequisite to *planning*: we cannot figure out if a given goal is achieved by a sequence of actions if we cannot determine what holds after doing the sequence. Similarly, the *high-level program execution task* (Levesque et al., 1997), which is that of finding a sequence of actions constituting a legal execution of a high-level program, also requires projection: to execute a program like "while there is a block on the table, pick up a block and put it away," one needs to be able to determine after various sequences of actions if there is still a block on the table. For these reasons being able to solve the projection problem efficiently is a clear desiderata.

Reiter (Reiter, 1991) proposed action theories of a very special form in the language of the *situation calculus* (McCarthy and Hayes, 1969). Such theories, called *basic action theories*, have a notable characteristic that allows us to base projection on special form of evaluation (*regression*) plus inference about the initial situation. This allows for a very efficient way of reasoning when we have complete information about the initial situation. Reiter's basic action theories are the starting point of our discussion.

4.2 BASIC ACTION THEORIES

The basic action theories account of action and change is formulated in the language of the situation calculus (McCarthy and Hayes, 1969; Reiter, 2000). We will not go over the language here except to note the following components: there is a special constant S_0 used to denote the *initial situation*, namely the one in which no actions have yet occurred; there is a distinguished binary function symbol *do* where $do(a, s)$ denotes the successor situation to s resulting from performing action a; relations whose truth values vary from situation to situation, are called (relational) *fluents*, and are denoted by predicate symbols taking a situation term as their last argument; and there is a special predicate $Poss(a, s)$ used to state that action a is executable in situation s.

Within this language, we can formulate action theories that describe how the world changes as the result of the available actions. In particular, basic action theories have the following form (Reiter, 1991):

- Some foundational, domain independent axioms.
- Unique names axioms for the primitive actions.
- Axioms describing the initial situation S_0.
- Action precondition axioms, one for each primitive action a, characterizing $Poss(a, s)$.
- Successor state axioms, one for each fluent F, of the following form:[9]

  $$F(\vec{x}, do(a, s)) \equiv \gamma(\vec{x}, a, s)$$

  which state under what conditions $F(\vec{x}, do(a, s))$ holds as function of what holds in situation s. These take the place of the so-called effect axioms, but also provide a solution to the frame problem (Reiter, 1991).

We will focus mainly on successor state axioms in the following.

Example 11 For example, the successor state axiom:

$$\begin{aligned} &Broken(x, do(a, s)) \equiv \\ &\qquad a = drop(x) \wedge Fragile(x) \\ &\quad \vee\ \exists b\, [a = explode(b) \wedge Bomb(b) \wedge Near(x, b, s)] \\ &\quad \vee\ a \neq repair(x) \wedge Broken(x, s) \end{aligned}$$

states that an object x is broken after doing action a if a is dropping it and x is fragile, a is exploding a bomb near it, or it was already broken, and a is not the action of repairing it. ■

In this setting the projection problem amounts to checking if

$$\Sigma \models \phi(do(\vec{A}, S_0))$$

where Σ is the basic action theory describing the domain of interest, $\vec{A}$ is a sequence of actions to perform, $do(\vec{A}, S_0)$ is the situation that results from performing the sequence of actions $\vec{A}$ starting in the initial situation S_0, and ϕ is a formula with a single situation term, a free variable ranging over situations. If the logical implication holds then we know that ϕ holds after performing $\vec{A}$ starting from S_0.

The special form of the successor state axioms allows us to *regress* fluents in the sense that whether or not they hold after performing an

```
Formula regression (Formula $\phi$, Situation $S$)
{
  while ($S \neq S_0$) {
       assume $S$ is $do(A, S')$;
       for each $F(\vec{t}, s)$ in $\phi$, simultaneously do {
         assume the SSA for $F$ is $F(\vec{z}, do(a, s)) \equiv \gamma(\vec{z}, a, s)$;
         replace $F(\vec{t}, s)$ by $\gamma(\vec{t}, A, s)$;
       }
       set $S = S'$;
  }
  return $\phi$;
}
```

Figure 1.1 Regression procedure for basic action theories

action can be determined by considering the action in question and what was true just before. By applying regression steps several times, we can regress each fluent in a formula ϕ all the way back to the initial situation. Intuitively we just have to use the regression procedure sketched in Figure 1.1 (see (Reiter, 1991) for the formal definition of regression). Observe that in the procedure, we use the pseudo-instruction `assume` S `is` $do(A, S')$`;` to make explicit the form of S, similarly for the SSA. Observe also that we do not instantiate the situation argument in Φ. It is the variable S that keeps track of the current situation. The formula returned is then to be evaluated in the initial situation, by substituting S_0 as the situation argument. For a formal definition of regression see (Reiter, 1991).

Note that using regression we are able to reduce a projection problem efficiently to an inference to be done in the initial situation. Now if we have *complete information* about the initial situation, then we just have to *evaluate* the formula obtained (using a variant of the procedure in Section 2) instead of making use of full logical inference. In other words, by using regression and making a closed-world assumption about the initial situation we get an efficient evaluation procedure for the entire projection task.

Of course, without this closed-world assumption, we cannot use evaluation (unless we restrict queries as we did in Section 3). In addition, basic action theories, by adopting this form of successor state axioms, also require a strong completeness assumption: after specifying the (perhaps conditional) effects of the given actions on fluents, and then allowing for possible ramifications of these actions (*e.g.,* (Lin and Reiter, 1994)), it is then assumed that a fluent changes *only if* it has been affected in one of these ways. What is not allowed, in other words, are cases where the value of a fluent does not depend only on the previous situation. This can arise in at least two ways. First, a fluent might change as the result of an action that is exogenous to the system. If a robot opens a door in a building, then when nobody else is around, it is justified in concluding that the door remains open until the robot closes it. But in a building with other occupants, doors will be opened and closed unpredictably. Secondly, the robot might have incomplete knowledge of the fluent in question. For example, a robot normally would not be able to infer the current temperature outdoors, since this is the result of a large number of unknown events and properties.

In cases such as these, the only way we can expect a robot to be able to perform the projection task for arbitrary queries using evaluation is if it has some sensing capabilities in order to determine the current value of certain fluents in the world. In (Levesque, 1996), sensing is modeled as an action performed by a robot that returns a binary measurement. The robot then uses so-called *sensed fluent axioms* to correlate the value returned with the state of various fluents. However, in this account, no attempt is made to be precise about the exact relation between sensing and regression. Moreover, there is no possibility of saying when regression should be used, and when sensing should be used.

In (De Giacomo and Levesque, 1999) a formal specification of a changing world is proposed which generalizes Reiter's solution to the frame problem to allow conditional successor state axioms, and generalizes the treatment of sensors by Levesque and others (*e.g.,* (Baral and Son, 1997; Golden and Weld, 1996; Poole, 1995; Weld et al., 1998)) to allow conditional sensing axioms. The specification is sufficiently general that in some cases, there is simply not enough information to perform the projection task even with sensing. However, in many cases, it allows for solving projection efficiently, by using an evaluation procedure that combines sensing and regression. In the following, we analyze such a proposal in greater detail.

4.3 GUARDED ACTION THEORIES

We assume that a robot has a number of onboard sensors that provide sensing readings at any time. Formally, we introduce a finite number of *sensing functions*, which are unary functions whose only argument is a situation. For example, $\mathsf{thermometer}(s)$, $\mathsf{sonar}(s)$, $\mathsf{depthGauge}(s)$, might all be real-valued sensing functions.[10]

We then define a *sensor-fluent formula* to be a formula of the language (without *Poss*, for simplicity) that uses at most one situation term, which is a variable, and that this term only appears as the final argument of a fluent or sensor function. We write $\phi(\vec{x}, s)$ when ϕ is a sensor-fluent formula with free variables among the $\vec{x}$ and s, and $\phi(\vec{t}, t_s)$ for the formula that results after the substitution of $\vec{x}$ by the vector of terms $\vec{t}$ and s by the situation term t_s. A *fluent formula* is one that mentions no sensor functions. A *sensor formula* is a sensor-fluent formula that mentions sensor function, but does not mention fluents, and is assumed to be easily evaluable given the values of the sensors.

A guarded action theory is like a basic action theory except that for each fluent, instead of a single successor state axiom, it contains any number of *guarded successor state axioms* and *guarded sensed fluent axioms.*

- A *guarded successor state axiom* (GSSA) is a formula of the form

  $$\alpha(\vec{x}, a, s) \supset [F(\vec{x}, do(a, s)) \equiv \gamma(\vec{x}, a, s)]$$

  where α is a sensor-fluent formula called the *guard* of the axiom, F is a relational fluent, and γ is a fluent formula.

- A *guarded sensed fluent axiom* (GSFA) is a formula of the form

  $$\alpha(\vec{x}, s) \supset [F(\vec{x}, s) \equiv \rho(\vec{x}, s)]$$

  where α is a sensor-fluent formula called the *guard* of the axiom, F is a relational fluent, and ρ is a sensor formula.

The following examples show what a guarded action theories can express.

Example 12 The outdoor temperature is unpredictable from state to state. However, when the robot is outdoors, its onboard thermometer measures that temperature.

$$\begin{aligned}&\mathit{Outdoors}(s) \supset\\ &\quad \mathit{OutdoorTemp}(n, s) \equiv \mathsf{thermometer}(s) = n\end{aligned}$$

Note that when the guard is false, *i.e.,* when the robot is indoors, nothing can be concluded regarding the outdoor temperature. ∎

Example 13 The indoor temperature is constant when the climate control is active, and otherwise unpredictable. However, when the robot is indoors, its onboard thermometer measures that temperature:

$$\begin{aligned}&\mathit{Indoors}(s) \supset\\ &\quad \mathit{IndoorTemp}(n, s) \equiv \mathsf{thermometer}(s) = n\\ &\mathit{ClimateControl}(s) \supset\\ &\quad \mathit{IndoorTemp}(n, do(a, s)) \equiv \mathit{IndoorTemp}(n, s)\end{aligned}$$

Note that in this case, if the climate control remains active, then a robot that goes first indoors and then outdoors will still be able to infer the current indoor temperature using both sensing and regressing. To our knowledge, no other representation for reasoning about actions can accommodate this combination. ■

Example 14 If the robot is alone in the building, the state of the door is completely determined by the robot's *open* and *close* actions. Either way, any time the robot is in front of the door, its onboard door sensor correctly determines the state of the door.

$$\begin{aligned}
&\mathit{Alone}(s) \supset \\
&\quad \mathit{DoorOpen}(x, do(a, s)) \equiv \\
&\qquad\quad a = \mathit{open}(x) \\
&\qquad \vee\ a \neq \mathit{close}(x) \wedge \mathit{DoorOpen}(x, s) \\
&\mathit{InFrontOf}(x, s) \supset \\
&\quad \mathit{DoorOpen}(x, s) \equiv \mathsf{doorSensor}(s) = 1
\end{aligned}$$

One intriguing possibility offered by this example is that on closing a door, and later coming back in front of the door to find it open, a security guard robot would be able to infer that $\neg \mathit{Alone}$. ■

Observe that guarded action theories are indeed an extension of basic action theories. We can handle a universally applicable successor state axiom like the one for *Broken* above by using the guard *TRUE*. Similarly, we can handle the case where nothing is known either about how to regress a fluent or how to sense its value (or both) by dropping GSSAs and GSFAs for the fluent all together.

Histories and the projection task. Once sensors are introduced, to determine if a fluent holds at some point, it is no longer sufficient to know the actions that have occurred; we also need to know the readings of the sensors along the way (*i.e.,* initially, and after each action). Consequently, we define a *history* as a sequence of the form $(\vec{\mu_0}) \cdot (A_1, \vec{\mu_1}) \cdots (A_n, \vec{\mu_n})$ where A_i $(1 \leq i \leq n)$ is a ground action term and $\vec{\mu_i} = \langle \mu_{i1}, \ldots, \mu_{im} \rangle$ $(0 \leq i \leq n)$ is a vector of values, with μ_{ij} understood as the reading of the j-th sensor after the i-th action. If λ is such a history, we then recursively define a ground situation term $\mathit{end}[\lambda]$ by $\mathit{end}[(\vec{\mu_0})] = S_0$ and $\mathit{end}[\lambda \cdot (A, \vec{\mu})] = do(A, t)$ where $t = \mathit{end}[\lambda]$. We also define a ground sensor formula $\mathit{Sensed}[\lambda]$ as $\bigwedge_{i=0}^{n} \bigwedge_{j=1}^{m} h_j(\mathit{end}[\lambda_i]) = \mu_{ij}$ where λ_i is the subhistory up to action i, $(\vec{\mu_0}) \cdots (A_i, \vec{\mu_i})$, and h_j is the j-th sensor function. So $\mathit{end}[\lambda]$ is the situation that results from doing the actions in λ and $\mathit{Sensed}[\lambda]$ is the formula that states that the sensors had the values specified by λ.[11]

The projection task, becomes as follows: given an action theory Σ as above, a history λ, and a formula $\phi(s)$, where s is the situation argument, determine whether or not

$$\Sigma \cup \mathit{Sensed}[\lambda] \models \phi(\mathit{end}[\lambda]).$$

Example 15 As an example, assume we have a robot with a single sensor that measures the temperature. One possible history then is as follows:

$$\begin{aligned}\lambda = \ & (26^o) \cdot \\ & (goIndoors, 20^o) \cdot \\ & (turnOnClimateControl, 19^o) \cdot \\ & (getGardenShears, 19^o) \cdot \\ & (goOutdoors, 27^o) \cdot \\ & (trimHedge, 28^o)\end{aligned}$$

This history tells us the robot initially sensed temperature 26^o, then it went indoors and sensed 20^o, then it turned the climate control on and sensed 19^o, then it took the garden shears, still sensing 19^o, then it went outdoors and started doing some gardening.

Let Σ be a guarded action theory for the robot that implies that all actions have the expected effect and moreover, that includes the GSSAs and GSFAs of Examples 12 and 13. Then we can infer the following projection:

$$\Sigma \cup \mathit{Sensed}[\lambda] \models \mathit{IndoorTemp}(19^o, \mathit{end}[\lambda]).$$

That is although the robot is outdoors and hence cannot sense the temperature, it can infer that the temperature indoors is still 19^o, since at one point in the history it was indoors and turned on the climate control when the temperature it was sensing was 19^o, and the climate control remains on, keeping the indoor temperature constant. ∎

4.4 GENERALIZED REGRESSION

In principle, the projection task as formulated can be solved using a general first-order theorem-prover. But our goal here is to keep the logical framework, but show that in common cases projection can be reduced using some form of regression plus inference about the initial situation, as done for basic action theories. In (De Giacomo and Levesque, 1999) a generalized form of regression that is a sensible compromise between

```
Formula  generalized-regression (Formula $\phi$, History $\lambda$)
{
  repeat {
      for each $F(\vec{t}, s)$ in $\phi$ {
          nondeterministically choose a GSFA
              $\alpha(\vec{z}, s) \supset [F(\vec{z}, s) \equiv \rho(\vec{z}, s)]$
          such that $\Sigma \cup Sensed[\lambda] \models \forall\alpha(\vec{t}, end[\lambda])$;
          replace $F(\vec{t}, s)$ by $\rho(\vec{t}, s)\backslash\lambda$;
      }
   if $\lambda = \lambda' \cdot (A, \vec{\mu})$ then {
      for each $F(\vec{t}, s)$ in  $\phi$, simultaneously do {
          nondeterministically choose a GSSA
              $\alpha(\vec{z}, a, s) \supset [F(\vec{z}, do(a, s)) \equiv \gamma(\vec{z}, a, s)]$
          such that $\Sigma \cup Sensed[\lambda'] \models \forall\alpha(\vec{t}, A, end[\lambda'])$;
          replace $F(\vec{t}, s)$ by $\gamma(\vec{t}, A, s)$;
      }
    }
    set $\lambda = \lambda'$;
   } until (no $F(\vec{t}, s)$ in $\phi$) or ($\lambda = (\vec{\mu_0})$);
   return $\phi$;
}
```

Figure 1.2 Regression procedure for generalized action theories

syntactic transformations and logical inference is proposed. Specifically, logical inference is required only in evaluating the *guards* to decide which GSFAs and GSSAs to apply. This implies that the regression technique proposed is effective in cases where the guards are easily evaluable.

One can get an intuitive idea of how generalized regression works by looking at the nondeterministic procedure sketched in Figure 1.2, where Σ is a guarded action theory, λ is a history, ϕ is a sensor-fluent formula, and the notation $\phi\backslash\lambda$ stands for formula that results from replacing every sensor function $h_j(s)$ in ϕ by the j-th component of the final sensor reading in λ. Observe that the procedure never instantiates the situation argument. It is the history λ that keeps track of current situation (*i.e.*, $end[\lambda]$). For a more formal definition of generalized regression, see (De Giacomo and Levesque, 1999).

Generalized regression is always sound, so to perform the projection task, it is sufficient to regress the formula and check whether the regressed formula holds in the initial situation. Unfortunately, regression in general cannot be *complete*. To see why, suppose nothing is known about fluent F; then a formula like $(F(s) \vee \neg F(s))$ will not regress even though it will be entailed by any history.

The other drawback of generalized regression is that we need to evaluate guards. However, evaluating a guard is just a sub-projection task, and so for certain "well structured" action theories, in which guards of the GSFAs for a fluent F do not depend circularly on F itself, we can again apply regression. Such theories are called *acyclic generalized action theories*.

4.5 JIT-HISTORIES

As noted above, we cannot expect to use generalized regression to evaluate sensor-fluent formulas in general: a tautology might be entailed even though nothing is entailed about the component fluents. However, in a practical setting, we can imagine never asking the robot to evaluate a formula unless the history is such that it knows enough about the component fluents, using the given GSSAs and GSFAs, and their component fluents. In general, we call a history *just-in-time* (JIT) for a formula, if the actions and sensing readings it contains are enough to guarantee that suitable formulas (including guards) can be evaluated at appropriate points to determine the truth value of all the fluents in the formula (see (De Giacomo and Levesque, 1999) for the formal definition).

Example 16 For example consider the history of the Example 15:

$$\begin{aligned}\lambda = \;& (26^o)\cdot\\ & (goIndoors, 20^o)\cdot\\ & (turnOnClimateControl, 19^o)\cdot\\ & (getGardenShears, 19^o)\cdot\\ & (goOutdoors, 27^o)\cdot\\ & (trimHedge, 28^o)\end{aligned}$$

It is easy to see that λ is a JIT history for *IndoorTemp*$(19^o, end[\lambda])$. Indeed at the end of λ the climate control is on, so we know that the indoors temperature is as it was in the previous situation. Thus we can regress the formula until we arrive to a point in the history where the robot was indoors, where the sensor readings measured the indoor temperature and the climate control is on. Observe that we do not need to require the robot to know whether the climate control was on before then, or even whether the robot is indoors now. ■

Although guarded action theories are assumed to be open-world, a JIT-history provides a sort of *dynamic* closed world assumption in that it ensures that the truth value of any fluent will be known whenever it is part of a formula whose truth value we need to determine. This allows us to evaluate complex formulas as we would if we had a normal closed world assumption, again using a variant of the procedure in Section 2 In (De Giacomo and Levesque, 1999) a procedure that evaluates a formula by generalized regression exploiting the notion of JIT-histories is presented.

5. CONCLUSION

In this paper, we have shown how a simple but efficient evaluation procedure that is logically correct only for closed-world knowledge bases could nonetheless be used in certain contexts with open-world ones. In the first case, we restrict queries to be in a certain normal form which, we conjecture, is without loss of expressive power; in the second case, we restrict queries to be for JIT-histories, where enough sensing information has been acquired to determine the truth values of the fluents in the query. For further discussion and directions for future work, see (Levesque, 1998) and (De Giacomo and Levesque, 1999).

Notes

1. Converting first-order reasoning problems into propositional ones remains a possibility (as done in Kautz et al., 1996, 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. Recent work on satisfiability with restricted first-order formulas may help here (Parkes, 1999).

2. It is interesting to observe that open-world reasoning has attracted interest of the database community as well (*e.g.,* Imielinski and Jr., 1984; Reiter, 1984; Vardi, 1985). More recently, researchers have looked at the problem of answering queries using materialized views, *i.e.,* answering queries using only a given set of materialized views (*e.g.,* Abiteboul and Duschka, 1998; Grahne and Mendelzon, 1999). This also is a form of reasoning with incomplete information.

3. Unless otherwise specified, by an atom, we do not include equality formulas. These are handled separately below.

4. If we were to allow for inconsistent KBs as well, we would have a *fourth* truth value, as in Belnap, 1977; Cadoli and Schaerf, 1992; Dunn, 1976; Ginsberg, 1988; Lakemeyer, 1990; Levesque, 1984; Patel-Schneider, 1985, among many others. From an efficiency point of view, nothing is gained by this move, so we forego it for simplicity.

5. 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.

6. Although it is not an issue here, the worst case complexity of this problem does not look good. Namely evaluating formulas of the form above, which are essentially conjunctive queries in databases, is polynomial in the size of the database, but NP-complete in the size of the formula (see Chandra and Merlin, 1977). Evaluating general first-order formulas is again polynomial in the size of the database, but PSPACE-complete in the size of the formula (see again Chandra and Merlin, 1977).

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

8. By a fluent, we mean a property of the world that changes as the result of performing actions.

9. Here and below, formulas should be read as universally quantified from the outside.

10. Syntactically, these look like functional fluents, so to avoid confusion, we only deal with relational fluents in this paper.

11. Obviously interesting histories λ have to satisfy certain legality criteria such as consistency of $\Sigma \cup Sensed[\lambda]$ and conformance to *Poss*.

References

Abiteboul, S. and Duschka, O. (1998). Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 254–265.

Baral, C. and Son, T. (1997). Approximate reasoning about actions in presence of sensing and incomplete information. In *Proc. of the 1997 Int. Logic Programming Symposium (ILPS'97)*, pages 387–401.

Belnap, N. (1977). A useful four-valued logic. In Dunn, J. and Epstein, G., editors, *Modern uses of multiple-valued logic*, pages 8–37. Reidel Publishing Company.

Blake, A. (1938). *Canonical expressions in Boolean algebra.* PhD thesis, University of Chicago.

Cadoli, M. and Schaerf, M. (1992). Approximate reasoning and non-omniscient agents. In *Proc. of the 4th Conf. on Theoretical Aspects of Reasoning about Knowledge (TARK'92)*, pages 169–183. Morgan Kaufmann Publishers.

Chandra, A. K. and Merlin, P. M. (1977). Optimal implementation of conjunctive queries in relational data bases. In *Proc. of the 9th ACM Sym. on Theory of Computing (STOC'77)*, pages 77–90.

De Giacomo, G. and Levesque, H. (1999). Projection using regression and sensors. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 160–165.

Dunn, M. (1976). Intuitive semantics for first-degree entailments and coupled trees. *Philosophical Studies*, 29:149–168.

Ginsberg, M. (1988). Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316.

Golden, K. and Weld, D. (1996). Representing sensing actions: the middle ground revisited. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 174–185.

Grahne, G. and Mendelzon, A. (1999). Tableau techniques for querying information sources through global schemas. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer-Verlag.

Hogg, T., Huberman, B., and Williams, C. (1996). Frontiers in problem solving: phase transitions and complexity. *Artificial Intelligence*, 81(1-2):1–15.

Imielinski, T. and Jr., W. L. (1984). Incomplete information in relational databases. *Journal of ACM*, 31(4):761–791.

Kautz, H., McAllester, D., and Selman, B. (1996). Encoding plans in propositional logic. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 374–384.

Lakemeyer, G. (1990). *Models of belief for decidable reasoning in incomplete knowledge bases.* PhD thesis, Department of Computer Science, University of Toronto.

Levesque, H. (1984). A logic of implicit and explicit belief. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI'84)*, pages 198–202.

Levesque, H. (1986). Making believers out of computers. *Artificial Intelligence*, 30:81–108.

Levesque, H. (1996). What is planning in the presence of sensing? In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 1139–1146.

Levesque, H. (1998). A completeness result for reasoning with incomplete first-order knowledge bases. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 14–23.

Levesque, H., Reiter, R., Lespérance, Y., Lin, F., and Scherl, R. (1997). GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31:59–84.

Lin, F. and Reiter, R. (1994). State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678.

McCarthy, J. (1968). Programs with common sense. In Minsky, M., editor, *Semantic Information Processing*, pages 403–418. The MIT Press.

McCarthy, J. and Hayes, P. (1969). Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502.

Parkes, A. (1999). *Lifted search engines for satisfiability.* PhD thesis, Dept. of Computer and Information Science, University of Oregon.

Patel-Schneider, P. (1985). A decidable first-order logic for knowledge representation. In *Proc. of the 9th Int. Joint Conf. on Artificial Intelligence (IJCAI'85)*, pages 455–458.

Poole, D. (1995). Logic programming for robot control. In *Proc. of the 14th Int. Joint Conf. on Artificial Intelligence (IJCAI'95)*, pages 150–157.

Reiter, R. (1984). Towards a logical reconstruction of relational database theory. In Brodie, M. L., Mylopoulos, J., and Schmidt, J. W., editors, *On Conceptual Modelling*. Springer-Verlag.

Reiter, R. (1991). The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press.

Reiter, R. (2000). *Knowledge in Action: Logical Foundation for Describing and Implementing Dynamical Systems*. Kluwer. In preparation.

Selman, B., Levesque, H., and Mitchell, D. (1992). A new method for solving hard instances of satisfiability. In *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI'92)*, pages 440–446.

Urquhart, A. (1986). Many-valued logic. In Gabbay, D. and Guenthner, F., editors, *Handbook of philosophical logic*, volume III, pages 71–116. Reidel Publishing Company.

Van Fraasen, B. (1966). Singular terms, truth-value gaps, and free logic. *Journal of philosophical logic*, 63:481–495.

Vardi, M. (1985). Querying logical databases. In *Proc. of the 4th ACM SIGACT SIGMOD Sym. on Principles of Database Systems (PODS'85)*, pages 57–65.

Vassiliou, Y. (1980). *A formal treatment of incomplete information in database management*. PhD thesis, Department of Computer Science, University of Toronto.

Weld, D., Anderson, C., and Smith, D. (1998). Extending graphplan to handle uncertainty and sensing actions. In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pages 897–904.