Query Evaluation and Progression in $\mathcal{AOL}$ Knowledge Bases

Gerhard Lakemeyer
Department of Computer Science
Aachen University of Technology
D-52056 Aachen
Germany
gerhard@cs.rwth-aachen.de

Hector J. Levesque
Department of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 3A6
hector@cs.toronto.edu

Abstract

Recently Lakemeyer and Levesque proposed the logic $\mathcal{AOL}$, which amalgamates both the situation calculus and Levesque's logic of only knowing. While very expressive the practical relevance of the formalism is unclear because it heavily relies on second-order logic. In this paper we demonstrate that the picture is not as bleak as it may seem. In particular, we show that for large classes of $\mathcal{AOL}$ knowledge bases and queries, including epistemic ones, query evaluation requires first-order reasoning only. We also provide a simple semantic definition of progressing a knowledge base. For a particular class of knowledge bases, adapted from earlier results by Lin and Reiter, we show that progression is first-order representable and easy to compute.

1 Introduction

A knowledge-based agent in a dynamic environment needs powerful facilities to query its knowledge base. In particular, it does not suffice to only ask what the world is like after any number of actions have occurred. As has been argued both in the case of static knowledge bases [6; 9] and in the context of reasoning about action [15; 17; 4], the query language should be able to explicitly refer to the agent's *knowledge*[1] in order to make distinctions such as knowing that versus knowing who [2] which otherwise cannot be made. This is best illustrated by an example.

Suppose we have a simple, stationary mail sorting robot whose task it is to pick up only the red letters in front of it. Initially the robot has no letters and it is told that there are two letters C and D and that at least one of them is red. (Let us also assume that, unbeknownst to the robot, both letters are red.) Then the robot should be able to answer the following queries:

1. Is there a red letter? Answer: *yes*.
2. Do you *know* which one is red? Answer: *no*.
3. Assume the robot now senses the colour of C.
   Do you now know of a particular letter that it is red? Answer: *yes*. (Note that even if C were not red, the answer would still be *yes*.)
4. The robot now picks up C.
   Are you holding all the red letters? Answer: *unknown*.
   (For all the robot knows, D could be red or not.)
5. Are you holding all the known red letters? Answer: *yes*.
   (C is the only letter known to be red.)

Recently, Lakemeyer and Levesque [4] have proposed the logic $\mathcal{AOL}$, which amalgamates the situation calculus [14] and Levesque's logic of only-knowing [7] and which has the expressiveness to handle queries such as the above. However, $\mathcal{AOL}$ employs heavy second-order machinery to achieve this and it is not clear how to use the logic in practice other than for specification purposes. In this paper we show that the picture is not as bleak as it may seem. In particular, we show that in $\mathcal{AOL}$ the evaluation of queries like those in the example requires first-order reasoning only.

Another important issue is knowledge base progression. In principle, the only information necessary to answer queries after a number of actions have occurred is the initial knowledge base together with the action sequence and the outcome of sensing actions. However, for long sequences of actions this seems hopelessly unrealistic from a computational point of view. It seems much more sensible to update the knowledge base appropriately after each action has occurred. Lin and Reiter [13] studied progression in the context of the situation calculus without sensing and epistemic notions. They show that progression can only be represented using second-order logic in general, but they identify interesting classes of theories where it remains first-order. Here we show how their approach can be applied to the more expressive language of $\mathcal{AOL}$ both at the semantic and the representational level. In particular, we adapt Lin and Reiter's definition of context-free action theories and show that progression remains first-order and efficiently computable in corresponding $\mathcal{AOL}$ knowledge bases.

The rest of the paper is organized as follows. In Section 2, we introduce the logic $\mathcal{AOL}$. In Section 3, we define how to query and progress an agent's knowledge at an abstract level. In Section 4, we consider concrete knowledge bases and discuss the issue of first-order query evaluation and progression there. The paper ends with some concluding remarks.[2]

[1] While we freely use the term knowledge, we really mean belief, but the difference is not important for the purposes of this paper.

[2] Some preliminary ideas about first-order query processing in

2 The Logic $\mathcal{AOL}$

Here we only give a brief introduction to the semantics of $\mathcal{AOL}$. The reader is referred to [4] for a more detailed account including a characterization using foundational axioms, which we omit here. (We also assume a basic familiarity with the situation calculus.)

The language of $\mathcal{AOL}$ is a dialect of the second-order predicate calculus with equality and has all the primitives of the situation calculus, and some more. There are three sorts of individuals: ordinary objects, actions, and situations. For each sort there is an infinite supply of variables. The situation variable *now* is reserved for special use. As in the situation calculus, we have the following primitives: the constant S_0 denotes the situation which corresponds to the real world before any actions have taken place; if a is an action and s a situation, then $do(a, s)$ denotes the situation resulting from doing a in s; the special predicate $Poss(a, s)$ has the intended meaning that a is executable in s; fluents like $Red(x, s)$ are relations, which have ordinary objects as arguments plus a situation argument in their final position, and are used to express how the world evolves from situation to situation; there are only finitely many fluents and action function symbols.

We also require two new special predicates, $SF(a, s)$ and $K_0(s)$, normally not present in the situation calculus, which are used to model sensing and knowledge and will be discussed in more detail in Section 2.2.

For simplicity, we also make the following restrictions: there are no constants or functions of the situation sort other than S_0 and *do*; action functions do not take situations as arguments; there are no function symbols of type object; and all predicates other than those mentioned above are fluents.

The language also includes a set of so-called standard names $\mathcal{N} = \{^{\#}1, ^{\#}2, \ldots\}$. The intended use of a standard name is to uniquely identify an object across all possible interpretations, which is useful when dealing with concepts like knowing that versus knowing who. Indeed, the semantics assumes a fixed domain of objects and these are isomorphic with the standard names. (See [6; 9] for more details.)

Atomic formulas are obtained in the usual way from the above primitives and formulas are built using the connectives $\neg, \wedge$, and $\forall$. Other connectives like $\supset$ and $\exists$ will be used as abbreviations in the usual way. We will use the following conventions: let $\vec{a} = a_1 \cdot a_2 \cdot \ldots \cdot a_n$ be a sequence of actions and s a situation. Then $do(\vec{a}, s)$ stands for $do(a_n, do(a_{n-1} \ldots, do(a_1, s) \ldots))$. ϵ denotes the empty sequence and we sometimes write $do(\epsilon, s)$ for s. Finally, we use TRUE as an abbreviation for $\forall x.(x = x)$ and FALSE for $\neg$TRUE.

2.1 Semantics

Rather than appealing to the standard semantics of FOL, $\mathcal{AOL}$ comes equipped with a nonstandard semantics derived from possible-world semantics [3], in particular, the semantics of the logic $\mathcal{OL}$ [7], which was developed to specify static knowledge bases.[3] As in possible-world semantics, the basic semantic building-block is a world. However, unlike the static case, a world in $\mathcal{AOL}$ determines what is true initially and after any number of actions have occurred. A situation is then interpreted simply as a world w indexed by a sequence of actions $\vec{a}$. In particular, every world "starts" with an initial situation where no actions have occurred yet. Besides the *real* world, whose initial situation serves as the denotation of S_0, a model in $\mathcal{AOL}$ also features a set of worlds e. As in modal logics of knowledge like $\mathcal{OL}$, e should be understood as the set of worlds which the agent considers epistemically possible. In Section 2.2, we will see how, using the special predicate K_0, the worlds in e can be accessed and how this gives us a way to define knowledge in dynamic domains.

To simplify the semantics, we assume that besides the standard names for objects there are also standard names for actions. These are terms of the form $A(n_1, \ldots, n_k)$ where A is an action function and each n_i is a standard name of an object. A primitive formula is an atom of the form $F(n_1, \ldots, n_k)$ where each n_i is a standard name, and $F(x_1, \ldots, x_k, s)$ is a relational fluent, or of the form $Poss(A)$ or $SF(A)$, where A is a standard name for an action. The set of all primitive formulas is $\mathcal{P}$.

Let Act^* be the set of all sequences of standard names for actions including the empty sequence ϵ.

Definition 2.1: A *world* w is a function:

$$w : \mathcal{P} \times Act^* \longrightarrow \{0, 1\}$$

Let $\mathcal{W}$ denote the set of all worlds.

Definition 2.2: A *situation* is a pair $(w, \vec{a})$, where $w \in \mathcal{W}$ and $\vec{a} \in Act^*$. An *initial* situation is one where $\vec{a} = \epsilon$.

Definition 2.3: An *action model* M is a pair $\langle e, w \rangle$, where $w \in \mathcal{W}$ and $e \subseteq \mathcal{W}$.

w is taken to specify the actual world, and e specifies the *epistemic state* as those worlds an agent has not yet ruled out as being the actual one. As we will see below, a situation term s will be interpreted semantically as a situation $(w, \vec{a})$, consisting of a world and a sequence of actions that have happened so far. A fluent $p(s)$ will be considered true if $w[p, \vec{a}] = 1$.

A variable map ν is a function that maps object, action, and situation variables into standard names for objects and actions, and into situations, respectively. In addition, ν assigns relations of the appropriate type[4] to relational variables. For a given ν, ν^x_o denotes the variable map which is like ν except that x is mapped into o.

The meaning of terms

We write $|\cdot|_{M,\nu}$ for the denotation of terms with respect to an action model $M = \langle e, w \rangle$ and a variable map ν. Then

$|n|_{M,\nu} = n$, where n is a standard name;

$|A(\vec{t})|_{M,\nu} = A(|\vec{t}|_{M,\nu})$, where $A(\vec{t})$ is an action term;

$\mathcal{AOL}$ first appeared in [5]. Progression was not handled at all in that paper.

[3] The reader who prefers classical logic is referred to [4], where we provide a second-order axiomatization which is sound and complete with respect to the nonstandard semantics.

[4] Since the type will always be obvious from the context, we leave this information implicit.

$|S_0|_{M,\nu} = (w, \epsilon)$;
$|do(t_a, t_s)|_{M,\nu} = (w', \vec{a} \cdot a)$, where $|t_s|_{M,\nu} = (w', \vec{a})$, and $|t_a|_{M,\nu} = a$;
$|x|_{M,\nu} = \nu(x)$, where x is any variable, including predicate variables.

Observe that in a model $M = \langle e, w \rangle$, the only way to refer to a situation that does not use the given world w is to use a situation variable.

The meaning of formulas

We write $M, \nu \models \alpha$ to mean formula α comes out true in action model M and variable map ν:

$M, \nu \models F(\vec{t}, t_s)$ iff $w'[F(|\vec{t}|_{M,\nu}), \vec{a}] = 1$, where $F(\vec{t}, t_s)$ is a relational fluent, and $|t_s|_{M,\nu} = (w', \vec{a})$;
$M, \nu \models X(\vec{t})$ iff $|\vec{t}|_{M,\nu} \in \nu(X)$ with X a relational var.;
$M, \nu \models \mathit{Poss}(t_a, t_s)$ iff $w'[\mathit{Poss}(|t_a|_{M,\nu}), \vec{a}] = 1$, where $|t_s|_{M,\nu} = (w', \vec{a})$;
$M, \nu \models \mathit{SF}(t_a, t_s)$ iff $w'[\mathit{SF}(|t_a|_{M,\nu}), \vec{a}] = 1$, where $|t_s|_{M,\nu} = (w', \vec{a})$;
$M, \nu \models K_0(t_s)$ iff $|t_s|_{M,\nu} = (w', \epsilon)$ and $w' \in e$;
$M, \nu \models t_1 = t_2$ iff $|t_1|_{M,\nu} = |t_2|_{M,\nu}$;
$M, \nu \models \neg\alpha$ iff $M, \nu \not\models \alpha$;
$M, \nu \models \alpha \wedge \beta$ iff $M, \nu \models \alpha$ and $M, \nu \models \beta$;
$M, \nu \models \forall x.\alpha$ iff $M, \nu^x_o \models \alpha$ for all o of the appropriate sort (object, action, situation, relation).

If α does not mention K_0, that is, the truth of α does not depend on e, we also write $w, \nu \models \alpha$ instead of $M, \nu \models \alpha$. Similarly, if α does not mention S_0 and, hence, does not depend on the real world, we write $e, \nu \models \alpha$. If α mentions neither S_0 nor K_0, we simply write $\nu \models \alpha$. Also, if α is a sentence, we omit the variable map and write, for example, $M \models \alpha$.

Finally, a formula α is valid in $\mathcal{AOL}$ if for all action models $M = \langle e, w \rangle$ and variable maps ν, $M, \nu \models \alpha$.

2.2 Knowledge and Action

To determine what is known initially (that is, in situation S_0), we only need to consider K_0. More precisely, a sentence is known initially just in case it holds in all situations s for which $K_0(s)$ holds. To find out what holds in successor situations, we use the predicates *SF* and *Poss*. First note that the logic itself imposes no constraints on either *SF* or *Poss*; it is up to the user in an application to write appropriate axioms. For *Poss*, these are the precondition axioms, which specify necessary and sufficient conditions under which an action is executable. So we might have, for example,

$$\mathit{Poss}(\mathsf{pickup}(x), s) \equiv \mathit{Letter}(x, s)$$

as a way of saying that the robot is able to pick up only letters. For *SF*, the user must write *sensed fluent axioms*, one for each action type, as discussed in [8]. The idea is that $\mathit{SF}(A, s)$ gives the condition sensed by action A in situation s. So we might have, for example,

$$\mathit{SF}(\mathsf{senseRed}(x), s) \equiv \mathit{Red}(x, s)$$

as a way of saying that the senseRed action in situation s tells the robot whether or not x is red. In case the action A has no sensing component (as in simple physical actions, like dropping an object), we require as a convention that the axiom states that $\mathit{SF}(A, s)$ is identically TRUE. Actions without a sensing component are referred to as *ordinary* actions.

With these terms, we can now define $K(s', s)$ as an abbreviation for a formula that characterizes when a situation s' is accessible from an arbitrary situation s:[5]

$$K(s', s) \doteq \forall R[\ldots \supset R(s', s)]$$

where the ellipsis stands for the conjunction of

$$\begin{array}{l} \forall s_1, s_2.\ \mathit{Init}(s_1) \wedge \mathit{Init}(s_2) \wedge K_0(s_2) \supset R(s_2, s_1) \\ \forall a, s_1, s_2.\ R(s_2, s_1) \wedge (\mathit{SF}(a, s_2) \equiv \mathit{SF}(a, s_1)) \wedge \\ \qquad (\mathit{Poss}(a, s_2) \equiv \mathit{Poss}(a, s_1)) \supset \\ \qquad\quad R(\mathit{do}(a, s_2), \mathit{do}(a, s_1)). \end{array}$$

Here $\mathit{Init}(s)$ stands for $\neg\exists a, s'.s = \mathit{do}(a, s')$.

If s is an initial situation, then the situations which are K-related to s are precisely those initial situations s' for which $K_0(s')$ holds. The general picture, after some actions have occurred, is best reflected by the following theorem, which shows that our definition yields the successor state axiom for a predicate K proposed in [17] as a solution to the frame problem for knowledge.[6]

Theorem 2.4: *[4]. The following sentence is valid:*

$$\begin{array}{l} \forall a, s, s'.\ \mathit{Poss}(a, s) \supset K(s', \mathit{do}(a, s)) \equiv \\ \qquad \exists s''.\ s' = \mathit{do}(a, s'') \wedge K(s'', s) \wedge \mathit{Poss}(a, s'') \\ \qquad\quad \wedge\ [\mathit{SF}(a, s) \equiv \mathit{SF}(a, s'')]. \end{array}$$

In other words, s' is K-related to $\mathit{do}(a, s)$ just in case there is some other s'' which is K-related to s and from which s' can be reached by doing a. Furthermore s and s'' must agree on the values of *SF* and *Poss* for action a.

Given K, knowledge can then be defined in a way similar to possible-world semantics [3; 1; 15] as truth in all accessible situations. Knowing is then denoted using the following macro, where α may contain the special situation variable now. Let α^{now}_s refer to α with all occurrences of now replaced by s. Then

$\mathtt{Knows}(\alpha, \mathtt{s}) \doteq \forall \mathtt{s}' \mathtt{K}(\mathtt{s}', \mathtt{s}) \supset \alpha^{now}_{\mathtt{s}'}$
where s' is a new variable occurring nowhere else in α.

Note that α itself may contain Knows with the understanding that macro expansion works from the innermost occurrence of Knows to the outside. For example, $\mathtt{Knows}(\neg\mathtt{Knows}(\mathit{Red}(\mathtt{x}, now), now), \mathtt{S}_0)$ stands for

$$\forall s K(s, S_0) \supset (\neg\forall s' K(s', s) \supset \mathit{Red}(x, s'))$$

and should be read as "the agent knows in S_0 that it does not know that x is red."

[5] We could have defined K as a predicate in the language as is usually done, but we have chosen not to in order to keep the formal apparatus as small as possible.

[6] Here we follow the notation from [8].

3 Queries and Progression

In this section, we will consider two related ways of answering queries in $\mathcal{AOL}$. For our purposes, a query is any formula with a single free situation variable, *now*. An example is $\exists x \mathit{Red}(x, now) \wedge \neg \mathtt{Knows}(\mathtt{Red}(\mathtt{x}, now), now)$, which asks whether it is *now* the case that there is a red object which is not known yet. The *now* in this query is intended to refer to a particular situation, either an initial situation or one that is the result of a sequence of actions. With this view, it is not possible to answer queries wrt an action model $M = \langle e, w \rangle$ alone, since we also need to specify what sequence of actions to use.

In our first specification of query answering, we are given an initial M, and a sequence of actions $\vec{a}$, and we answer according to what would be known in the situation resulting from doing $\vec{a}$. In other words, we answer a query α with *yes* if according to M, α is known in $do(\vec{a},S_0)$:

$$\mathrm{ASK}_0[\alpha, M, \vec{a}] = \begin{cases} \textit{yes} \text{ if } M \models \mathtt{Knows}(\alpha, do(\vec{\mathtt{a}}, \mathtt{S_0})) \\ \textit{no} \text{ if } M \models \mathtt{Knows}(\neg\alpha, do(\vec{\mathtt{a}}, \mathtt{S_0})) \\ \textit{unknown} \text{ otherwise.} \end{cases}$$

Note the difference between $\mathtt{Knows}(\alpha[now], do(\vec{\mathtt{a}}, \mathtt{S_0}))$ as above, and $\mathtt{Knows}(\alpha[do(\vec{\mathtt{a}}, now)], \mathtt{S_0})$. In the former, we are asking if α would be known after doing $\vec{a}$; in the latter, we are asking if it is known initially that α would be true after doing $\vec{a}$. It is not hard to show that the former is implied by the latter, but not vice-versa.

While this is a simple form of query answering, note that it needs to use the world w in M to decide what is known. If $\vec{a}$ consists of a single sensing action like $\mathsf{senseRed}(C)$, then after doing the sensing, the agent should know whether C is red or not. But which one is known is determined by w, which specifies (via *SF*) how sensing will turn out.

There is, however, a different view where we only need the epistemic state e to answer a query. The idea is that while an agent performs her actions, her epistemic state gets updated to reflect the changes caused by those actions. In particular, a sensing action leads to the removal of worlds which contradict the sensed value. We can define $\mathrm{SUCC}[e, w, \vec{a}]$ to be the epistemic state that results from executing $\vec{a}$ starting with initial state e with sensing as specified by w, by the following:

1. $\mathrm{SUCC}[e, w, \epsilon] = e$.
2. If $\mathrm{SUCC}[e, w, \vec{a}] = e'$, then $\mathrm{SUCC}[e, w, \vec{a} \cdot A] = \{w' \mid w' \in e' \text{ and } \nu^{s}_{(w,\vec{a})(w',\vec{a})}{}^{s'} \models [\mathit{SF}(A, s) \equiv \mathit{SF}(A, s')] \wedge [\mathit{Poss}(A, s) \equiv \mathit{Poss}(A, s')]\}$

Now given an e that is equal to $\mathrm{SUCC}[e_0, w_0, \vec{a}]$, we can define a new query operation for any query α which does not mention S_0:

$$\mathrm{ASK}[\alpha, e, \vec{a}] = \begin{cases} \textit{yes} \text{ if for all } w \in e,\ e, \nu^{now}_{(w,\vec{a})} \models \alpha. \\ \textit{no} \text{ if for all } w \in e,\ e, \nu^{now}_{(w,\vec{a})} \models \neg\alpha. \\ \textit{unknown} \text{ otherwise.} \end{cases}$$

Restricting ourselves to queries that do not mention S_0 is necessary since ASK does not carry with it the real world, which is needed as the denotation of S_0. In fact, mentioning S_0 within a query does not make much sense in the first place. Consider, for example, $\alpha = P(S_0)$. Asking whether α is true is completely independent of any epistemic state e and depends only on the initial state of the real world.

In order to compare our two notions of ASK, it is necessary to restrict the class of queries even further. In fact, we restrict ourselves to queries whose only situation term is *now*. In particular, this has the effect that we cannot ask about other past or future situations.

Definition 3.1: The interaction language $\mathcal{IL}$.
Atomic formulas whose only situation term is *now* are $\mathcal{IL}$-formulas. If α and β are $\mathcal{IL}$-formulas, then $\neg\alpha$, $\alpha \wedge \beta$, $\forall x \alpha$, where x is an object variable, and $\mathtt{Knows}(\alpha, now)$ are $\mathcal{IL}$-formulas. Nothing else is an $\mathcal{IL}$-formula. From now on, unless stated otherwise, a *query* is an $\mathcal{IL}$-formula where *now* is the only free variable.

An example query in $\mathcal{IL}$ is

$$\exists x \mathit{Red}(x, now) \wedge \neg \mathtt{Knows}(\mathtt{Red}(\mathtt{x}, now), now).$$

The formula

$$\exists x \mathit{Red}(x, now) \wedge \neg \mathtt{Knows}(\mathtt{Red}(\mathtt{x}, now), do(\mathsf{senseRed}, now)),$$

on the other hand, is not in $\mathcal{IL}$.

The formulas of $\mathcal{IL}$ are interpreted by first converting them into $\mathcal{AOL}$-formulas using the definition of $\mathtt{Knows}$ introduced in the previous section.

We then have the following relationship between ASK_0 and ASK:

Theorem 3.2: *For any $\alpha \in \mathcal{IL}$, e, w and $\vec{a}$,*

$$\mathrm{ASK}_0[\alpha, \langle e, w \rangle, \vec{a}] = \mathrm{ASK}[\alpha, \mathrm{SUCC}[e, w, \vec{a}], \vec{a}].$$

The theorem can be strengthened considerably as it holds for many queries outside of $\mathcal{IL}$ as well. In a nutshell, the only restriction needed is that a query does not refer to what is known *before* the actions $\vec{a}$ have occurred. Roughly, this is because $\mathrm{SUCC}[e, w, \vec{a}]$ knows more about the past than e because it has fewer worlds than e. However, the formulation of a broader class of queries for which the theorem holds turns out to be somewhat awkward. $\mathcal{IL}$, on the other hand, is simple and intuitive. Moreover, it is $\mathcal{IL}$ for which we develop a first-order query evaluation method in Section 4.2.

3.1 Progression

For ASK to make sense, we needed to assume that e reflected the epistemic changes that occurred during the execution of $\vec{a}$, as reflected in SUCC. In a different context, Lin and Reiter (LR) [13] have called the process of updating a knowledge base of an acting agent *progression* and they studied it in detail in the framework of the standard situation calculus.

One major difference between progression and the SUCC operation above is that in the former we attempt to forget the history of actions, and treat the resulting knowledge base as if it were an initial one.[7] Indeed, for many applications, it is sufficient to maintain information about a single "current" situation. Our definition of progression below adapts the ideas of LR to the more expressive language of $\mathcal{AOL}$. In fact, our

[7] See [12] for a formalization of forgetting.

formulation is somewhat simpler, which is possible because the semantics assumes a fixed set of worlds. It is also more general because LR do not deal with sensing.

We can define a progression operator $\mathrm{PROG}[e, w, \vec{a}]$ analogous to SUCC that produces a new epistemic state, but which loses information about the past. Given worlds w and w', we say that w' *agrees with* w *after* $\vec{a}$ if for all $\vec{c}$ and p, $w'[p, \vec{a} \cdot \vec{c}] = w[p, \vec{a} \cdot \vec{c}]$. Note that w and w' may differ arbitrarily in all situations before the last action of $\vec{a}$ has been performed. Then we define PROG by the following:

1. $\mathrm{PROG}[e, w, \epsilon] = e$.
2. If $\mathrm{PROG}[e, w, \vec{a}] = e'$, then $\mathrm{PROG}[e, w, \vec{a} \cdot A] = \{w'' \mid \exists w' \in e', w' \text{ agrees with } w'' \text{ after } \vec{a} \cdot A \text{ and } \nu^{s}_{(w,\vec{a})}{}^{s'}_{(w',\vec{a})} \models [SF(A, s) \equiv SF(A, s')] \wedge [Poss(A, s) \equiv Poss(A, s')]\}$

When $e = \mathrm{PROG}[e_0, w_0, \vec{a}]$, we say that e is a *progression* at $\vec{a}$ wrt $\langle e_0, w_0 \rangle$.

The following theorem states that progression is faithful in that it agrees with the original epistemic state for queries in $\mathcal{IL}$ about what is true after a sequence of actions has occurred.

Theorem 3.3: *Let $M = \langle e, w \rangle$ and $M_{\vec{a}} = \langle e_{\vec{a}}, w \rangle$, where $e_{\vec{a}}$ is a progression at $\vec{a}$ wrt M. Then for all queries $\alpha \in \mathcal{IL}$,*

1. $\mathrm{ASK}_0[\alpha, M, \vec{a} \cdot \vec{c}] = \mathrm{ASK}_0[\alpha, M_{\vec{a}}, \vec{a} \cdot \vec{c}]$.
2. $\mathrm{ASK}_0[\alpha, M, \vec{a}] = \mathrm{ASK}[\alpha, e_{\vec{a}}, \vec{a}]$.

Note that in the case of the empty sequence of actions, $\mathrm{ASK}_0[\alpha, M, \epsilon] = \mathrm{ASK}[\alpha, e, \epsilon]$ follows immediately.

4 $\mathcal{AOL}$ Knowledge Bases

So far, we have only talked about the agent's knowledge in the abstract, namely as a set of worlds, which include all possible ways they could evolve in the future. Let us now turn to representing the agent's knowledge symbolically and see how this connects with the semantic view taken so far.

In the situation calculus an application domain is typically characterized by the following types of axioms: action precondition axioms, successor state axioms, and axioms describing the current (often initial) situation. Successor state axioms were proposed by Reiter as a solution to the frame problem [16]. When there are sensing actions, there is also a fourth type called *sensed fluent axioms* specifying what the outcome of sensing is.

$\mathcal{AOL}$-knowledge bases, as we envisage them, consist of formulas of these types and they have a special syntactic form. We call a formula *objective* if it does not mention the predicate K_0.

A formula ϕ is called *simple* in t_s if ϕ is first-order and objective, t_s is the only situation argument occurring in any of the predicates, and any variable in t_s occurs only free in α. ($\exists x.Red(x, do(A, s))$ is simple in $do(A, s)$, whereas $\exists s, x.Red(x, do(A, s))$ is not.)

In the following, let A be an action and F a fluent. Let $\phi(\vec{u})$ denote a formula whose free variables are among the variables in $\vec{u}$.

Let $s \preceq s'$ denote that situation s' is a successor of s, which is defined as:

$$s \preceq s' \doteq \forall R[\ldots \supset R(s, s')]$$

with the ellipsis standing for the conjunction of

$\forall s_1.\ R(s_1, s_1)$
$\forall a, s_1.\ R(s_1, do(a, s_1))$
$\forall s_1, s_2, s_3.\ R(s_1, s_2) \wedge R(s_2, s_3) \supset R(s_1, s_3)$

Action Precondition Axioms:
$\forall s \forall \vec{x}.now \preceq s \supset [Poss(A(\vec{x}), s) \equiv \phi(\vec{x}, s)]$,[8]
where $\phi(\vec{x}, s)$ is simple in s.

Sensed Fluent Axioms:
$\forall s \forall \vec{x}.now \preceq s \supset [SF(A(\vec{x}), s) \equiv \phi(\vec{x}, s)]$
where $\phi(\vec{x}, s)$ is simple in s.

Successor State Axioms:
$\forall s \forall a \forall \vec{x}.now \preceq s \supset [Poss(a, s) \supset [F(\vec{x}, do(a, s)) \equiv \phi(\vec{x}, a, s)]]$, where $\phi(\vec{x}, a, s)$ is simple in s.

Current State Axioms:
ϕ, where ϕ is simple in $do(\vec{a}, now)$.

A knowledge base (at $\vec{a}$) is then a collection of formulas

$$\mathrm{KB} = \mathrm{KB}_{cur} \cup \mathrm{KB}_{Poss} \cup \mathrm{KB}_{SF} \cup \mathrm{KB}_{ss},$$

where KB_{Poss}, KB_{SF}, and KB_{ss} contain the action preconditions, sensed fuent axioms, and successor state axioms, respectively, and KB_{cur} is the set of current state axioms for a fixed $\vec{a}$. A knowledge base at ϵ is called an *initial knowledge base*.

We define the epistemic state corresponding to a KB as the set of all worlds satisfying the formulas in KB, where *now* is interpreted by initial situations. Formally,

$$\Re[\![\mathrm{KB}]\!] = \{w \mid w, \nu^{now}_{(w,\epsilon)} \models \mathrm{KB}\}.$$

Defining the epistemic state this way reflects the intuition that the KB is *all* the agent knows, hence she cannot rule out any world compatible with the sentences in KB. (See [4] for how to formalize "all I know" in $\mathcal{AOL}$.)

4.1 An Example KB

Here we consider the mail-sorting robot example in more detail. There are letters of different colours laid out in front of the robot and its task is to pick up only the red letters. To keep matters simple, there are only two actions, pickup(x), which is possible if x is a letter, and senseRed(x), which tells the robot whether the sensed object is red and which is always possible. There are three fluents, *Letter*, *Red*, and *HoldRLs*. *Letter* and *Red* never change and *HoldRLs*(x, s) is true if the robot is holding the red letter x in situation s.

We can formalize this by defining appropriate precondition axioms, sensed fluent axioms and successor state axioms, all parameterized by now.

Let *ALL*(now) stand for the set of these formulas:

[8] In the situation calculus without epistemic concepts, s ranges over all situations, namely those reachable from S_0. Here we need to relativize quantifi cation wrt *now* because there are initial situations other than S_0.

$\forall s, x.now \preceq s \supset \mathit{Poss}(\mathsf{pickup}(x), s) \equiv \mathit{Letter}(x, s)$
$\forall s, x.now \preceq s \supset \mathit{Poss}(\mathsf{senseRed}(x), s) \equiv \text{TRUE}$
$\forall s, x.now \preceq s \supset \mathit{SF}(\mathsf{pickup}(x), s) \equiv \text{TRUE}$
$\forall s, x.now \preceq s \supset \mathit{SF}(\mathsf{senseRed}(x), s) \equiv \mathit{Red}(x, s)$
$\forall s, a, x.now \preceq s \supset \mathit{Letter}(x, do(a, s)) \equiv \mathit{Letter}(x, s)$
$\forall s, a, x.now \preceq s \supset \mathit{Red}(x, do(a, s)) \equiv \mathit{Red}(x, s)$
$\forall s, a, x.now \preceq s \supset \mathit{HoldRLs}(x, do(a, s)) \equiv$
$[(a = \mathsf{pickup}(x) \wedge \mathit{Red}(x, s)) \vee \mathit{HoldRLs}(x, s)]$

Initially, the robot knows that there are at least two letters C and D and that one of them is red. Hence let

$$\mathrm{KB}_{cur} = \left\{ \begin{array}{l} \mathit{Letter}(C, now), \mathit{Letter}(D, now), \\ (\mathit{Red}(C, now) \vee \mathit{Red}(D, now)), \\ \forall x. \neg \mathit{HoldRLs}(x, now). \end{array} \right\}$$

Let $\mathrm{KB} = \mathit{ALL}(now) \cup \mathrm{KB}_{cur}$.

Let the real world w be any world such that $w \models \mathit{ALL}(now)^{now}_{S_0} \wedge \mathit{Letter}(C, S_0) \wedge \mathit{Red}(C, S_0) \wedge \mathit{Letter}(D, S_0) \wedge \mathit{Red}(D, S_0)$, that is, the actions indeed behave as the robot expects them to and there are at least two red letters C and D. Finally, let $M = \langle \Re[\![\mathrm{KB}]\!], w \rangle$ be our action model.

4.2 First-Order Query Evaluation

By lifting results from Levesque [6; 9], we show that answering epistemic queries for KB's like the above requires only first-order reasoning.

For any formula α simple in $do(\vec{a}, now)$ let $\alpha\!\downarrow$ be α with all occurrences of $do(\vec{a}, now)$ removed. For example, $\mathit{Red}(C, now)\!\downarrow = \mathit{Red}(C)$. Let S_{U} denote the set of sentences expressing the unique names assumption for standard names and actions, and let $\models_{\text{FOL}}$ denote classical first-order logical implication.

The following definition of $\mathrm{RES}[\phi, \mathrm{KB}]$ shows how to compute in FOL the known instances of ϕ and representing it as a first-order equality expression.

Definition 4.1: Let $\mathrm{KB} = \mathrm{KB}_{cur} \cup \mathrm{KB}_{Poss} \cup \mathrm{KB}_{SF} \cup \mathrm{KB}_{ss}$ and ϕ an objective query and let $n_1, \ldots, n_k$ be all the standard names occurring in KB and ϕ and let n' be a name not occurring in KB or ϕ. Then $\mathrm{RES}[\phi, \mathrm{KB}]$ is defined as:

1. If $\phi\!\downarrow$ has no free variables, then $\mathrm{RES}[\phi, \mathrm{KB}]$ is
   TRUE, if $\mathrm{KB}_{cur}\!\downarrow \cup S_{\mathrm{U}} \models_{\text{FOL}} \phi\!\downarrow$, and
   FALSE, otherwise.
2. If x is a free variable in $\phi\!\downarrow$, then $\mathrm{RES}[\phi, \mathrm{KB}]$ is
   $((x = n_1) \wedge \mathrm{RES}[\phi^x_{n_1}, \mathrm{KB}]) \vee \ldots$
   $((x = n_k) \wedge \mathrm{RES}[\phi^x_{n_k}, \mathrm{KB}]) \vee$
   $((x \neq n_1) \wedge \ldots \wedge (x \neq n_k) \wedge \mathrm{RES}[\phi^x_{n'}, \mathrm{KB}]^{n'}_x)$.

If we consider our example KB, then $\mathrm{RES}[\mathit{Letter}(x, now)]$ reduces (after simplification) to $(x = C) \vee (x = D)$ whereas $\mathrm{RES}[\mathit{Red}(x, now)]$ reduces to FALSE because there are no known red things. The next definition applies RES to all occurrences of `Knows` within a query using a recursive descent denoted by $\| \cdot \|_{\mathrm{KB}}$. The idea is that any occurrence of $\mathtt{Knows}(\alpha, now)$ in a query is replaced by an equality expression describing the known instances of α.

Definition 4.2:

Given a KB as defined above and an arbitrary query α, $\|\alpha\|_{\mathrm{KB}}$ is the objective formula simple in *now* defined by

$\|\alpha\|_{\mathrm{KB}} = \alpha$, when α is objective;
$\|\neg\alpha\|_{\mathrm{KB}} = \neg\|\alpha\|_{\mathrm{KB}}$;
$\|(\alpha \wedge \beta)\|_{\mathrm{KB}} = (\|\alpha\|_{\mathrm{KB}} \wedge \|\beta\|_{\mathrm{KB}})$;
$\|\forall x \alpha\|_{\mathrm{KB}} = \forall x \|\alpha\|_{\mathrm{KB}}$;
$\|\mathtt{Knows}(\alpha, \mathtt{now})\|_{\mathrm{KB}} = \mathrm{RES}[\|\alpha\|_{\mathrm{KB}}, \mathrm{KB}]$.

Theorem 4.3: *Let* KB *be a knowledge base at* $\vec{a}$ *with current state axioms* KB_{cur}. *Then* $\mathrm{ASK}[\alpha, \Re[\![\mathrm{KB}]\!], \vec{a}] =$ *yes iff*

$$\mathrm{KB}_{cur}\!\downarrow \cup S_{\mathrm{U}} \models_{\text{FOL}} \|\alpha\|_{\mathrm{KB}}\!\downarrow .$$

In essence, the theorem says that answering an epistemic query can be achieved by computing a finite number of first-order implications. Restricting ourselves to queries in $\mathcal{IL}$ is essential in this case.

To illustrate what this theorem says consider the example KB and the query $\alpha = \exists x \mathit{Red}(x, now) \wedge \neg\mathtt{Knows}(\mathit{Red}(\mathtt{x}, now))$. Then $\mathrm{ASK}[\alpha, \Re[\![\mathrm{KB}]\!], now] =$ *yes* because of the following: $\mathrm{RES}[\mathit{Red}(x, now), \mathrm{KB}]$ simplifies to FALSE because there are no known instances of red objects. Hence $\|\alpha\|_{\mathrm{KB}}\!\downarrow$ is equivalent to $\exists x \mathit{Red}(x) \wedge \neg\text{FALSE}$ and, furthermore, $\mathrm{KB}_{cur}\!\downarrow \cup S_{\mathrm{U}} \models_{\text{FOL}} \exists x \mathit{Red}(x)$.

Being able to reduce query evaluation in $\mathcal{AOL}$ to first-order reasoning under certain restrictions is somewhat analogous to a result by Lin and Reiter [13] for the standard (non-epistemic) situation calculus. They show that, even though their foundational axioms for the situation calculus include a second-order axiom to characterize the set of all situations, this axiom is not needed when doing temporal projection, that is, when inferring whether a formula ϕ simple in $do(\vec{a}, S_0)$ follows from the domain theory together with the foundational axioms. There are also other examples such as [11] which show that theories which are inherently second-order nevertheless have interesting special cases where first-order reasoning alone suffices.

4.3 Context-Free Knowledge Bases

Lin and Reiter showed that in their framework, progression is not always first-order definable. We conjecture that the same is true in $\mathcal{AOL}$, but just as in LR's case there are interesting classes of knowledge bases which are not only first-order representable but where progression is also easily computable. LR discuss in particular the classes they call *relatively-complete* and *context-free* action theories. Here we adapt and extend context-free action theories for $\mathcal{AOL}$ and obtain very similar results. (The same is true for relatively complete theories, but we omit them for space reasons.)

A fluent F is called *situation independent* if its successor state axiom has the form $\forall s \forall a \forall \vec{x}. now \preceq s \supset [\mathit{Poss}(a, s) \supset [F(\vec{x}, do(a, s)) \equiv F(\vec{x}, s)]]$, that is, F never changes. Otherwise F is called situation dependent. A formula is called situation independent if it contains only situation independent fluents.

Definition 4.4: [Lin and Reiter] A KB is *context-free* if

- KB_{ss} consists of successor state axioms of the form $\forall s \forall a \forall \vec{x}. now \preceq s \supset [\mathit{Poss}(a, s) \supset [F(\vec{x}, do(a, s)) \equiv \gamma^+_F(\vec{x}, a, s) \vee (F(\vec{x}, s) \wedge \neg\gamma^-_F(\vec{x}, a, s))]]$, where $\gamma^+_F(\vec{x}, a, s)$ and $\gamma^-_F(\vec{x}, a, s)$ are situation independent.[9]

[9] The idea is that γ^+_F describes the conditions which cause F to be true and γ^-_F those which cause it to be false.

- KB_{cur} consists of situation independent formulas and formulas of the form $\forall\vec{x}.\psi \supset F(\vec{x}, do(\vec{a}, now))$ or $\forall\vec{x}.\psi \supset \neg F(\vec{x}, do(\vec{a}, now))$, where ψ is a situation independent formula with free variables in $\vec{x}$ and *now*.
- For every action precondition axiom $\forall s\forall\vec{x}.now \preceq s \supset [Poss(A(\vec{x}), s) \equiv \phi(\vec{x}, s)]$, $\phi(\vec{x}, s)$ is situation independent.
- For every sensed fluent axiom $\forall s\forall\vec{x}.now \preceq s \supset [SF(A(\vec{x}), s) \equiv \psi(\vec{x}, s)]$, $\psi(\vec{x}, s)$ is situation independent.

The conditions on the sensed fluent and action precondition axioms are missing in LR's definition because they do not deal with sensing and they do not consider the case where an agent successfully performs an action even though she does not know that it is possible. In a sense, finding out that an action is possible by doing it can be thought of as a special form of sensing. Note also that *SF* and *Poss* are treated completely symmetricly in our semantic definition of progression.

Definition 4.5: Let $\mathrm{KB} = \mathrm{KB}_{cur} \cup \mathrm{KB}_{Poss} \cup \mathrm{KB}_{SF} \cup \mathrm{KB}_{ss}$ be a context-free knowledge base at $\vec{a}$, w_0 a world, $A = Act(\vec{n})$ an action, and let $s_1 = do(\vec{a}, now)$ and $s_2 = do(\vec{a} \cdot A, now)$. Let the action precondition and sensed fluent axioms for A be

$$\forall s\forall\vec{x}.now \preceq s \supset [Poss(Act(\vec{x}), s) \equiv \phi_A(\vec{x}, s)] \text{ and}$$
$$\forall s\forall\vec{x}.now \preceq s \supset [SF(Act(\vec{x}), s) \equiv \psi_A(\vec{x}, s)].$$

Then let $\mathrm{KB}^A = \mathrm{KB}^A_{cur} \cup \mathrm{KB}_{Poss} \cup \mathrm{KB}_{SF} \cup \mathrm{KB}_{ss}$, where $\mathrm{KB}^A_{cur}$is constructed as follows:

1. Let A be a sensing action. Then:
   - If $\phi \in \mathrm{KB}_{cur}$ then $\phi^{s_1}_{s_2} \in \mathrm{KB}^A_{cur}$;
   - if $w_0 \models Poss(A, do(\vec{a}, S_0))$ then $\phi_A(\vec{n}, s_2) \in \mathrm{KB}^A_{cur}$ else $\neg\phi_A(\vec{n}, s_2) \in \mathrm{KB}^A_{cur}$;
   - if $w_0 \models SF(A, do(\vec{a}, S_0))$ then $\psi_A(\vec{n}, s_2) \in \mathrm{KB}^A_{cur}$ else $\neg\psi_A(\vec{n}, s_2) \in \mathrm{KB}^A_{cur}$.
2. Let A be an ordinary action. Then:
   - If $\phi \in \mathrm{KB}_{cur}$ is sit. independent, then $\phi^{s_1}_{s_2} \in \mathrm{KB}^A_{cur}$;
   - for any situation dependent fluent F add to KB^A_{cur} $\forall\vec{x}.\gamma^+_F(\vec{x}, A, s_2) \supset F(\vec{x}, s_2)$ and $\forall\vec{x}.\gamma^-_F(\vec{x}, A, s_2) \supset \neg F(\vec{x}, s_2)$;
   - If $\forall\vec{x}.\psi \supset F(\vec{x}, s_1)$ is in KB_{cur}, then add $\forall\vec{x}.\psi^{s_1}_{s_2} \wedge \neg\gamma^-_F(\vec{x}, A, s_2) \supset F(\vec{x}, s_2)$;
   - If $\forall\vec{x}.\psi \supset \neg F(\vec{x}, s_1)$ is in KB_{cur}, then add $\forall\vec{x}.\psi^{s_1}_{s_2} \wedge \neg\gamma^+_F(\vec{x}, A, s_2) \supset \neg F(\vec{x}, s_2))$;
   - if $w_0 \models Poss(A, do(\vec{a}, S_0))$ then $\phi_A(\vec{n}, s_2) \in \mathrm{KB}^A_{cur}$ else $\neg\phi_A(\vec{n}, s_2) \in \mathrm{KB}^A_{cur}$.

Note the different treatment depending on whether A is a sensing action or not. In the former case, the old contents of KB_{cur} is simply copied to the new knowledge base with the new situation s_2 replacing the old s_1. If A is an ordinary action, we need to treat the situation dependent fluents in KB_{cur} in a special way in order to reflect the changes that result from doing A. In the case of a sensing action we also need to record the values of ϕ_A and ψ_A depending on the truth value of $Poss(A)$ and $SF(A)$ at $(w_0, \vec{a})$. If A is an ordinary action, this needs to be done only for ϕ_A because we assume that ψ_A is equivalent to TRUE for ordinary actions.

It is not hard to see that the property of being context-free is preserved by our syntactic form of progression.

Lemma 4.6: *Let* KB, KB^A, *and* A *be as in Definition 4.5. Then* KB^A *is context-free.*

In their paper [13], LR describe some very simple (and reasonable) consistency requirements for context-free knowledge bases.[10] We will not repeat those conditions here and simply refer to them as LR-consistency. We are now ready to show that syntactic progression of context-free KB's conforms with our semantic definition.

Theorem 4.7: *Let* KB_0 *be an initial knowledge base,* w_0 *a world and* $e_0 = \Re[\![\mathrm{KB}_0]\!]$. *Let* KB, KB^A *and* A *be as in Definition 4.5 such that* $\Re[\![\mathrm{KB}]\!]$ *is a progression at* $\vec{a}$ *wrt* $\langle e_0, w_0\rangle$.

If KB *is LR-consistent, then* $\Re[\![\mathrm{KB}^A]\!]$ *is a progression of* $\Re[\![\mathrm{KB}]\!]$ *at* $\vec{a}$ *wrt* $\langle e_0, w_0\rangle$.

Note that, by definition, KB_0 is itself a progression at ϵ wrt $\langle e_0, w_0\rangle$. Hence, the theorem tells us that, starting in an initial context-free knowledge base, doing an action A will lead to a progression which itself is represented by a context-free knowledge base, and this process iterates.

To illustrate how progression works, let us consider the initial KB and the corresponding action model $M = \langle\Re[\![\mathrm{KB}]\!], w\rangle$ from Section 4.1. First, it is easy to verify that it conforms to the definition of a context-free KB.

1. Let us consider progressing KB by $A = \mathsf{senseRed}(C)$ resulting in KB^A with corresponding KB^A. Let s_1 stand for $do(\mathsf{senseRed}(C), now)$.
Since A is a sensing action (case (1) of Def. 4.5), we obtain KB^A_{cur} simply by replacing every occurrence of *now* in KB_{cur} by s_1 and adding $Red(C, s_1)$ to it, because we assume that $M \models SF(\mathsf{senseRed}(C), S_0)$. Then $\Re[\![\mathrm{KB}^A]\!]$ is a progression at A.
Let $\alpha = \exists x Red(x, now) \wedge \mathtt{Knows}(\mathtt{Red}(\mathtt{x}, now), now)$. Then $\mathrm{ASK}[\alpha, \Re[\![\mathrm{KB}^A]\!], A] =$ *yes* because now there is a known red letter, namely C.

2. Let us now progress KB^A by $A' = \mathsf{pickup}(C)$ resulting in $\mathrm{KB}^{AA'}$ with corresponding $\mathrm{KB}^{AA'}_{cur}$. Let s_2 stand for $do(\mathsf{pickup}(C), s_1)$.

Starting with the empty set we construct $\mathrm{KB}^{AA'}_{cur}$ by adding the following sentences:[11]

- $Letter(C, s_2), Letter(D, s_2), Red(C, s_2)$

(The disjunction $(Red(C, s_2) \vee Red(D, s_2))$ is omitted because it is clearly subsumed by $Red(C, s_2)$.)

Given the successor state axiom for $F = HoldRLs$, we obtain

$\gamma^-_F = \text{FALSE}$ and
$\gamma^+_F(x, A, s_2) = [pickup(C) = pickup(x) \wedge Red(x, s_2)]$.

[10] One such requirement is that γ^+_F and γ^-_F may never be true simultaneously. The example KB is LR-consistent.

[11] For simplicity, we omit adding sentences that turn out to be valid or subsumed by others.

Hence we add

- $\forall x.\gamma_F^+(x, A, s_2) \supset \mathit{HoldRLs}(x, s_2)$

Finally, the last case of Definition 4.5 applies and we add

- $\forall x.\neg\gamma_F^+(x, A, s_2) \supset \neg\mathit{HoldRLs}(x, s_2)$

Given the unique names assumption for standard names of objects and actions, $\gamma_F^+(x, A, s_2)$ is true just in case $x = C$, that is, the agent is holding precisely C in s_2.

Given this progressed knowledge base it is then not hard to show that the robot does not know in s_2 whether it is holding all the red letters. Formally, let

$$\alpha = \forall x.\mathit{Red}(x, now) \wedge \mathit{Letter}(x, now) \supset \mathtt{Knows}(\mathit{Red}(\mathtt{x}, now) \wedge \mathit{Letter}(\mathtt{x}, now), now).$$

Then $\mathrm{ASK}[\alpha, \Re[\![\mathrm{KB}^{AA'}]\!], A \cdot A'] = \mathit{unknown}$. This is because there are worlds in $\Re[\![\mathrm{KB}^{AA'}]\!]$ where C is the only red letter and others where there are red letters other than C after doing $A \cdot A'$.

5 Conclusions

Using the second-order logic $\mathcal{AOL}$, we specified a query facility for knowledge bases in dynamic worlds. Despite the expressiveness of the logic, we showed that query evaluation often requires only first-order reasoning. Moreover, by adapting and extending results by Lin and Reiter, we gave a semantic definition of progression and showed that it is first-order representable in the case of context-free knowledge bases.

Future work includes finding more powerful classes of knowledge bases with first-order progressions and applying the results to the action programming language GOLOG [10]. We defined progression in a way that is very close to the original definition by Lin and Reiter. The exact relationship between the two still needs to be determined. Also, our earlier definition of SUCC can be thought of as a progression operator in its own right. It is more powerful in that nothing about the past is forgotten. It is an interesting open problem to determine syntactic variants of this notion of progression.

References

[1] Hintikka, J., *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, 1962.

[2] Kaplan, D., Quantifying In, in L. Linsky (ed.), *Reference and Modality*, Oxford University Press, Oxford, 1971.

[3] Kripke, S. A., Semantical considerations on modal logic. *Acta Philosophica Fennica* **16**, 1963, pp. 83–94.

[4] Lakemeyer, G. and Levesque, H. J. $\mathcal{AOL}$: a logic of acting, sensing, knowing, and only knowing. *Proc. of the Sixth International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, San Francisco, 1998.

[5] Lakemeyer, G. and Levesque, H. J. Querying $\mathcal{AOL}$ Knowledge Bases. Preliminary Report. Festschrift in Honour of W. Bibel, Kluwer Academic Press, to appear.

[6] Levesque, H. J., Foundations of a Functional Approach to Knowledge Representation, *Artificial Intelligence*, **23**, 1984, pp. 155-212.

[7] Levesque, H. J., All I Know: A Study in Autoepistemic Logic. *Artificial Intelligence*, North Holland, **42**, 1990, pp. 263–309.

[8] Levesque, H. J., What is Planning in the Presence of Sensing. AAAI-96, AAAI Press, 1996.

[9] Levesque, H. J. and Lakemeyer, G., *The Logic of Knowledge Bases*, Monograph, forthcoming.

[10] Levesque, H. J., Reiter, R., Lespérance, Y., Lin,F. and Scherl., R. B., GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, **31**, 59-84, 1997.

[11] Lifschitz, V., Computing Circumscription, *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Francisco, 1985, pp. 121–127.

[12] Lin, F. and Reiter, R., Forget It!, in *Proc. of the AAAI Fall Symposium on Relevance*, New Orleans, 1994, pp. 154–159.

[13] Lin, F. and Reiter, R., How to Progress a Database. *Artificial Intelligence*, **92**, 1997, pp.131-167.

[14] McCarthy, J., *Situations, Actions and Causal Laws.* Technical Report, Stanford University, 1963. Also in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, 1968, pp. 410–417.

[15] Moore, R. C., A Formal Theory of Knowledge and Action. In J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985, pp. 319–358.

[16] Reiter, R., The Frame Problem in the Situation Calculus: A simple Solution (sometimes) and a Completeness Result for Goal Regression. In V. Lifschitz (ed.), *Artificial Intelligence and Mathematical Theory of Computation*, Academic Press, 1991, pp. 359–380.

[17] Scherl, R. and Levesque, H. J., The Frame Problem and Knowledge Producing Actions. in *Proc. of the National Conference on Artificial Intelligence (AAAI-93)*, AAAI Press, 1993, 689–695.