Foundations for Knowledge-Based Programs using $\mathcal{ES}$

Jens Claßen and **Gerhard Lakemeyer**
Department of Computer Science
RWTH Aachen
52056 Aachen
Germany
⟨classen|gerhard⟩@cs.rwth-aachen.de

Abstract

Reiter proposed a semantics for knowledge-based Golog programs with sensing where program execution can be conditioned on tests involving explicit references to what the agent knows and does not know. An important result of this work is that reasoning about knowledge after the execution of actions can be reduced to classical reasoning from an initial first-order theory. However, it is limited in that tests can only refer to what is known about the current state, knowledge about knowledge is not considered, and the reduction does not apply to formulas with quantifying-in. This is in large part due to the choice of the underlying formalism, which is Reiter's version of the situation calculus. In this paper we show that, by moving to a new situation calculus recently proposed by Lakemeyer and Levesque, we cannot only reconstruct Reiter's foundations for knowledge-based programs but we can significantly go beyond them, which includes removing the above restrictions and more.

Introduction

In the action programming language Golog (Levesque *et al.* 1997), whose semantics is based on Reiter's variant of the situation calculus (McCarthy & Hayes 1969; Reiter 2001b), program constructs such as if-then-else or while statements use tests which are sentences in first-order logic and refer to what is currently true, that is, after execution of a sequence of primitive actions which led to this point in the program. Reiter (2001a) observed that it is useful to extend these tests and allow them to refer explicitly to the agent's epistemic state, when the agent has incomplete information about the world. He formalized this idea based on Scherl and Levesque's (2003) extension of the situation calculus to account for knowledge. A central result of this work was that the evaluation of tests involving knowledge can often be reduced to classical first-order reasoning about the initial state. Reiter also considers sensing actions and shows how to integrate the result of such actions into the description of the initial state using ideas similar to (Giacomo & Levesque 1999).

Nevertheless, the work has a number of limitations. For one, Reiter has to assume that knowledge is always true, which may not be realistic in many applications. Perhaps more importantly, the language in which tests are formulated is very restricted despite the ability to explicitly mention the agent's knowledge. In particular, tests can only refer to what is known about the current state, knowledge about knowledge is not considered, and the reduction does not apply to formulas with quantifying-in, which are needed to distinguish between "knowing that" and "knowing what." To illustrate why such features are useful, let us consider an example involving a robot in an environment consisting of rooms and boxes, as in Shakey's world.

Imagine that the robot has only two actions at its disposal: $push(x, y, z)$, allowing it to push some box x from room y to room z, and $lookFor(x, y)$, which is a sensing action that provides the information of whether or not box x is currently in room y (by using cameras or other means). For simplicity, we completely abstract from the robot's position, i.e. there is no $goto$ action. We simply assume that performing either of the two actions contains the subtask of first getting to room y and then do the actual pushing or looking, respectively. An example program fragment for our robot could then be the following:

```
if $\neg\exists y.\mathit{Know}(At(box1, y)) \wedge$
$\mathit{Know}([lookFor(box1, room1)]\exists y.\mathit{Know}(At(box1, y)))$
  then $lookFor(box1, room1)$;
if $\mathit{Know}(\mathit{Poss}(push(box1, room1, room2)))$
  then $push(box1, room1, room2)$;
```

That is, if the location of $box1$ is currently not known, yet it is known that after looking for $box1$ in $room1$ the location of $box1$ will be known, then do it. Afterwards, push the box from $room1$ to $room2$, if possible.

Suppose that the robot initially does not know where $box1$ is located; it only knows that it has to be in one of $room1$ and $room2$. Then it is true that after looking for $box1$ in $room1$, its location will be known: If $lookFor(box1, room1)$ returns "TRUE", then the box is in $room1$; if the sensing result is "FALSE", it follows from the initial knowledge that it has to be in $room2$.[1] Therefore, the robot executes the looking

[1]We make the assumption that the sensing action does not have any side effects, i.e. performing it does not change the state of the world, but only the agent's mental state.

action and either learns that $box1$ actually is in $room1$ or that it is not and thus has to be in $room2$. Assuming that pushing box x from y to z is possible whenever x is currently located at y, then whether or not the robot afterwards pushes $box1$ from $room1$ to $room2$ depends on the outcome of the sensing action.

Note the use of knowledge about what is known after an action has occurred in the example, together with quantifying-in ("knowing what"), which cannot be expressed in Reiter's approach. There are at least two reasons for this: for one, tests for Reiter are situation-suppressed formulas and hence cannot refer to future situations; for another, in order to make inferences about what the agent *does not* know, a meta-theoretic knowledge closure assumption is made,[2] which is somewhat awkward and makes a principled account of meta knowledge with quantifying-in difficult, if not impossible.

In this paper we develop a new foundation for knowledge-based programs by using the recently proposed variant of the situation calculus called $\mathcal{ES}$ (Lakemeyer & Levesque 2004). It does not use situation terms in the language and hence can directly serve as the language for tests. In fact, the tests in the above example are formulas in $\mathcal{ES}$. Moreover, $\mathcal{ES}$ has a built-in notion of "this is all I know" adapted from Levesque's logic of only-knowing (Levesque & Lakemeyer 2001), which naturally models meta knowledge with quantifying-in and allows for false beliefs. The main results are the following: for the evaluation of tests, we define an ASK routine to query the agent's knowledge base and show that this always reduces to non-modal, first-order reasoning, leaning on results obtained in (Lakemeyer & Levesque 2004); for the online execution of sensing actions, we define an EXE routine and show that, analogous to Reiter, sensing results can always be incorporated into the knowledge base using regression; in addition, we define a TELL routine, which allows users to give information to the agent after any number of actions have been executed, and we show that this information can also be added to the knowledge base using regression. Both user interaction with the knowledge-base and the execution of knowledge-based programs can thus be reduced to first-order reasoning. The interaction language itself is more expressive than Reiter's as it allows references to future situations, meta knowledge and "knowing what."

Knowledge and action has been a concern for a long time in AI, starting perhaps with Moore's work (1977), which was later refined for the situation calculus (Scherl & Levesque 2003) and its close relative, the fluent calculus (Jin & Thielscher 2004). Knowledge has also been incorporated into other action languages like $\mathcal{A}$ (Lobo, Mendez, & Taylor 1997) and planning formalisms (Petrick & Bacchus 2002). Besides Reiter, the work reported here is also inspired by (Lakemeyer & Levesque 1999), which defines querying knowledge bases in the logic $\mathcal{AOL}$ (Lakemeyer & Levesque 1998). An important difference is that there reasoning relied on progressing a knowledge base after every action, while we pursue a regression-based approach. Also, while $\mathcal{AOL}$ does incorporate a notion of only-knowing, it turned out to be hard to work with, in contrast to $\mathcal{ES}$.

The rest of the paper is organized as follows. First we introduce the syntax and semantics of $\mathcal{ES}$, followed by a discussion of basic action theories in $\mathcal{ES}$, which enable the use of regression and the reduction of reasoning about knowledge to first-order reasoning. Then we give semantic definitions of ASK,TELL, and EXE, followed by representation theorems showing that they reduce to reasoning or knowledge-base updates involving only non-modal first-order sentences. The final section concludes the paper.

The Logic $\mathcal{ES}$

The language

The language consists of formulas over symbols from the following vocabulary:

- variables $V = \{x_1, x_2, \ldots, y_1, y_2, \ldots, a_1, a_2, \ldots\}$;
- fluent predicates of arity k: $F^k = \{F_1^k, F_2^k, \ldots\}$; for example, *At*; we assume this list includes the distinguished predicates *Poss* and *SF*;
- rigid functions of arity k: $G^k = \{g_1^k, g_2^k, \ldots\}$; for example, $box2$, $push$; G^0 is also referred to as the set of standard names;
- connectives and other symbols: $=$, $\wedge$, $\neg$, $\forall$, *Know*, *OKnow*, $\square$, round and square parentheses, period, comma.

To keep the formalism simple, we do not include rigid (non-fluent) predicates or fluent (non-rigid) functions. The special predicate *Poss* is used to define action preconditions and *SF* (for sensed fluent value) is used to define what the outcome of a sensing action is. We will see examples for both below. The *terms* of the language are the least set of expressions such that

1. Every first-order variable is a term;
2. If $t_1, \ldots, t_k$ are terms, then so is $g^k(t_1, \ldots, t_k)$.

We let R denote the set of all rigid terms (here, all ground terms). For simplicity, instead of having variables of the *action* sort distinct from those of the *object* sort as in the situation calculus, we lump both of these together and allow ourselves to use any term as an action or as an object.[3] (See (Lakemeyer & Levesque 2005) for a version of $\mathcal{ES}$ with different sorts for objects and actions, as well as rigid predicates and fluent functions.)
Finally, the *well-formed formulas* of the language form the least set such that

1. If $t_1, \ldots, t_k$ are terms, then $F^k(t_1, \ldots, t_k)$ is an (atomic) formula;
2. If t_1 and t_2 are terms, then $(t_1 = t_2)$ is a formula;
3. If t is a term and α is a formula, then $[t]\alpha$ is a formula;
4. If α and β are formulas, then so are $(\alpha \wedge \beta)$, $\neg\alpha$, $\forall x.\alpha$, $\square\alpha$, *Know*(α), *OKnow*(α).

[2]Roughly, this adds formulas of the form $\neg$*Know*(ϕ) to a theory for those ϕ which do not follow from the initial theory.

[3]Equivalently, the version in this paper can be thought of as having action terms but no object terms.

We read $[t]\alpha$ as "α holds after action t", $\Box\alpha$ as "α holds after any sequence of actions," $\mathit{Know}(\alpha)$ as "α is known", and $\mathit{OKnow}(\alpha)$ as "α is all that is known" or "the agent only-knows α." As usual, we treat $\exists x.\alpha$, $(\alpha \vee \beta)$, $(\alpha \supset \beta)$, and $(\alpha \equiv \beta)$ as abbreviations. We call a formula without free variables a *sentence*.

We use the notation α^x_t to mean the result of simultaneously replacing all free occurrences of the variable x by the term t. For convenience, we use the notation $[\sigma]\alpha$ for any sequence of actions σ. It is defined inductively as follows:

1. $[\langle\,\rangle]\alpha \overset{def}{=} \alpha$
2. $[r \cdot \sigma]\alpha \overset{def}{=} [r][\sigma]\alpha$

In the following, we will sometimes refer to subsets of the language and use the following terminology:

A formula without	*is called*
Know or *OKnow* operators	objective
OKnow operators	basic
fluents, $\Box$, or $[t]$ outside the scope of *Know* or *OKnow*	subjective
$\Box$ operators	bounded
$\Box$ or $[t]$ operators	static
Know, *OKnow*, $\Box$, $[t]$, *Poss*, *SF*	fluent

While an objective formula only refers to what is true in the actual world, subjective formulas only refer to what is known or believed about the world. Bounded formulas may only refer to what is true after a finite number of actions have been performed and static formulas can only talk about what is currently true. Fluent formulas, which are objective and static, correspond to objective situation-suppressed formulas in Reiter's situation calculus.

The semantics

Intuitively, a world w will determine which fluents are true, but not just initially, also after any sequence of actions. Let P denote the set of all pairs $\sigma{:}\rho$ where $\sigma \in R^*$ is considered a sequence of actions, and $\rho = F(r_1, \ldots, r_k)$ is a ground fluent atom. In general, formulas are interpreted relative to a model $M = \langle e, w\rangle$ where $e \subseteq W$ and $w \in W$, and where $W = [P \to \{0, 1\}]$. The e determines all the agent knows initially, and is referred to as the agent's *epistemic state.*

First-order variables are interpreted substitutionally over the rigid terms R, that is, R is treated as being isomorphic to a fixed universe of discourse. This is similar to $\mathcal{OL}$ (Levesque & Lakemeyer 2001), where standard names are used as the domain. In order to define what an agent knows or only-knows, we need the notion of two worlds w and w' agreeing on the sensing results, that is, the values of *SF*, with respect to a sequence of actions σ. This is denoted by $w' \simeq_\sigma w$ and defined inductively by the following:

1. when $\sigma = \langle\,\rangle$, $w' \simeq_\sigma w$, for every w' and w;
2. $w' \simeq_{\sigma \cdot r} w$ iff
   $w' \simeq_\sigma w$ and $w'[\sigma{:}\mathit{SF}(r)] = w[\sigma{:}\mathit{SF}(r)]$.

Here is the complete semantic definition: Given a model $M = \langle e, w\rangle$, for any formula α with no free variables, we define $e, w \models \alpha$ as $e, w, \langle\,\rangle \models \alpha$ where

$e, w, \sigma \models F(r_1, \ldots, r_k)$ iff $w[\sigma{:}F(r_1, \ldots, r_k)] = 1$;
$e, w, \sigma \models (r_1 = r_2)$ iff r_1 and r_2 are identical;
$e, w, \sigma \models (\alpha \wedge \beta)$ iff $e, w, \sigma \models \alpha$ and $e, w, \sigma \models \beta$;
$e, w, \sigma \models \neg\alpha$ iff $e, w, \sigma \not\models \alpha$;
$e, w, \sigma \models \forall x.\, \alpha$ iff $e, w, \sigma \models \alpha^x_r$, for every $r \in R$;
$e, w, \sigma \models [r]\alpha$ iff $e, w, \sigma \cdot r \models \alpha$;
$e, w, \sigma \models \Box\alpha$ iff $e, w, \sigma \cdot \sigma' \models \alpha$, for every $\sigma' \in R^*$;
$e, w, \sigma \models \mathit{Know}(\alpha)$ iff
  for all $w' \simeq_\sigma w$, if $w' \in e$ then $e, w', \sigma \models \alpha$;
$e, w, \sigma \models \mathit{OKnow}(\alpha)$ iff
  for all $w' \simeq_\sigma w$, $w' \in e$ iff $e, w', \sigma \models \alpha$.

When α is a sentence, we write $e, w \models \alpha$. When α is *objective*, we write $w \models \alpha$; when α is *subjective*, we write $e \models \alpha$. When Σ is a set of sentences and α is a sentence, we write $\Sigma \models \alpha$ (read: Σ logically entails α) to mean that for every e and w, if $e, w \models \alpha'$ for every $\alpha' \in \Sigma$, then $e, w \models \alpha$. Finally, we write $\models \alpha$ (read: α is valid) to mean $\{\} \models \alpha$.

Notice that the rules for knowing and only-knowing only consider worlds in e which agree with w on the sensing for σ. It is not hard to see that *Know* is just a *weak S5* operator (Chellas 1980). Note also that *OKnow* differs from *Know* only in that the "then" is replaced by an "iff." This has the effect that e has to be maximal for a sentence to be only-known, which is perhaps best seen when considering a fluent formula before any action has occurred. For example, $e \models \mathit{OKnow}(At(box1, room2))$ iff

$$e = \{w \mid w \models At(box1, room2)\}.$$

In other words, as far as objective sentences are concerned, e knows only the logical consequences of $At(box1, room2)$ and nothing else.

Basic Action Theories

As shown in (Lakemeyer & Levesque 2004), we are able to define basic action theories in a way very similar to those originally introduced by Reiter:

Definition 1 (Basic Action Theory) *Given a set of fluent predicates $\mathcal{F}$, a set of sentences Σ is called a* basic action theory *over $\mathcal{F}$ iff it only mentions the fluents in $\mathcal{F}$ and is of the form $\Sigma = \Sigma_0 \cup \Sigma_{pre} \cup \Sigma_{post} \cup \Sigma_{sense}$, where*

- *Σ_0 is a finite set of fluent sentences,*
- *Σ_{pre} is a singleton of the form[4] $\Box$ $\mathit{Poss}(a) \equiv \pi$, where π is fluent with a being the only free variable;*
- *Σ_{post} is a finite set of successor state axioms of the form[5] $\Box$ $[a]F(\vec{x}) \equiv \gamma_F$, one for each fluent $F \in \mathcal{F} \setminus \{\mathit{Poss}, \mathit{SF}\}$, where γ_F is a fluent sentence whose free variables are among $\vec{x}$ and a;*
- *Σ_{sense} is a singleton of the form $\Box$ $\mathit{SF}(a) \equiv \varphi$, where φ is fluent with a being the only free variable.*

[4]We follow the convention that free variables are universally quantified from the outside. We also assume that $\Box$ has lower syntactic precedence than the logical connectives, so that $\Box$ $\mathit{Poss}(a) \equiv \pi$ stands for $\forall a.\Box(\mathit{Poss}(a) \equiv \pi)$.

[5]The $[t]$ construct has higher precedence than the logical connectives. So $\Box$ $[a]F(\vec{x}) \equiv \gamma_F$ abbreviates $\forall a.\Box([a]F(\vec{x}) \equiv \gamma_F)$.

The idea is that Σ_0 represents the initial database, Σ_{pre} is one large precondition axiom, Σ_{post} the set of successor state axioms for all fluents in $\mathcal{F}$ (incorporating Reiter's solution (1991) to the frame problem) and Σ_{sense} defines the sensing results for actions. In our example, we imagine having at least one box and two rooms, and the box is in one of the rooms. The initial database then is:

$$\begin{array}{l} Box(box1) \wedge Room(room1) \wedge Room(room2), \\ At(box1, room1) \vee At(box1, room2) \end{array} \tag{1}$$

Further we need the following axioms expressing state constraints for the initial situation. By the successor state axioms, they will hold in all *Poss*ible future situations.

$$\begin{array}{lcl} At(x,y) & \supset & Box(x) \wedge Room(y) \wedge \\ & & \neg(\exists z.\ At(x,z) \wedge y \neq z), \\ Box(x) & \supset & \exists y.\ At(x,y) \end{array} \tag{2}$$

It is possible to push a box x from y to z iff x is in y and z is a room; the looking action is always possible. The precondition axiom therefore is:

$$\begin{array}{lcl} \Box\, Poss(a) & \equiv & \exists x,y,z.\ a = push(x,y,z) \wedge \\ & & At(x,y) \wedge Room(z) \vee \\ & & \exists x,y.\ a = lookFor(x,y) \end{array}$$

The fluents Box and $Room$ are situation-independent, i.e. their truth value does not change by doing actions. After some action a, x is located at y iff it was pushed there or if it was already located at y and not pushed somewhere else. We thus get the following successor state axioms:

$$\begin{array}{lcl} \Box\, [a]Box(x) & \equiv & Box(x), \\ \Box\, [a]Room(x) & \equiv & Room(x), \\ \Box\, [a]At(x,y) & \equiv & \exists z.\ a = push(x,z,y) \vee At(x,y) \wedge \\ & & \neg(\exists z.\ a = push(x,y,z) \wedge y \neq z) \end{array}$$

We assume that pushing always returns TRUE as a default sensing result. Looking, on the other hand, returns the truth value of the fluent At:

$$\begin{array}{lcl} \Box\, SF(a) & \equiv & \exists x,y,z.\ a = push(x,y,z) \vee \\ & & \exists x,y.\ a = lookFor(x,y) \wedge At(x,y) \end{array} \tag{3}$$

Regression

For basic action theories, (Lakemeyer & Levesque 2004) introduce an $\mathcal{ES}$ equivalent of Reiter's regression operator. The idea behind regression is that whenever we encounter a subformula of the form $[a]F(\vec{x})$ (i.e. $F(\vec{x})$ is true after a), we may substitute it by γ_F, the right-hand side of the successor state axiom of F. This is sound in the sense that the axiom defines the two expressions to be equivalent. The result of the substitution will be true in exactly the same worlds satisfying the action theory Σ as the original one, but contains one less modal operator $[a]$. Iteratively applying such substitution steps, we will end up with a static formula that describes exactly the conditions on the initial situation under which the original, non-static formula holds.

Formally, for any bounded, objective sentence α, let $\mathcal{R}[\alpha]$, *the regression of* α *wrt* Σ, be the fluent formula $\mathcal{R}[\langle\,\rangle, \alpha]$, where for any sequence of terms σ (not necessarily ground), $\mathcal{R}[\sigma, \alpha]$ is defined inductively on α by:

1. $\mathcal{R}[\sigma, (t_1 = t_2)] = (t_1 = t_2)$;
2. $\mathcal{R}[\sigma, \neg\alpha] = \neg\mathcal{R}[\sigma, \alpha]$;
3. $\mathcal{R}[\sigma, (\alpha \wedge \beta)] = (\mathcal{R}[\sigma, \alpha] \wedge \mathcal{R}[\sigma, \beta])$;
4. $\mathcal{R}[\sigma, \forall x\alpha] = \forall x\mathcal{R}[\sigma, \alpha]$;
5. $\mathcal{R}[\sigma, [t]\alpha] = \mathcal{R}[\sigma \cdot t, \alpha]$;
6. $\mathcal{R}[\sigma, Poss(t)] = \mathcal{R}[\sigma, \pi^a_t]$;
7. $\mathcal{R}[\sigma, SF(t)] = \mathcal{R}[\sigma, \varphi^a_t]$;
8. $\mathcal{R}[\sigma, F(t_1, \ldots, t_k)]$ is defined inductively on σ by:
   (a) $\mathcal{R}[\langle\,\rangle, F(t_1, \ldots, t_k)] = F(t_1, \ldots, t_k))$;
   (b) $\mathcal{R}[\sigma \cdot t, F(t_1, \ldots, t_k)] = \mathcal{R}[\sigma, (\gamma_F)^{a\,y_1}_{t\,t_1} \cdots {}^{y_k}_{t_k}]$.

For illustration, consider the sentence

$$[push(box1, room1, room2)]At(box1, room2),$$

which says that after pushing $box1$ from $room1$ to $room2$, the box is located in $room2$. We abbreviate $push(box1, room1, room2)$ by p. Then, given our example action theory and using Rule 5,

$$\mathcal{R}[\langle\,\rangle, [p]At(box1, room2)]$$

can be replaced by

$$\mathcal{R}[\langle p\rangle, At(box1, room2)].$$

Applying Rule 8b with the successor state axiom for At and substituting a by p, x by $box1$, and y by $room2$ yields

$$\begin{array}{l} \mathcal{R}[\langle\,\rangle, \exists z.p = push(box1, z, room2) \vee At(box1, room2) \\ \qquad \wedge\neg(\exists z.p = push(box1, room2, z) \wedge room2 \neq z)\ ]. \end{array}$$

Using Rules 1–4, 8a, and the definitions for $\vee$ and $\exists$ this reduces simply to

$$\begin{array}{l} \exists z.p = push(box1, z, room2) \vee At(box1, room2) \\ \wedge\neg(\exists z.p = push(box1, room2, z) \wedge room2 \neq z). \end{array} \tag{4}$$

Note that, since all ground terms are rigid designators, two terms $f(b)$ and $f(c)$ are equal just in case $b = c$. Hence, with p being shorthand for $push(box1, room1, room2)$, $p = push(box1, z, room2)$ holds iff

$$(box1 = box1) \wedge (room1 = z) \wedge (room2 = room2)$$

holds and $p = push(box1, room2, z)$ iff

$$(box1 = box1) \wedge (room1 = room2) \wedge (room2 = z).$$

Therefore (4) can be replaced by the equivalent sentence

$$\begin{array}{l} \exists z.(box1 = box1) \wedge (room1 = z) \wedge (room2 = room2) \\ \quad \vee At(box1, room2) \wedge \\ \quad \neg(\exists z.\ (box1 = box1) \wedge (room1 = room2) \wedge \\ \qquad\qquad (room2 = z) \wedge (room2 \neq z) \qquad), \end{array} \tag{5}$$

which simplifies to TRUE $\vee$ $At(box1, room2) \wedge$ TRUE and hence to TRUE. This tells us that, according to our action theory Σ, there are no special conditions under which $At(box1, room2)$ is true after p. It is simply the case (and corresponds to our intuition) that it is always true that after pushing $box1$ to $room2$, $box1$ is in $room2$, no matter what the world was like before. Whether it is possible to perform action p is a whole different question. The reader may verify that $\mathcal{R}[\mathit{Poss}(p)]$ is, with simplifications, equivalent to $At(box1, room1)$; p therefore is only executable in an initial world state where $box1$ actually is in $room1$, which again agrees with our intuition about the executability of that action.

In general, it is possible to transform an objective, bounded formula into a fluent one that is equivalent wrt to Σ. More precisely, Lakemeyer and Levesque obtain the following:

Theorem 2 *Let Σ be a basic action theory and let α be an objective, bounded sentence. Then $\mathcal{R}[\alpha]$ is a fluent sentence and satisfies*

$$\Sigma \models \alpha \quad \textit{iff} \quad \Sigma_0 \models \mathcal{R}[\alpha].$$

As an immediate consequence of the theorem we obtain

Corollary 3 *Let $w \models \Sigma$ and ϕ be bounded and objective. Then $\mathcal{R}[\phi]$ is a fluent sentence and $w \models \mathcal{R}[\phi]$ iff $w \models \phi$.*

Lakemeyer and Levesque also show how regression can be extended to cope with knowledge, based on the following theorem.

Theorem 4

$$\models \Box[a]\mathit{Know}(\phi) \equiv$$
$$\mathit{SF}(a) \land \mathit{Know}(\mathit{SF}(a) \supset [a]\phi) \lor$$
$$\neg \mathit{SF}(a) \land \mathit{Know}(\neg \mathit{SF}(a) \supset [a]\phi)$$

The theorem, which can be seen as a kind of successor state axiom for knowledge, leads to two additional regression rules for knowledge:

9. $\mathcal{R}[\sigma \cdot t, \mathit{Know}(\alpha)] = \mathcal{R}[\sigma, \beta^a_t]$,
   where β is the right-hand side of the equivalence in Theorem 4.
10. $\mathcal{R}[\langle\rangle, \mathit{Know}(\alpha)] = \mathit{Know}(\mathcal{R}[\langle\rangle, \alpha])$.

In (Lakemeyer & Levesque 2004) the rules were actually slightly more complicated as they used two different basic action theories, one which refers to what is true in the actual world and another which is believed by the agent. In this paper, we are only concerned with what the agent believes and never need to refer to a basic action theory other than the one held by the agent, hence the simplification. The following theorem is an easy consequence of Theorem 5 of (Lakemeyer & Levesque 2004).

Theorem 5 *Let Σ be a basic action theory and α be a bounded, basic sentence. Then*
$\models \mathit{OKnow}(\Sigma) \supset \mathit{Know}(\alpha)$ *iff*
$\models \mathit{OKnow}(\Sigma_0) \supset \mathit{Know}(\mathcal{R}[\langle\rangle, \alpha])$.

Reducing knowledge to first-order reasoning

When we apply the extended regression operator to a basic, bounded formula, the result is basic and static, i.e. we get a formula that only talks about the initial situation, but may still contain *Know* operators. We may however eliminate those as well using another result from (Lakemeyer & Levesque 2004). The idea is, given the description Σ_0 of the initial situation, to replace a subformula $\mathit{Know}(\beta)$ by an objective formula ϕ which describes the known instances of β. The result is an adaptation of the Representation Theorem of $\mathcal{OL}$ (Levesque & Lakemeyer 2001) and makes use of the fact that each valid $\mathcal{OL}$ sentence, as defined below, is also valid in $\mathcal{ES}$.

Definition 6 *An $\mathcal{OL}$* formula *is static, does not mention Poss or SF, and the only rigid terms appearing are standard names (i.e. terms from G^0) and variables.*

The basic building block of the construction now is given in the following definition.

Definition 7 *Let ϕ be an objective $\mathcal{OL}$ formula and Σ_0 be a finite set of objective $\mathcal{OL}$ sentences. Then $\mathrm{RES}[\![\phi, \Sigma_0]\!]$ is defined by:*

1. *If ϕ has no free variables, then $\mathrm{RES}[\![\phi, \Sigma_0]\!]$ is*
   TRUE, *if $\Sigma_0 \models \phi$, and* FALSE, *otherwise.*
2. *If x is a free variable in ϕ:*
   Let $n_1, \ldots, n_k$ be all of the standard names in ϕ and in Σ_0, and n' some name that does not appear in ϕ or in Σ_0. Then $\mathrm{RES}[\![\phi, \Sigma_0]\!]$ is
   $((x = n_1) \land \mathrm{RES}[\![\phi^x_{n_1}, \Sigma_0]\!]) \lor \cdots$
   $((x = n_k) \land \mathrm{RES}[\![\phi^x_{n_k}, \Sigma_0]\!]) \lor$
   $((x \neq n_1) \land \ldots \land (x \neq n_k) \land \mathrm{RES}[\![\phi^x_{n'}, \Sigma_0]\!]^{n'}_x)$

The intuition here is that to determine for which individuals ϕ is known to hold (according to Σ_0), we only have to check the (finitely many) names n_i in ϕ and Σ_0 and one further name n', which then serves as a representative for all individuals that are not mentioned. The method is correct since if we can prove that $\phi^x_{n'}$ is (not) known to hold, we can simply substitute n' by any non-appearing ground term r and use the same proof for showing that ϕ^x_r is also (not) known. This already suffices because our semantics assumes that the set of ground terms also represents the universe of discourse. Notice that after the recursive evaluation of RES with x replaced by n', all appearances of n' are substituted back by x. The new names are therefore only used temporarily for the purpose of the construction and do not appear in the final result.

In the example initial database of our robot, the only standard names are $box1$, $room1$ and $room2$. One arbitrary name that does not appear is $box17$. $\mathrm{RES}[\![Box(x), \Sigma_0]\!]$ then for instance is

$$((x = box1) \land \mathrm{RES}[\![Box(box1), \Sigma_0]\!]) \lor$$
$$((x = room1) \land \mathrm{RES}[\![Box(room1), \Sigma_0]\!]) \lor$$
$$((x = room2) \land \mathrm{RES}[\![Box(room2), \Sigma_0]\!]) \lor$$
$$((x \neq box1) \land (x \neq room1) \land (x \neq room2)$$
$$\land \mathrm{RES}[\![Box(box17), \Sigma_0]\!]^{box17}_x)$$

Since neither of $Box(room1)$, $Box(room2)$ and $Box(box17)$ is entailed by Σ_0, all of the corresponding RES subformulas are equal to FALSE. Only $\mathrm{RES}[\![Box(box1), \Sigma_0]\!]$ evaluates to TRUE, which is why the whole formula can be simplified to $(x = box1)$.

For the complete definition, we fill further need the notion of quasi-$\mathcal{OL}$ formulas.

Definition 8 *A* quasi-$\mathcal{OL}$ formula *is static, does not mention Poss and SF, and function symbols of arity greater than zero only appear in equations of the form*

$$f(v_1, \ldots, v_m) = g(w_1, \ldots, w_n),$$

where each v_i and w_j is either a standard name or a variable.

The only extension to the definition of $\mathcal{OL}$ formulas is that a quasi-$\mathcal{OL}$ formula may also mention function symbols of higher arity, but only inside equality statements of the above restricted form.

The reduction of sentences containing *Know* to purely objective ones now is achieved using the following definition.

Definition 9 *Given a finite set of objective $\mathcal{OL}$ sentences Σ_0 and a basic quasi-$\mathcal{OL}$ formula α, $\|\alpha\|_{\Sigma_0}$ is defined by*

$\|\alpha\|_{\Sigma_0} = \alpha$, *when α is an objective $\mathcal{OL}$ formula,*
$\|f(\vec{v}) = g(\vec{w})\|_{\Sigma_0} = \text{FALSE}$, *if f and g are distinct,*[6]
$\|f(\vec{v}) = f(\vec{w})\|_{\Sigma_0} = (\vec{v} = \vec{w})$,[7]
$\|\neg\alpha\|_{\Sigma_0} = \neg\|\alpha\|_{\Sigma_0}$,
$\|(\alpha \wedge \beta)\|_{\Sigma_0} = (\|\alpha\|_{\Sigma_0} \wedge \|\beta\|_{\Sigma_0})$,
$\|\forall v \alpha\|_{\Sigma_0} = \forall v \|\alpha\|_{\Sigma_0}$,
$\|\mathit{Know}(\alpha)\|_{\Sigma_0} = \text{RES}[\![\|\alpha\|_{\Sigma_0}, \Sigma_0]\!]$

That is, we substitute subformulas *Know*(α) by objective formulas describing the individuals for which α is known to be true. In the example, $\|\forall x.Box(x) \supset \mathit{Know}(Box(x))\|_{\Sigma_0}$ is simply $\forall x.Box(x) \supset (x = box1)$, when we use the simplified version of $\text{RES}[\![Box(x), \Sigma_0]\!]$ above. Intuitively, this is correct as $box1$ is the only box known to the robot.

Definition 9 is an extension of the original $\|\cdot\|_{\Sigma_0}$ which could only handle $\mathcal{OL}$ formulas. We newly introduced the second and third item here to enable it to treat quasi-$\mathcal{OL}$ formulas as well, which will be needed in the section "Representation Theorems" when we apply the operator to regression results that may still contain equations between action terms. As long as those equations are of the restricted form given in Definition 8, they can be eliminated using the two new items, thus turning a quasi-$\mathcal{OL}$ formula into an equivalent $\mathcal{OL}$ expression. Notice that in the regression example above, we already applied these reduction steps when we simplified (4) to (5). It is easy to see that we have:

Corollary 10 *$\|\alpha\|_{\Sigma_0}$ is an objective $\mathcal{OL}$ sentence.*

The case for *Know*(α) therefore is well defined: the argument handed over to RES does not contain function symbols of arity greater than zero. Further, the soundness of the simplification method follows directly from the unique names assumption that is integrated in the semantics of $\mathcal{ES}$:

Corollary 11 *If α is an objective quasi-$\mathcal{OL}$ sentence, then $w \models \alpha$ iff $w \models \|\alpha\|_{\Sigma_0}$.*

The following theorem is an easy consequence of a theorem in (Lakemeyer & Levesque 2004) and establishes, together with the previous result, that reasoning about knowledge ultimately reduces to computing first-order entailments.

Theorem 12 *Let Σ_0 be a set of objective $\mathcal{OL}$ sentences and α a basic quasi-$\mathcal{OL}$ sentence.*
Then $\models \mathit{OKnow}(\Sigma_0) \supset \mathit{Know}(\alpha)$ iff $\models \Sigma_0 \supset \|\alpha\|_{\Sigma_0}$.

[6] $f(\vec{v})$ means $f(v_1, \ldots, v_k)$, if f is a k-ary function symbol. f and g do not have to be of the same arity.

[7] Let $\vec{v}$ stand for $v_1, \ldots, v_k$ and $\vec{w}$ stand for $w_1, \ldots, w_k$. Then $(\vec{v} = \vec{w})$ is shorthand for $(v_1 = w_1) \wedge \ldots \wedge (v_k = w_k)$.

Interaction Operations

With these preliminaries, we are now ready to specify what it means for an agent to query its knowledge base and update it given sensing results or user input. In particular, we are interested in operations to do the following:

1. initialize the knowledge base, providing (generally incomplete) initial world knowledge and general knowledge about actions and sensing;
2. update with sensing results after an action was executed physically;
3. pose queries (i.e. evaluating test conditions) about the current or future situations;
4. and provide the system with new information about the current situation (i.e. user input).

This motivates the definition of the operations below, assuming that the system's internal state (e, σ) consists of an epistemic state e that represents the worlds the agent considers possible (the system's knowledge) and a history σ of the actions performed so far:

Definition 13 (Interaction Operations) *Let Σ be a set of objective sentences, e an epistemic state, σ a sequence of ground terms, α a sentence, t a ground term and $i \in \{0, 1\}$. We define*

1. $\text{INIT}[\Sigma] = (e, \langle\rangle)$
   where $e = \{w \mid w \models \Sigma\}$
2. $\text{EXE}[(e, \sigma), t, i] = (e', \sigma \cdot t)$
   where $e' = \{w \in e \mid w[\sigma{:}\mathit{SF}(t)] = i\}$
3. $\text{ASK}[(e, \sigma), \alpha] =$ *"yes" iff $e, w, \sigma \models \alpha$ for all $w \in e$ (and "no" otherwise)*
4. $\text{TELL}[(e, \sigma), \alpha] = (e', \sigma)$
   where $e' = \{w \in e \mid e, w, \sigma \models \alpha\}$

By providing some initial objective knowledge Σ (e.g. a basic action theory), we obtain an initial state of the system where only Σ is known and no actions have been performed so far. For updating the system's state after an action, we have to provide the action t that was executed and the binary sensing result i; the system then incorporates the knowledge gained from sensing and updates its action history. If a query in form of a sentence α is posed, the system evaluates it wrt the current e and σ. A user input α finally forces the agent to believe α.

Note that we did not use *Know*(α) when defining the yes-answer for ASK. This is because the semantics of *Know* requires access to the real world (via $\simeq_\sigma$), which the agent does not have. However, as long as the epistemic state e is changed only as a result of EXE or TELL, then *all* worlds in e will agree with any "real" world compatible with the sensing results or user inputs[8].

Let us see how the interaction with the KB would look like when we want to execute our example program from

[8] cp. also Theorem 19.

the introduction:

if $\neg\exists y.\mathit{Know}(At(box1, y)) \ \wedge$
$\mathit{Know}([lookFor(box1, room1)]\exists y.\mathit{Know}(At(box1, y)))$
  then $lookFor(box1, room1)$;
if $\mathit{Know}(\mathit{Poss}(push(box1, room1, room2)))$
  then $push(box1, room1, room2)$;

In the following, let l denote $lookFor(box1, room1)$ and p stand for $push(box1, room1, room2)$.

1. We begin with initializing the system with the example action theory from above. Therefore let $(e_1, \langle\,\rangle) :=$ INIT$[\Sigma]$.
2. The first condition amounts to determining
$$\text{ASK}[(e_1, \langle\,\rangle),\ \ \neg\exists y.\mathit{Know}(At(box1, y)) \wedge [l]\exists y.\mathit{Know}(At(box1, y)))\ \ ].$$
We have that for all $w \in e_1$,
$$e_1, w, \langle\,\rangle \models \neg\exists y.\mathit{Know}(At(box1, y))$$
since there is no single ground term r such that $e_1, w, \langle\,\rangle \models \mathit{Know}(At(box1, r))$. The reason for this is that because of (1) there are worlds in e_1 where only $At(box1, room1)$ holds and some, where only $At(box1, room2)$ is true. The reader may verify that for all worlds $w \in e_1$ it also holds that
$$e_1, w, \langle\,\rangle \models [l]\exists y.\ \mathit{Know}(At(box1, y)).$$
Therefore the answer is "yes".
3. Because of the answer in the last item, the system now performs the action l. Let us assume that the action's return value is "TRUE", i.e.
$$(e_2, \langle l\rangle) := \text{EXE}[(e_1, \langle\,\rangle), l, 1].$$
e_2 consists of all worlds w such that $w \models \Sigma$ and $w \models \mathit{SF}(l)$. The latter is equivalent to $w \models At(box1, room1)$ according to (3). Since Σ contains the sentences (2), it follows that $w \models \neg At(box1, t)$ for all terms t other than $room1$.
4. Next, we turn to ASK$[(e_2, \langle l\rangle),\ \mathit{Poss}(p)]$.
According to Σ_{pre}, $\mathit{Poss}(p)$ is equivalent to $At(box1, room1) \wedge Room(room2)$. In the last item we saw that all $w \in e_2$ satisfy $At(box1, room1)$. Since the action theory is defined in a way such that $lookFor$ does not change the truth value of At, we have $w, \langle l\rangle \models At(box1, room1)$. Also $w \models Room(room2)$, since this was part of Σ_0. Because the truth value of $Room$ cannot be changed by any action, we additionally have $w, \langle l\rangle \models Room(room2)$. Hence $w, \langle l\rangle \models \mathit{Poss}(p)$ for all $w \in e_2$ and the answer is again "yes".
5. The system now has to perform p. Assuming that this non-sensing action yields a default sensing result of "TRUE", we have
$$(e_3, \langle l, p\rangle) := \text{EXE}[(e_2, \langle l\rangle), p, 1].$$
Because of $\Sigma_{\text{sense}} \models [l]\mathit{SF}(p)$, the system knew the outcome of the action already in advance and therefore did not gain any new knowledge (except for the fact that p was executed). Therefore it holds that $e_3 = e_2 = \{w \mid w \models \Sigma \wedge At(box1, room1)\}$.
6. After the program has terminated, imagine that a user wants to tell the agent that it knows all the boxes. This presents new information to the system since it currently only knows one box ($box1$), but is unsure whether there are more. Telling the system that it knows all the boxes thus amounts to telling it that there are no boxes other than $box1$.
$$(e_4, \langle l, p\rangle) := \text{TELL}[(e_3, \langle l, p\rangle),\ \forall x.Box(x) \supset \mathit{Know}(Box(x)),]$$
yielding
$$e_4 = \{w \in e_3 \mid w \models \forall x.Box(x) \supset (x = box1)\}.$$

Representation Theorems

As explained earlier, our objective is to represent the system's state by a collection of standard first-order logic formulas and reduce the computation of the interaction operations to first-order reasoning, which then allows to execute knowledge-based programs using a standard first-order theorem prover. As a first step, we will see how our meta-theoretic definition of states and interaction operations translates to a characterization by valid $\mathcal{ES}$ sentences.

For a set of objective sentences Σ, let $\Re[\![\Sigma]\!]$ be the set $\{w \mid w \models \Sigma\}$. Observe that this epistemic state is, according to the semantics, the unique e such that $e \models \mathit{OKnow}(\Sigma)$. We obtain the following theorem:

Theorem 14 *Let Σ be a basic action theory, σ a sequence of ground terms, α a sentence, ϕ a fluent sentence, t a ground term and $i \in \{0, 1\}$. Then*

1. INIT$[\Sigma] = (\Re[\![\Sigma]\!], \langle\,\rangle)$
2. EXE$[(\Re[\![\Sigma]\!], \sigma), t, i] = (\Re[\![\Sigma \wedge \phi]\!], \sigma \cdot t)$ *iff*
  $\models \mathit{OKnow}(\Sigma) \supset \mathit{Know}([\sigma](\neg)\mathit{SF}(t) \equiv \phi)$
3. ASK$[(\Re[\![\Sigma]\!], \sigma), \alpha] =$ *"yes" iff*
  $\models \mathit{OKnow}(\Sigma) \supset [\sigma]\mathit{Know}(\alpha)$
4. TELL$[(\Re[\![\Sigma]\!], \sigma), \alpha] = (\Re[\![\Sigma \wedge \phi]\!], \sigma)$ *iff*
  $\models \mathit{OKnow}(\Sigma) \supset \mathit{Know}([\sigma]\alpha \equiv \phi)$

Proof:

1. follows immediately from Definition 13.
2. EXE$[(\Re[\![\Sigma]\!], \sigma), t, i] = (\Re[\![\Sigma \wedge \phi]\!], \sigma \cdot t)$
  iff (by definition)
 $\{w \in \Re[\![\Sigma]\!] \mid w[\sigma{:}\mathit{SF}(t)] = i\} = \{w \mid w \models \Sigma \wedge \phi\}$
  iff (equivalent rewriting)
 $\{w \in \Re[\![\Sigma]\!] \mid w[\sigma{:}\mathit{SF}(t)] = i\} = \{w \in \Re[\![\Sigma]\!] \mid w \models \phi\}$
  iff (equivalent rewriting)
 for all $w \in \Re[\![\Sigma]\!]$: $w[\sigma{:}\mathit{SF}(t)] = i$ iff $w \models \phi$
  iff (equivalent rewriting)
 for all $w \in \Re[\![\Sigma]\!]$: $w \models [\sigma](\neg)\mathit{SF}(t)$ iff $w \models \phi$
 $\models \mathit{OKnow}(\Sigma) \supset \mathit{Know}([\sigma](\neg)\mathit{SF}(t) \equiv \phi)$.
3. "$\Rightarrow$":
 ASK$[(\Re[\![\Sigma]\!], \sigma), \alpha] =$ "yes"
  $\Rightarrow$ (by definition)
 for all $w \in \Re[\![\Sigma]\!]$: $\Re[\![\Sigma]\!], w, \sigma \models \alpha$
  $\Rightarrow$ (for any world w')
 for all $w \in \Re[\![\Sigma]\!]$ with $w \simeq_\sigma w'$: $\Re[\![\Sigma]\!], w, \sigma \models \alpha$
  $\Rightarrow$ (by the semantics)
 if $\Re[\![\Sigma]\!], w' \models \mathit{OKnow}(\Sigma)$, then $\Re[\![\Sigma]\!], w' \models [\sigma]\mathit{Know}(\alpha)$

$\Rightarrow$ (equivalent rewriting)
$\models$ *OKnow*$(\Sigma) \supset [\sigma]$*Know*$(\alpha)$.

"$\Leftarrow$":
ASK$[(\Re[\![\Sigma]\!], \sigma), \alpha]$ = "no"
$\Rightarrow$ (by definition)
there is some $w \in \Re[\![\Sigma]\!]$ such that $\Re[\![\Sigma]\!], w, \sigma \not\models \alpha$
$\Rightarrow$ (by the semantics and since $w \in \Re[\![\Sigma]\!]$ and $w \simeq_\sigma w$)
$\Re[\![\Sigma]\!], w \models$ *OKnow*(Σ), but $\Re[\![\Sigma]\!], w \not\models [\sigma]$*Know*$(\alpha)$
$\Rightarrow$ (equivalent rewriting)
$\not\models$ *OKnow*$(\Sigma) \supset [\sigma]$*Know*$(\alpha)$.

4. TELL$[(\Re[\![\Sigma]\!], \sigma), \alpha] = (\Re[\![\Sigma \wedge \phi]\!], \sigma)$
iff (by definition)
$\{w \in \Re[\![\Sigma]\!] \mid \Re[\![\Sigma]\!], w, \sigma \models \alpha\} = \{w \mid w \models \Sigma \wedge \phi\}$
iff (equivalent rewriting)
$\{w \in \Re[\![\Sigma]\!] \mid \Re[\![\Sigma]\!], w, \sigma \models \alpha\} = \{w \in \Re[\![\Sigma]\!] \mid w \models \phi\}$
iff (equivalent rewriting)
for all $w \in \Re[\![\Sigma]\!]$: $\Re[\![\Sigma]\!], w \models [\sigma]\alpha$ iff $w \models \phi$
iff (by the semantics)
$\models$ *OKnow*$(\Sigma) \supset$ *Know*$([\sigma]\alpha \equiv \phi)$. ∎

The theorem tells us that the initial state of the system can simply be represented by the initial action theory itself together with an empty action history. After an action t was executed and a sensing result provided, it suffices to augment the current representation Σ by a fluent sentence that is known to be equivalent to $[\sigma]SF(t)$ (respectively $[\sigma]\neg SF(t)$ if $i = 0$) and add t to the action history. A basic, bounded query α can be answered by testing whether only-knowing the action theory entails currently knowing α. If the system is told some sentence α about the current situation σ, the knowledge base is to be augmented by a fluent sentence that is known to be equivalent to $[\sigma]\alpha$.

Notice that for a fluent sentence ϕ and a basic action theory Σ, the theory $\Sigma \wedge \phi$ is also always a basic action theory in the sense of Definition 1 (simply view ϕ as being part of the new Σ_0). Therefore, if we start in a state given by INIT$[\Sigma]$, successively applying TELL and EXE always leads to a state that is itself representable by some action theory Σ^*, and we may pose queries using ASK.

Achieving the final reduction of states and operations to first-order reasoning now is straight forward considering the techniques introduced earlier: We may eliminate actions using regression and can treat knowledge by means of $\| \cdot \|_{\Sigma_0}$ and thus acquire the ϕ in question in Theorem 14. Using this idea, we obtain our main result in form of representation theorems for the semantic definition of our interaction operations:

Theorem 15 *Let Σ be a basic action theory over $\mathcal{F}$, e an epistemic state, α a basic, bounded sentence over $\mathcal{F}$, $\sigma \in R^*$, t a ground term and $i \in \{0, 1\}$. Further let Σ_0 be a set of $\mathcal{OL}$ sentences and $\mathcal{R}[\sigma, (\neg)SF(t)]$ and $\mathcal{R}[\sigma, \alpha]$ be quasi-$\mathcal{OL}$ sentences. Then*

1. INIT$[\Sigma] = (e', \langle\rangle)$ *iff* $e' = \Re[\![\Sigma]\!]$
2. EXE$[(\Re[\![\Sigma]\!], \sigma), t, i] = (e', \sigma \cdot t)$ *iff*
$e' = \Re[\![\Sigma \wedge \|\mathcal{R}[\sigma, (\neg)SF(t)]\|_{\Sigma_0}]\!]$
3. ASK$[(\Re[\![\Sigma]\!], \sigma), \alpha]$ = *"yes" iff*
$\models \Sigma_0 \supset \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0}$
4. TELL$[(\Re[\![\Sigma]\!], \sigma), \alpha] = (e', \sigma)$ *iff*
$e' = \Re[\![\Sigma \wedge \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0}]\!]$

Proof:

1. follows immediately from Definition 13.
2. EXE$[(\Re[\![\Sigma]\!], \sigma), t, i] =$
$(\Re[\![\Sigma \wedge \|\mathcal{R}[\sigma, (\neg)SF(t)]\|_{\Sigma_0}]\!], \sigma \cdot t)$
iff (by Theorem 14)
$\models$ *OKnow*(Σ)
$\supset$ *Know*$([\sigma](\neg)SF(t) \equiv \|\mathcal{R}[\sigma, (\neg)SF(t)]\|_{\Sigma_0})$
iff (by Theorem 5)
$\models$ *OKnow*(Σ_0)
$\supset$ *Know*$(\mathcal{R}[\langle\rangle, [\sigma](\neg)SF(t) \equiv \|\mathcal{R}[\sigma, (\neg)SF(t)]\|_{\Sigma_0}])$
iff (by Theorem 12)
$\models \Sigma_0 \supset \|\mathcal{R}[\langle\rangle, [\sigma](\neg)SF(t) \equiv \|\mathcal{R}[\sigma, (\neg)SF(t)]\|_{\Sigma_0}]\|_{\Sigma_0}$
iff (equivalent rewriting)
$\models \Sigma_0 \supset \|\mathcal{R}[\sigma, (\neg)SF(t)] \equiv \mathcal{R}[\sigma, (\neg)SF(t)]\|_{\Sigma_0}$
iff (equivalent rewriting)
$\models \Sigma_0 \supset$ TRUE,
which is a tautology, therefore the proposition holds.
3. ASK$[(\Re[\![\Sigma]\!], \sigma), \alpha]$ = "yes"
iff (by definition)
for all $w \in \Re[\![\Sigma]\!]$: $\Re[\![\Sigma]\!], w \models [\sigma]\alpha$
iff (by the semantics)
$\models$ *OKnow*$(\Sigma) \supset$ *Know*$([\sigma]\alpha)$
iff (by Theorem 5)
$\models$ *OKnow*$(\Sigma_0) \supset$ *Know*$(\mathcal{R}[\sigma, \alpha])$
iff (by Theorem 12)
$\models \Sigma_0 \supset \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0}$
4. TELL$[(\Re[\![\Sigma]\!], \sigma), \alpha] = (\Re[\![\Sigma \wedge \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0}]\!], \sigma)$
iff (by Theorem 14)
$\models$ *OKnow*$(\Sigma) \supset$ *Know*$([\sigma]\alpha \equiv \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0})$
iff (by Theorem 5)
$\models$ *OKnow*$(\Sigma_0) \supset$ *Know*$(\mathcal{R}[\langle\rangle, [\sigma]\alpha \equiv \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0}])$
iff (by Theorem 12)
$\models \Sigma_0 \supset \|\mathcal{R}[\langle\rangle, [\sigma]\alpha \equiv \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0}]\|_{\Sigma_0}$
iff (equivalent rewriting)
$\models \Sigma_0 \supset (\|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0} \equiv \|\mathcal{R}[\sigma, \alpha]\|_{\Sigma_0})$
iff (equivalent rewriting)
$\models \Sigma_0 \supset$ TRUE,
which is a tautology, therefore the proposition holds.

In items 2 and 4 we are able to eliminate the nesting of $\mathcal{R}$ because of the fact that applying regression to an already regressed sentence (which is basic and static) does not change that sentence anymore. We further use a similar property of $\| \cdot \|_{\Sigma_0}$, whose result is always already an objective $\mathcal{OL}$ sentence. ∎

For EXE, the ϕ from Theorem 14 with which Σ is to be augmented is therefore the regression of $SF(t)$ (if $i = 1$) respectively $\neg SF(t)$ (if $i = 0$) through the current action history σ, simplified by $\| \cdot \|_{\Sigma_0}$. Similarly, the argument for a TELL operation has first to be regressed through σ and then be transformed into a fluent sentence using $\| \cdot \|_{\Sigma_0}$, before it can be added to the current knowledge base. ASK can be implemented the same way, with the difference that the result of the transformation is not used to augment Σ, but has to be tested against Σ_0. Since only fluent sentences are involved in this case, the test amounts to computing standard first-order entailments.

Here, we regress both inside and outside of *Know* wrt the agent's belief Σ, as defined in the section about regression. Theorem 14 illustrates why we do not need a different, "real world" Σ' like in (Lakemeyer & Levesque 2004): for ASK, we want to check whether α is *known*. Equally, EXE and TELL asserts $(\neg)\mathit{SF}(t)$ and α to be *known* afterwards. In each case we are therefore already inside an implicit *Know* operator, i.e. we are only concerned with what the agent believes and do not make any assumptions about what the outside world is like. All we "see" of it are the sensing results i; it is unnecessary to require that they conform to some outside action theory Σ'.

The conditions of the theorem that require Σ_0 to be a set of $\mathcal{OL}$ sentences and $\mathcal{R}[\sigma, (\neg)\mathit{SF}(t)]$ and $\mathcal{R}[\sigma, \alpha]$ to be quasi-$\mathcal{OL}$ sentences are necessary in order to use Theorem 12. They do not apply for general Σ and α, but for a large class of "reasonable" theories used for practical implementations. Below we will state some simple, yet sufficient conditions on action theories and queries that assure that the assumptions of Theorem 15 are satisfied. The intuition behind them is that usually, objects of the application domain are denoted by standard names (like $box1$ and $room2$), whereas function symbols that take one or more arguments are only used as actions (e.g. $push(x, y, z)$ and $lookFor(x, y)$). Although our semantics does not distinguish between sorts action and object, we are free to make this distinction when defining our action theory. Further, since Σ_0 is supposed to talk about the initial situation, it does not make much sense that it mentions any actions. Finally, there are usually no fluents other than *Poss* and *SF* that take an action as argument ($At(x, y)$, $Box(x)$ and $Room(x)$ do not).

To formalize these intuitions, we will need the notion of a flat term. A *flat term* is of the form $f(v_1, \ldots, v_k)$ for some k-ary function symbol f, where each v_i is either a standard name or a variable; the k here may also be zero. If x is a variable and n a standard name, then for instance n and $f(x, n)$ are flat terms while x and $f(f(x, n), x)$ are not. Allowing equations between such flat terms are what extends $\mathcal{OL}$ formulas to quasi-$\mathcal{OL}$ formulas (cp. Definition 8). In the following, "function symbols" will refer to f that have one or more arguments (i.e. symbols from G^k for $k > 0$). We require that

1. Σ_0 does not contain function symbols;
2. Σ_{pre}, Σ_{post} and Σ_{sense} mention function symbols only in expressions of the form $(a = f(\vec{v}))$, where a is the free action variable in π, each γ_F and φ (i.e. the right-hand side of the respective axioms) and $f(\vec{v})$ is flat;
3. the action variable a in π, all γ_F and φ also appears only in equations $(a = f(\vec{v}))$, where $f(\vec{v})$ is flat;
4. σ and t are flat;
5. α mentions function symbols only in equations of the form $f(\vec{v}) = g(\vec{w})$ (both flat) and as arguments of *Poss*, *SF* and $[r]$;
6. in α, the arguments of *Poss*, *SF* and $[r]$ are flat terms (and therefore not variables).

We then have:

Lemma 16 *Let Σ, σ, t and α satisfy conditions 1–6 above. Then*

1. *Σ_0 is a set of $\mathcal{OL}$ sentences.*
2. *$\mathcal{R}[\sigma, (\neg)\mathit{SF}(t)]$ is a quasi-$\mathcal{OL}$sentence.*
3. *$\mathcal{R}[\sigma, \alpha]$ is a quasi-$\mathcal{OL}$ sentence.*

Proof: The first item should be clear. Since $(\neg)\mathit{SF}(t)$ is a sentence that satisfies the conditions on α, item 2 is covered by the last item, whose proof is a simple, but tedious induction on the structure of α. ∎

We want to emphasize again that the conditions 1–6 do not really restrict the possible application domains of our approach, if we agree to the assumptions that there are no object functions and that no fluents except *Poss* and *SF* have action arguments. If the successor state axioms are constructed following Reiter's solution to the frame problem, then they even already have the form satisfying conditions 2 and 3. Likewise, the form of Σ_{pre} and Σ_{sense} as a case distinction (i.e. a disjunction over finitely many actions, cp. the example theory) corresponds to the situation calculus theories from (Scherl & Levesque 2003) that have one precondition axiom and one sense effect axiom for each action. Further, the restricted form of possible α still constitutes a gain in expressiveness compared to (Reiter 2001a), who does not allow references to future situations or quantifying-in. Like us, Reiter rules out functional fluents like $lastBoxMoved$; this does however not limit our expressive capabilities since we can use a relational fluent $lastBoxMoved(x)$ and ensure that it is always only true for a single value for x.

Now notice that our example action theory and the test conditions in the program fulfill all of the requirements. We may therefore apply Theorem 15, which looks as follows:

1. Clearly, $e_1 = \Re[\![\Sigma]\!]$.
2. To check that

$$\text{ASK}[(e_1, \langle\rangle), \neg\exists y.\mathit{Know}(At(box1, y)) \wedge \\ [l]\exists y.\ \mathit{Know}(At(box1, y))\ ] = \text{"yes"},$$

let α_1 stand for $\neg\exists y.\mathit{Know}(At(box1, y))$ and α_2 be $[l]\exists y.\mathit{Know}(At(box1, y))$. We therefore have to determine $\|\mathcal{R}[\langle\rangle, \alpha_1 \wedge \alpha_2]\|_{\Sigma_0}$, which is the same as $\|\mathcal{R}[\langle\rangle, \alpha_1]\|_{\Sigma_0} \wedge \|\mathcal{R}[\langle\rangle, \alpha_2]\|_{\Sigma_0}$.

$\mathcal{R}[\langle\rangle, \alpha_1]$ is simply α_1, because regressing through the empty action sequence has no effect here. Applying $\|\cdot\|_{\Sigma_0}$ means substituting $\mathit{Know}(At(box1, y))$ by FALSE, since there is no single y such that $At(box1, y)$ is known: $box1$ may equally be in $room1$ or in $room2$. $\|\mathcal{R}[\langle\rangle, \alpha_1]\|_{\Sigma_0}$ therefore reduces to TRUE, which is clearly entailed by Σ_0.

$\mathcal{R}[\langle\rangle, \alpha_2]$ is $\exists y.\mathcal{R}[\langle l\rangle, Know(At(box1, y))]$, which, applying regression rule 9, equals

$$\mathcal{R}[\langle\rangle, \exists y.\ \mathit{SF}(l) \wedge \mathit{Know}(\mathit{SF}(l) \supset [l]At(box1, y)) \vee \\ \neg\mathit{SF}(l) \wedge \mathit{Know}(\mathit{SF}(l) \supset [l]At(box1, y))\ ]$$

Both inside and outside of *Know*, regression replaces $\mathit{SF}(lookFor(box1, room1))$ by $At(box1, room1)$. The regression of $At(box1, y)$ through l further is simply (a formula that is equivalent to) $At(box1, y)$ again, because the sensing action $lookFor$ is defined in Σ to not change any fluents. We obtain

$$\begin{aligned}\mathcal{R}[\langle\,\rangle, \exists y.\, & At(box1, room1) \wedge \\ & \quad \textit{Know}(At(box1, room1) \supset At(box1, y)) \vee \\ & \neg At(box1, room1) \wedge \\ & \quad \textit{Know}(\neg At(box1, room1) \supset At(box1, y))]\end{aligned}$$

Now for $At(box1, room1) \supset At(box1, y)$, the known instances are given by $(y = room1)$: If $box1$ is in $room1$, it cannot be in any other room because of (2). If it is however not in $room1$, it has to be in $room2$ according to (1); therefore the known instances of $\neg At(box1, room1) \supset At(box1, y)$ are simply $(y = room2)$. Putting this together, we get that $\|\mathcal{R}[\langle\,\rangle, \alpha_2]\|_{\Sigma_0}$ is (with simplifications)

$$\begin{aligned}\exists y.\, (& At(box1, room1) \wedge (y = room1) \vee \\ & \neg At(box1, room1) \wedge (y = room2)\ ),\end{aligned}$$

which is also entailed by Σ_0.

3. We have $e_2 = \Re[\![\Sigma \wedge \|\mathcal{R}[\langle\,\rangle, \textit{SF}(l)]\|_{\Sigma_0}]\!]$
$= \Re[\![\Sigma \wedge At(box1, room1)]\!]$.
To see why, observe that $\mathcal{R}[\langle\,\rangle, \textit{SF}(l)]$ is

$$\begin{aligned}&\exists x, y, z.\ lookFor(box1, room1) = push(x, y, z) \vee \\ &\exists x, y.\ lookFor(box1, room1) = lookFor(x, y) \\ &\qquad \wedge At(x, y)\end{aligned}$$

Applying $\|\cdot\|_{\Sigma_0}$ yields

$$\begin{aligned}&\exists x, y, z.\ \text{FALSE} \vee \\ &\exists x, y.\ (box1 = x) \wedge (room1 = y) \wedge At(x, y)\end{aligned}$$

which can be further simplified to $At(box1, room1)$.

4. ASK$[(e_2, \langle l\rangle), \textit{Poss}(p)]$ ="yes":

$\mathcal{R}[\langle l\rangle, \textit{Poss}(p)]$ simplifies to $\mathcal{R}[\langle l\rangle, At(box1, room1) \wedge Room(room2)]$, which reduces to $At(box1, room1) \wedge Room(room2)$. Since this is an objective sentence without function symbols, applying $\|\cdot\|_{\Sigma_0^*}$ has no effect. Then $\models \Sigma_0^* \supset At(box1, room1) \wedge Room(room2)$, where $\Sigma_0^* = \Sigma_0 \wedge At(box1, room1)$.

5. $e_3 = \Re[\![\Sigma^* \wedge \|\mathcal{R}[\langle l\rangle, \textit{SF}(p)]\|_{\Sigma_0}]\!] = e_2$,
since $\|\mathcal{R}[\langle l\rangle, \textit{SF}(p)]\|_{\Sigma_0}$ simplifies to TRUE.

6. Finally, $e_4 = \Re[\![\Sigma^* \wedge \forall x. Box(x) \supset (x = box1)]\!]$, because regression has no effect on Box and $box1$ is the only box known to the robot.

While executing a knowledge-based program like the above, queries have to be answered wrt to the current state of the system, which is the result of an initialization and the execution of a finite number of actions. For this situation, we will therefore recapitulate by stating a final theorem that shows the correspondence between the three different views that were taken in this paper on knowledge-based programs:

1. in terms of the meta-theoretic interaction operations (Definition 13)
2. expressed as a valid sentence about only-knowing (Theorem 14)
3. reduced to first-order provability (Theorem 15)

We first need a bit of notation for the formula representing the sensing results of an action history:

Definition 17 *Let σ be a sequence of ground action terms and ι a sequence of binary values of the same length. We define $\Psi(\sigma, \iota)$ inductively as follows:*

- $\Psi(\langle\,\rangle, \langle\,\rangle) = \text{TRUE}$
- $\Psi(\sigma \cdot t, \iota \cdot i) = \begin{cases} \Psi(\sigma, \iota) \wedge [\sigma]\textit{SF}(t), & i = 1 \\ \Psi(\sigma, \iota) \wedge [\sigma]\neg\textit{SF}(t), & i = 0 \end{cases}$

A simple consequence of this definition:

Corollary 18 $w \simeq_\sigma w'$ *for some* w' *with* $w' \models \Psi(\sigma, \iota)$ *iff* $w \models \Psi(\sigma, \iota)$.

Proof: Simple induction on σ. ∎

Further we extend the definition of EXE to action sequences as its successive application on the single actions. We then have:

Theorem 19 *Let Σ be a basic action theory, σ a sequence of ground terms, ι a sequence of binary values of the same length and α a basic, bounded sentence over $\mathcal{F}$. Further let Σ_0 be a set of $\mathcal{OL}$ sentences and $\mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]$ and $\mathcal{R}[\langle\,\rangle, [\sigma]\alpha]$ be quasi-$\mathcal{OL}$ sentences. Then the following are equivalent:*

1. ASK[EXE[INIT[Σ], σ, ι], α] = *"yes"*
2. $\models \textit{OKnow}(\Sigma) \wedge \Psi(\sigma, \iota) \supset [\sigma]\textit{Know}(\alpha)$
3. $\models \Sigma_0 \wedge \|\mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]\|_{\Sigma_0} \supset \|\mathcal{R}[\langle\,\rangle, [\sigma]\alpha]\|_{\Sigma_0}$

Proof: "1. iff 2.":
ASK[EXE[INIT[Σ], σ, ι], α] = "yes"
iff (by Definitions 13 and 17)
for all $w \in \Re[\![\Sigma \wedge \Psi(\sigma, \iota)]\!]$: $\Re[\![\Sigma \wedge \Psi(\sigma, \iota)]\!], w, \sigma \models \alpha$
iff (by the semantics)
$\models \textit{OKnow}(\Sigma \wedge \Psi(\sigma, \iota)) \supset \textit{Know}([\sigma]\alpha)$
iff (by Lemma 20 below)
$\models \textit{OKnow}(\Sigma) \wedge \Psi(\sigma, \iota) \supset [\sigma]\textit{Know}(\alpha)$.

"2. iff 3.":
$\models \textit{OKnow}(\Sigma) \wedge \Psi(\sigma, \iota) \supset [\sigma]\textit{Know}(\alpha)$
iff (by Lemma 20 below)
$\models \textit{OKnow}(\Sigma \wedge \Psi(\sigma, \iota)) \supset \textit{Know}([\sigma]\alpha)$
iff (by Corollary 3 and since $\mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]$ is objective)
$\models \textit{OKnow}(\Sigma \wedge \mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]) \supset \textit{Know}([\sigma]\alpha)$
iff (by Theorem 5, viewing $\mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]$ as part of Σ_0)
$\models \textit{OKnow}(\Sigma_0 \wedge \mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]) \supset \textit{Know}(\mathcal{R}[\langle\,\rangle, [\sigma]\alpha])$
iff (by Corollary 11)
$\models \textit{OKnow}(\Sigma_0 \wedge \|\mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]\|_{\Sigma_0}) \supset \textit{Know}(\mathcal{R}[\langle\,\rangle, [\sigma]\alpha])$
iff (by Thm. 12, with $\|\mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]\|_{\Sigma_0}$ as part of Σ_0)
$\models \Sigma_0 \wedge \|\mathcal{R}[\langle\,\rangle, \Psi(\sigma, \iota)]\|_{\Sigma_0} \supset \|\mathcal{R}[\langle\,\rangle, [\sigma]\alpha]\|_{\Sigma_0}$. ∎

To complete the proof of the theorem, we only need the following lemma:

Lemma 20 *Let Σ be a basic action theory and α a basic, bounded sentence over $\mathcal{F}$. Then*
$\models \textit{OKnow}(\Sigma) \wedge \Psi(\sigma, \iota) \supset [\sigma]\textit{Know}(\alpha)$ *iff*
$\models \textit{OKnow}(\Sigma \wedge \Psi(\sigma, \iota)) \supset \textit{Know}([\sigma]\alpha)$.

Proof: $\models \mathit{OKnow}(\Sigma) \wedge \Psi(\sigma,\iota) \supset [\sigma]\mathit{Know}(\alpha)$
iff (by the semantics)
for all w such that $w \models \Sigma$ and $w \simeq_\sigma w'$ for some w' with $w' \models \Psi(\sigma,\iota)$, we have that $\Re[\![\Sigma]\!], w, \sigma \models \alpha$
iff (by Corollary 18)
for all w with $w \models \Sigma \wedge \Psi(\sigma,\iota)$, $\Re[\![\Sigma]\!], w, \sigma \models \alpha$
iff (see below)
for all w with $w \models \Sigma \wedge \Psi(\sigma,\iota)$, $\Re[\![\Sigma \wedge \Psi(\sigma,\iota)]\!], w, \sigma \models \alpha$
iff (by the semantics)
$\models \mathit{OKnow}(\Sigma \wedge \Psi(\sigma,\iota)) \supset \mathit{Know}([\sigma]\alpha)$

Let $e = \Re[\![\Sigma \wedge \Psi(\sigma,\iota)]\!]$ and $e' = \Re[\![\Sigma]\!]$. We show by induction that for w with $w \models \Psi(\sigma,\iota)$,

$$e', w, \sigma\cdot\sigma' \models \beta \text{ iff } e, w, \sigma\cdot\sigma' \models \beta.$$

The cases "$=$", "$F(\vec{r})$", "$\wedge$","$\neg$" and "$\forall$" follow immediately.

"$[r]$":
$e', w, \sigma\cdot\sigma' \models [r]\beta$ iff $e', w, \sigma\cdot\sigma'\cdot r \models \beta$ iff (by induction) $e, w, \sigma\cdot\sigma'\cdot r \models \beta$ iff $e, w, \sigma\cdot\sigma' \models [r]\beta$.

"*Know*":
$e', w, \sigma\cdot\sigma' \models \mathit{Know}(\beta)$
iff (by the semantics)
for all $w' \in e'$ with $w' \simeq_{\sigma\cdot\sigma'} w$, we have $e', w', \sigma\cdot\sigma' \models \beta$
iff (using Corollary 18 and $w \models \Psi(\sigma,\iota)$)
for all w' with $w' \models \Sigma \wedge \Psi(\sigma,\iota)$ and $w' \simeq_{\sigma\cdot\sigma'} w$, we have $e', w', \sigma\cdot\sigma' \models \beta$
iff (equivalent rewriting)
for all $w' \in e$ with $w' \simeq_{\sigma\cdot\sigma'} w$ and $w' \models \Psi(\sigma,\iota)$, we have $e', w', \sigma\cdot\sigma' \models \beta$
iff (by induction)
for all $w' \in e$ with $w' \simeq_{\sigma\cdot\sigma'} w$ and $w' \models \Psi(\sigma,\iota)$, we have $e, w', \sigma\cdot\sigma' \models \beta$
iff (by the semantics)
$e, w, \sigma\cdot\sigma' \models \mathit{Know}(\beta)$ ∎

Conclusions

In this paper we considered what it means for an agent whose knowledge is given in terms of a basic action theory in $\mathcal{ES}$ to pose queries to its own knowledge base after executing any number of actions, how to incorporate the result of sensing information and to accept user input. Compared to Reiter's proposal we retained the advantage of reducing reasoning about knowledge to first-order reasoning, but we gained considerable expressiveness as we can now handle knowledge about knowledge and quantifying-in, for example. There are essentially two features of $\mathcal{ES}$ that have made this possible. One is the use of standard names or rigid designators, which allows for a simple model theory which does not need to appeal to arbitrary first-order structures. The other is the explicit notion of only-knowing, which generalizes Reiter's knowledge closure and is semantically well-founded.

We believe that our results provide us with the foundations to now tackle the next step, a knowledge-based version of Golog that uses $\mathcal{ES}$ instead of Reiter's situation calculus. It was shown in (Lakemeyer & Levesque 2005) how to reconstruct the original Golog in $\mathcal{ES}$. We are currently working on an extension and implementation which allows knowledge-based tests and on-line execution as proposed in this paper.

References

Chellas, B. 1980. *Modal logic: an introduction*. Cambridge, United Kingdom: Cambridge University Press.

Giacomo, G. D., and Levesque, H. J. 1999. Projection using regression and sensors. In *Proc. IJCAI-99*, 160–165.

Jin, Y., and Thielscher, M. 2004. Representing beliefs in the fluent calculus. In *Proc. ECAI-04*, 823–827.

Lakemeyer, G., and Levesque, H. J. 1998. AOL: a logic of acting, sensing, knowing, and only knowing. In *Proc. KR-98*.

Lakemeyer, G., and Levesque, H. J. 1999. Query evaluation and progression in AOL knowledge bases. In *Proc. IJCAI-99*.

Lakemeyer, G., and Levesque, H. J. 2004. Situations, si! situation terms, no! In *Proc. KR-04*. AAAI Press.

Lakemeyer, G., and Levesque, H. J. 2005. Semantics for a useful fragment of the situation calculus. In *Proc. IJCAI-05*.

Levesque, H. J., and Lakemeyer, G. 2001. *The Logic of Knowledge Bases*. MIT Press.

Levesque, H. J.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. B. 1997. GOLOG: A logic programming language for dynamic domains. *J. Log. Program.* 31(1-3):59–83.

Lobo, J.; Mendez, G.; and Taylor, S. R. 1997. Adding knowledge to the action description language A. In *Proc. AAAI/IAAI '97*, 454–459.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh: University Press. 463–502.

Moore, R. C. 1977. Reasoning about knowledge and action. In *Proc. IJCAI-77*, 223–227.

Petrick, R. P. A., and Bacchus, F. 2002. A knowledge-based approach to planning with incomplete information and sensing. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proc. AIPS-02*, 212–222. AAAI.

Reiter, R. 1991. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. *Artificial intelligence and mathematical theory of computation: papers in honor of John McCarthy* 359–380.

Reiter, R. 2001a. On knowledge-based programming with sensing in the situation calculus. *ACM Trans. Comput. Logic* 2(4):433–457.

Reiter, R. 2001b. *Knowledge in action : logical foundations for specifying and implementing dynamical systems*. Cambridge, Mass.: MIT Press. The frame problem and the situation calculus.

Scherl, R. B., and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artif. Intell.* 144(1-2):1–39.