Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

ELSEVIER

Artificial Intelligence 144 (2003) 1–39

Artificial Intelligence

www.elsevier.com/locate/artint

Knowledge, action, and the frame problem

Richard B. Scherl [a,*], Hector J. Levesque [b]

[a] *Computer Science Department, Monmouth University, West Long Branch, NJ 07764, USA*
[b] *Department of Computer Science, University of Toronto, Toronto, ON, Canada M5S 3A6*

Received 11 January 1994; received in revised form 19 April 2002

Abstract

This paper proposes a method for handling the frame problem for knowledge-producing actions. An example of a knowledge-producing action is a sensing operation performed by a robot to determine whether or not there is an object of a particular shape within its grasp. The work is an extension of Reiter's approach to the frame problem for ordinary actions and Moore's work on knowledge and action. The properties of our specification are that knowledge-producing actions do not affect fluents other than the knowledge fluent, and actions that are not knowledge-producing only affect the knowledge fluent as appropriate. In addition, *memory* emerges as a side-effect: if something is known in a certain situation, it remains known at successor situations, unless something relevant has changed. Also, it will be shown that a form of *regression* examined by Reiter for reducing reasoning about future situations to reasoning about the initial situation now also applies to knowledge-producing actions.

Keywords: Knowledge; Action; Situation calculus; Frame problem

1. Introduction

The situation calculus [24] provides a formalism for reasoning about actions and their effects on the world. Axioms are used to specify the prerequisites of actions as well as their effects, that is, the fluents that they change. In general [23], it is also necessary to provide frame axioms to specify which fluents remain unchanged by the actions. In the worst case this might require an axiom for every combination of action and fluent. Reiter [32] (generalizing the work of Haas [11], Schubert [41] and Pednault [29]) has given a

* Corresponding author.
E-mail addresses: rscherl@monmouth.edu (R.B. Scherl), hector@cs.toronto.edu (H.J. Levesque).

0004-3702/02/$ – see front matter
doi:10.1016/S0004-3702(02)00365-X

set of conditions under which the explicit specification of frame axioms can be avoided. This approach to dealing with the frame problem and the resulting style of axiomatization has proven useful as the foundation for the high-level robot programming language Golog [19].

In this paper, we extend Reiter's approach to the frame problem to cover *knowledge-producing actions*, that is, actions whose effects are to change a state of knowledge. The extension preserves Reiter's solution for actions that change the state of the world and also handles actions that change the knowledge of an agent. The result is a uniform style of axiomatization for both types of actions. We show that our solution has the desired properties with respect to changes in knowledge.

A standard example of a knowledge-producing action is that of reading a number on a piece of paper. Consider the problem of dialing the combination of a safe [23,27,28]. If an agent is at the same place as the safe, and knows the combination of the safe, then he can open the safe by performing the action of dialing that combination. If an agent is at the same place as both the safe and a piece of paper and he knows that the combination of the safe is written on the paper, he can open the safe by first reading the piece of paper, and then dialing that combination. The effect of the read action, then, is to change the knowledge state of the agent, typically to satisfy the prerequisite of a later action. Another example of a knowledge-producing action is performing an experiment to determine whether or not a solution is an acid [28]. Still other examples are a sensing operation performed by a robot to determine the shapes of objects within its grasp [16,18] and the execution of UNIX commands such as **ls** [5].

To incorporate knowledge-producing actions like these into the situation calculus, it is necessary to treat knowledge as a fluent that can be affected by actions. This is precisely the approach taken by Moore [27]. What is new here is that the knowledge fluent and knowledge-producing actions are handled in a way that avoids the frame problem: we will be able to prove as a consequence of our specification that knowledge-producing actions do not affect fluents other than the knowledge fluent, and that actions that are not knowledge-producing only affect the knowledge fluent as appropriate. In addition, we will show that *memory* emerges as a side-effect: if something is known in a certain situation, it remains known at successor situations, unless something relevant has changed. We will also show that a form of *regression* examined by Reiter for reducing reasoning about future situations to reasoning about the initial situation now also applies to knowledge-producing actions. This has the desirable effect of allowing us to reduce reasoning about knowledge and action to reasoning about knowledge in the initial situation, where standard theorem-proving techniques for modal logics may be used. Finally, we show that if certain useful properties of knowledge (such as positive introspection) are specified to hold in the initial state, they will continue to hold automatically at all successor situations.

In Section 2, we briefly review the situation calculus and Reiter's approach to the frame problem. Then in Section 3, we introduce an epistemic fluent into the situation calculus as an accessibility relation over situations, as was done by Moore [27,28]. Our method of handling the frame problem for knowledge-producing actions, based on the epistemic fluent, is developed and illustrated over the next three sections. Section 4 shows how this epistemic fluent can form the foundation for an integrated theory of knowledge and action. Section 5 illustrates the approach with a simple example. The correctness

of the formulation is demonstrated in Section 6. Then in Section 7, we consider the issue of knowledge about knowledge. Section 8 develops a method of regression for the situation calculus with knowledge-producing actions. This forms the basis for a method for automating reasoning with theories involving knowledge and knowledge-producing actions. An extended example is given in Section 9. Finally in the conclusion, Section 10, related work is discussed.

2. The situation calculus and the frame problem

The situation calculus (following the presentation in [32]) is a first-order language for representing dynamically changing worlds in which all of the changes are the result of named *actions* performed by some agent. Terms are used to represent states of the world, i.e., *situations*. If α is an action and s a situation, the result of performing α in s is represented by $\text{DO}(\alpha, s)$. The constant S_0 is used to denote the initial situation. Relations whose truth values vary from situation to situation, called *fluents*, are denoted by a predicate symbol taking a situation term as the last argument. For example, $\text{BROKEN}(x, s)$ means that object x is broken in situation s. Functions whose denotations vary from situation to situation are called *functional fluents*. They are denoted by a function symbol with an extra argument taking a situation term, as in $\text{PHONE-NUMBER}(\text{BILL}, s)$.

It is assumed that the axiomatizer has provided for each action $\alpha(\vec{x})$, an *action precondition axiom* of the form[1] given in (1), where $\pi_\alpha(\vec{x}, s)$ is the formula for $\alpha(\vec{x})$'s action preconditions.

Action Precondition Axiom.

$$\text{POSS}\big(\alpha(\vec{x}), s\big) \equiv \pi_\alpha(\vec{x}, s) \tag{1}$$

An action precondition axiom for the action *drop* is given below.

$$\text{POSS}\big(\text{DROP}(x), s\big) \equiv \text{HOLDING}(x, s) \tag{2}$$

Furthermore, the axiomatizer has provided for each fluent F, two *general effect axioms* of the form given in (3) and (4).

General Positive Effect Axiom for Fluent F.

$$\gamma_{\text{F}}^{+}(a, s) \rightarrow \text{F}\big(\text{DO}(a, s)\big) \tag{3}$$

General Negative Effect Axiom for Fluent F.

$$\gamma_{\text{F}}^{-}(a, s) \rightarrow \neg\text{F}\big(\text{DO}(a, s)\big) \tag{4}$$

[1] By convention, variables are indicated by lower-case letters in italic font. When quantifiers are not indicated, the variables are implicitly universally quantified.

Here $\gamma_F^+(a,s)$ is a formula describing under what conditions doing the action a in situation s leads the fluent F to become true in the successor situation $\text{DO}(a,s)$ and similarly $\gamma_F^-(a,s)$ is a formula describing the conditions under which performing action a in situation s results in the fluent F becoming false in situation $\text{DO}(a,s)$.

For example, (5) is a positive effect axiom for the fluent BROKEN.

$$\begin{aligned}&\big[\big(a = \text{DROP}(y) \wedge \text{FRAGILE}(y)\big)\\ &\qquad\vee\\ &\big(\exists b\, a = \text{EXPLODE}(b) \wedge \text{NEXTO}(b,y,s)\big)\big]\\ &\qquad\rightarrow \text{BROKEN}\big(y, \text{DO}(a,s)\big)\end{aligned} \tag{5}$$

Sentence (6) is a negative effect axiom for BROKEN.

$$a = \text{REPAIR}(y) \rightarrow \neg\text{BROKEN}\big(y, \text{DO}(a,s)\big) \tag{6}$$

It is also necessary to add *frame axioms* that specify when fluents remain unchanged. The frame problem arises because the number of these frame axioms in the general case is $2 \times \mathcal{A} \times \mathcal{F}$, where $\mathcal{A}$ is the number of actions and $\mathcal{F}$ is the number of fluents.

The approach to handling the frame problem [29,32,41] rests on a *completeness assumption*. This assumption is that axioms (3) and (4) characterize all the conditions under which action a can lead to a fluent F's becoming true (respectively, false) in the successor situation. Therefore, if action a is possible and F's truth value changes from *false* to *true* as a result of doing a, then $\gamma_F^+(a,s)$ must be *true* and similarly for a change from *true* to *false* ($\gamma_F^-(a,s)$ must be true). Additionally, *unique name axioms* are added for actions and situations.

Reiter [32] shows how to derive a set of *successor state axioms* of the form given in (7) from the axioms (positive effect, negative effect and unique name) and the completeness assumption.

Successor State Axiom.

$$\text{F}\big(\text{DO}(a,s)\big) \equiv \gamma_F^+(a,s) \vee \big(\text{F}(s) \wedge \neg\gamma_F^-(a,s)\big) \tag{7}$$

Similar successor state axioms may be written for functional fluents. A successor state axiom is needed for each fluent F, and an action precondition axiom is needed for each action a. The unique name axioms need not be explicitly represented as their effects can be compiled. Therefore only $\mathcal{F} + \mathcal{A}$ axioms are needed.

From (5) and (6), the following successor state axiom for BROKEN is obtained.

$$\begin{aligned}&\text{BROKEN}\big(y, \text{DO}(a,s)\big) \equiv\\ &\quad\big(a = \text{DROP}(y) \wedge \text{FRAGILE}(y)\big) \vee\\ &\quad\big(\exists b\, a = \text{EXPLODE}(b) \wedge \text{NEXTO}(b,y,s)\big) \vee\\ &\quad\big(\text{BROKEN}(y,s) \wedge a \neq \text{REPAIR}(y)\big)\end{aligned} \tag{8}$$

Now note for example that if $\neg\text{BROKEN}(\text{OBJ}_1, \text{S}_0)$ holds, then it also follows (given the unique name axioms) that $\neg\text{BROKEN}(\text{OBJ}_1, \text{DO}(\text{DROP}(\text{OBJ}_2), \text{S}_0))$ holds as well.

This discussion has assumed that there are no ramifications, i.e., indirect effects of actions. This can be ensured by prohibiting state constraints, i.e., sentences that specify

an interaction between fluents. An example of such a sentence is $\forall s\ \mathrm{P}(s) \equiv \mathrm{Q}(s)$. The assumption that there are no state constraints in the axiomatization of the domain will be made throughout this paper. In [21,26], the approach discussed in this section is extended to work with state constraints by compiling the effects of the state constraints into the successor state axioms.

3. An epistemic fluent

The approach we take to formalizing knowledge is to adapt the standard possible-world model of knowledge to the situation calculus, as first done by Moore [27]. Informally, we think of there being a binary accessibility relation over situations, where a situation s' is understood as being accessible from a situation s if as far as the agent knows in situation s, he might be in situation s'. So something is known in s if it is true in every s' accessible from s, and conversely something is not known if it is false in some accessible situation.

To treat knowledge as a fluent, we introduce a binary relation $\mathrm{K}(s', s)$, read as "s' is accessible from s" and treat it the same way we would any other fluent. In other words, from the point of view of the situation calculus, the last argument to K is the official situation argument (expressing what is known in situation s), and the first argument is just an auxiliary like the y in $\textsc{Broken}(y, s)$.[2]

It is also necessary to axiomatize further properties of the K fluent. This issue will be discussed in more detail in Section 7. But for now it is sufficient to mention that since we want a logic of knowledge, it is necessary that the K relation be reflexive. Therefore, we need to ensure that our axiomatization entails $\forall s\ \mathrm{K}(s, s)$. Details on how this is done will be given in Section 7.

We can now introduce the notation $\mathbf{Knows}(\mathrm{P}, s)$ (read as P is known in situation s) as an abbreviation for a formula that uses K. For example

$$\mathbf{Knows}\big(\textsc{Broken}(y), s\big) \stackrel{\text{def}}{=} \forall s'\ \mathrm{K}(s', s) \rightarrow \textsc{Broken}(y, s')$$

Note that this notation supplies the appropriate situation argument to the fluent on expansion (and other conventions are certainly possible). For the case of equality literals, the convention is to supply the situation argument to each non-variable argument of the equality predicate. For example:

$$\begin{aligned}&\mathbf{Knows}\big(\textsc{number}(\textsc{bill}) = \textsc{number}(\textsc{mary}), s\big) \stackrel{\text{def}}{=}\\ &\quad \forall s'\ \mathrm{K}(s', s)\ \rightarrow\ \textsc{number}(\textsc{bill}, s') = \textsc{number}(\textsc{mary}, s')\end{aligned}$$

This notation can be generalized inductively to arbitrary formulas so that, for example

$$\begin{aligned}&\exists x\ \mathbf{Knows}\big(\exists y\big[\textsc{Nexto}(x, y) \wedge \neg\textsc{Broken}(y)\big], s\big) \stackrel{\text{def}}{=}\\ &\quad \exists x\ \forall s'\ \mathrm{K}(s', s) \rightarrow \exists y\big[\textsc{Nexto}(x, y, s') \wedge \neg\textsc{Broken}(y, s')\big]\end{aligned}$$

We will however restrict our attention to knowledge about atomic formulas in both this and the next section. In Section 6.3, we discuss the generalization of the results to knowledge of

[2] Note that using this convention means that the arguments to K are reversed from their normal modal logic use.

non-atomic formula. Finally, in Section 7, we explore issues raised by arbitrary formulas, in particular those expressing knowledge about knowledge.

Turning now to knowledge-producing actions, there are two sorts of actions to consider: actions whose effect is to make known the truth value of some formula, and actions that make known the value of some term. In the first case, we might imagine a SENSEP action for a fluent P, such that after doing a SENSEP, the truth value of P is known. We introduce the notation **Kwhether**(P, s) as an abbreviation for a formula indicating that the truth value of a fluent P is known.

$$\textbf{Kwhether}(\mathrm{P}, s) \stackrel{\text{def}}{=} \textbf{Knows}(\mathrm{P}, s) \vee \textbf{Knows}(\neg \mathrm{P}, s)$$

It will follow from our specification in the next section that **Kwhether**(P, DO(SENSEP, s)) holds. In the second case, we might imagine an action READ_τ for a term τ, such that after doing a READ_τ, the denotation of τ is known. For this case, we introduce the notation **Kref**(τ, s) defined as follows:

$$\textbf{Kref}(\tau, s) \stackrel{\text{def}}{=} \exists x\ \textbf{Knows}(\tau = x, s) \quad \text{where } x \text{ does not appear in } \tau.$$

It will follow from the specification developed in the next section that **Kref**(τ, DO(READ_τ, s)) holds.

4. Integrating knowledge and action

The approach being developed here rests on the specification of a successor state axiom for the K relation. This successor state axiom will ensure that for all situations DO(a, s), the K relation will be completely determined by the K relation at s and the action a.

The successor state axiom for K will be developed in several steps through a diagrammatic illustration of possible models for an axiomatization. First, we illustrate the initial picture, without any actions. Then, we add a successor state axiom for K that works with ordinary non-knowledge-producing actions. Finally, we add knowledge-producing actions.

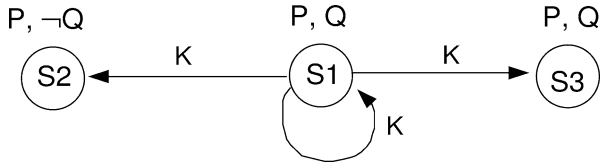

Fig. 1. Initial situation.

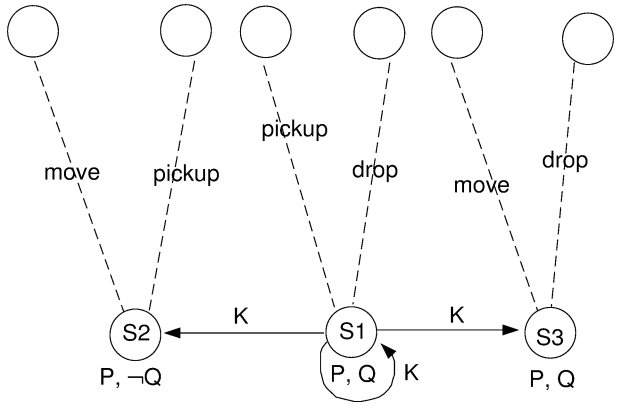

Fig. 2. Multiple actions.

4.1. The initial picture: Without actions

For illustration, consider Fig. 1, which depicts a representation[3] of a model for an axiomatization of the initial situation (without any actions). We can imagine that the term S_0 denotes the situation S1 in the figure. Three situations (S1, S2 and S3) are accessible via the K relation from S1. Proposition P is true in all of these situations,[4] while proposition Q is true in S1 and S3, but is false in S2. Therefore the agent in S1 knows P, but does not know Q. In other words, the picture depicts a model of the sentences **Knows**(P, S_0) and ¬**Knows**(Q, S_0).

4.2. Adding ordinary actions

As illustrated in Fig. 2, from this model of an axiomatization of S_0 and the DO function along with the presence of actions in the language, we have additional situations present in the model. The function denoted by DO maps the initial set of situations to these other situations. (These in turn are mapped to yet other situations, and so on.) These situations intuitively represent the occurrence of actions. The situations S1, S2, and S3 are mapped by DO and the action terms MOVE, PICKUP, or DROP to various other situations. The question is what is the K relation between these situations. Our axiomatization of the K relation places constraints on the K relation in the models. We first cover the simpler case of non-knowledge-producing actions and then discuss knowledge-producing actions.

[3] For simplicity, we omit some of the edges representing the K relation. For example, the edges indicating that the relation is reflexive are omitted from S2 and S3.

[4] For expository purposes we speak informally of a proposition being true in a situation rather than saying that the situation is in the relation denoted by the predicate symbol P.

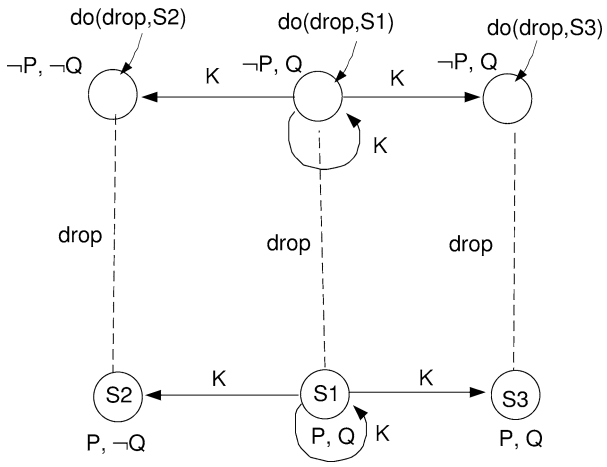

Fig. 3. The K relation.

For non-knowledge-producing actions (e.g., $\text{DROP}(x)$), the specification (based on Moore [27,28]) is as follows:

$$\begin{aligned}&\text{K}\big(s'', \text{DO}(\text{DROP}(x), s)\big) \equiv \\ &\quad \exists s' \left(\text{POSS}\big(\text{DROP}(x), s'\big) \wedge \text{K}(s', s) \wedge s'' = \text{DO}\big(\text{DROP}(x), s'\big)\right)\end{aligned} \tag{9}$$

The idea here is that as far as the agent at world s knows, he could be in any of the worlds s' such that $\text{K}(s', s)$. At $\text{DO}(\text{DROP}(x), s)$ as far as the agent knows, he can be in any of the worlds $\text{DO}(\text{DROP}(x), s')$ for any s' such that both $\text{K}(s', s)$ and $\text{POSS}(\text{DROP}(x), s')$ hold. So the only change in knowledge (required by (9)) that occurs in moving from s to $\text{DO}(\text{DROP}(x), s)$ is the knowledge that the action DROP has been performed. (Other changes may be required by the successor state axioms of the various fluents. This issue will be discussed later.)

To continue our illustration, consider Fig. 3, which extends the initial arrangement depicted in Fig. 1 to include situations resulting from the DO function applied to DROP and the K relation between these situations. Here we see the situation `do(drop,S1)`, denoted by $\text{DO}(\text{DROP}, \text{S}_0)$, which represents the result of performing a drop action in the situation denoted by S_0. Our axiomatization requires that this situation be K related only to the situations `do(drop,S1)`, `do(drop,S2)` and `do(drop,S3)`. Therefore, the agent in effect is modeled as knowing that the drop action has occurred since every situation K related to `do(drop,S1)` is one that results from the DO function and the action DROP.

We suppose for purposes of the running example that the successor state axioms for P and Q are as follows:

$$\text{P}\big(\text{DO}(a, s)\big) \equiv a \neq \text{DROP}(y) \ \wedge \ \text{P}(s) \tag{10}$$

$$\text{Q}\big(\text{DO}(a, s)\big) \equiv \text{Q}(s) \tag{11}$$

The DROP action does not affect the truth of Q, but makes P false. So, we see that proposition P is false in each of `do(drop,S1)`, `do(drop,S2)` and `do(drop,S3)`, while proposition Q is true in `do(drop,S1)` and `do(drop,S3)`, but is false in

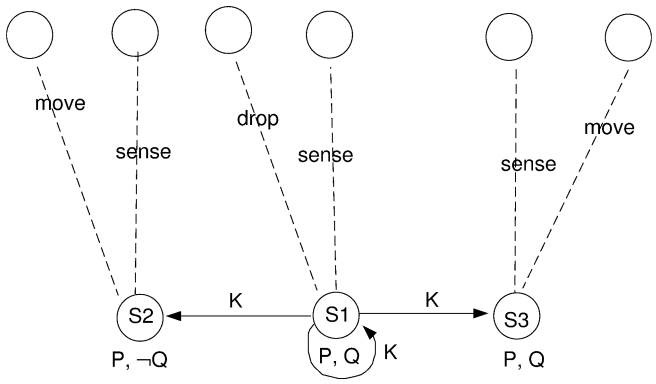

Fig. 4. Sensing actions.

`do(drop,S2)`. Therefore the agent in `do(drop,S1)` knows ¬P, but still does not know Q. The following two sentences hold in this model: **Knows**(¬P, DO(DROP, S_0)) and ¬**Knows**(Q, DO(DROP, S_0)). The agent's knowledge of Q has remained the same, and the knowledge of P is a result of the knowledge of P in the previous situation along with the knowledge of the effect of the action DROP.

4.3. Adding knowledge-producing actions

Now consider the simple case of a knowledge-producing action SENSE_Q that determines whether or not the fluent Q is true (following Moore [27,28]). Fig. 4 extends the same initial picture of Fig. 1. But now we now have the possibility of sensing actions occurring as well as ordinary actions.

We imagine that the action has an associated sensing result function.[5] This result is "YES" if "Q" is true and "NO" otherwise. The symbols are given in quotes to indicate that they are not fluents. We axiomatize the sensing result as follows:

$$\begin{aligned}&\text{SR}(\text{SENSE}_\text{Q}, s) = r \equiv \\ &\quad \big(r = \text{“YES”} \wedge \text{Q}(s)\big) \vee \big(r = \text{“NO”} \wedge \neg\text{Q}(s)\big)\end{aligned} \tag{12}$$

The question that we need to consider is what situations are K accessible from $\text{DO}(\text{SENSE}_\text{Q}, s_0)$.

$$\begin{aligned}&\text{K}\big(s'', \text{DO}(\text{SENSE}_\text{Q}, s)\big) \equiv \\ &\quad \exists s' \big(\text{POSS}(\text{SENSE}_\text{Q}, s') \wedge \text{K}(s', s) \wedge \\ &\quad s'' = \text{DO}(\text{SENSE}_\text{Q}, s') \wedge \text{SR}(\text{SENSE}_\text{Q}, s) = \text{SR}(\text{SENSE}_\text{Q}, s')\big)\end{aligned} \tag{13}$$

[5] In [38], we did not use the sensing result function (SR) in our axiomatization of sensing actions. This resulted in a relatively complex successor-state axiom for the K fluent. The current presentation is an improvement upon the approach first used in [20], and then extended in [44]. By using the sensing result function (SR), it is no longer necessary to distinguish between knowledge-producing actions of the *read* type (actions that make known the denotation of a functional fluent) and of the *sense* type (actions that make known a relational fluent). They are all handled in a uniform fashion.

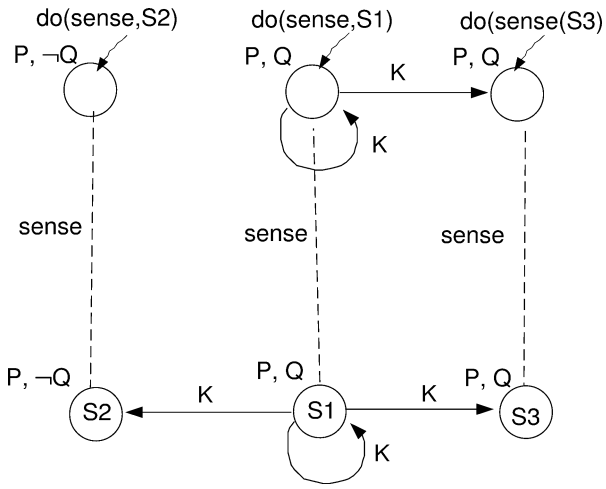

Fig. 5. Producing knowledge.

Again, as far as the agent at world s knows, he could be in any of the worlds s' such that $\mathrm{K}(s', s)$ holds. At $\mathrm{DO}(\mathrm{SENSE_Q}, s)$ as far as the agent knows, he can be in any of the worlds $\mathrm{DO}(\mathrm{SENSE_Q}, s')$ such that $\mathrm{K}(s', s)$ and $\mathrm{POSS}(\mathrm{SENSE_Q}, s')$ hold (by (13)), and also $\mathrm{Q}(s) \equiv \mathrm{Q}(s')$ (by the combination of (12) and (13)) holds. The idea here is that in moving from s to $\mathrm{DO}(\mathrm{SENSE_Q}, s)$, the agent not only knows that the action $\mathrm{SENSE_Q}$ has been performed (since every accessible situation results from the DO function and the $\mathrm{SENSE_Q}$ action), but also the truth value of the predicate Q. Observe that the successor state axiom for Q (sentence (11)) guarantees that Q is true at $\mathrm{DO}(\mathrm{SENSE_Q}, s)$ if and only if Q is true at s, and similarly for s' and $\mathrm{DO}(\mathrm{SENSE_Q}, s')$. Therefore, Q has the same truth value in all worlds s'' such that $\mathrm{K}(s'', \mathrm{DO}(\mathrm{SENSE_Q}, s))$, and so **Kwhether**$(\mathrm{Q}, \mathrm{DO}(\mathrm{SENSE_Q}, s))$ is true.

To return to our running example, consider Fig. 5, which is the illustration of the result of a $\mathrm{SENSE_Q}$ action. Note that the only situations accessible via the K relation from `do(sense,S1)` (denoted by $\mathrm{DO}(\mathrm{SENSE_Q}, s_0)$) are `do(sense, S1)` and `do(sense,S3)`. The situation `do(sense,S2)` is not K accessible. Therefore **Knows**$(\mathrm{P}, \mathrm{DO}(\mathrm{SENSE_Q}, \mathrm{S_0}))$ is true as it was before the action was executed, but also now **Knows**$(\mathrm{Q}, \mathrm{DO}(\mathrm{SENSE_Q}, \mathrm{S_0}))$ is true. The knowledge of the agent being modeled has increased.

In general, there may be many knowledge-producing actions, as well as many ordinary actions. To characterize all of these, we have a function SR (for sensing result), and for each action α, a sensing-result axiom of the form:

$$\mathrm{SR}\big(\alpha(\vec{x}), s\big) = r \equiv \phi_\alpha(\vec{x}, r, s) \tag{14}$$

For ordinary actions, the result is always the same, with the specific result not being significant. For example, we could have:

$$\mathrm{SR}\big(\mathrm{PICKUP}(x), s\big) = r \equiv r = \text{“OK”} \tag{15}$$

In the case of a READ_τ action that makes the denotation of the term τ known, we would have:

$$\text{SR}(\text{READ}_\tau, s) = r \equiv r = \tau(s) \tag{16}$$

Therefore, τ has the same denotation in all worlds s'' such that $\text{K}(s'', \text{DO}(\text{READ}_\tau, s))$, and so $\textbf{Kref}(\tau, \text{DO}(\text{READ}_\tau, s))$ is true.

Consider as another example the following:

$$\begin{aligned}&\text{SR}(\text{SENSE\_WEATHER}, s) = r \equiv \\ &\quad (r = \text{“SUNNY”} \vee r = \text{“RAINY”} \vee r = \text{“SNOW”}) \wedge \text{WEATHER}(s) = r\end{aligned} \tag{17}$$

In this case the sensing result function has three possible values.

The successor state axiom for K is as follows:

Successor State Axiom for K.

$$\begin{aligned}&K\big(s'', \text{DO}(a, s)\big) \equiv \\ &\quad \big(\exists s'\ s'' = \text{DO}(a, s') \\ &\qquad \wedge K(s', s) \wedge \text{POSS}(a, s') \\ &\qquad\quad \wedge \text{SR}(a, s) = \text{SR}(a, s')\big)\end{aligned} \tag{18}$$

The relation K at a particular situation $\text{DO}(a, s)$ is completely determined by the relation at s and the action a.

Two additional issues need to be addressed. The first is that we adopt the following axiomatization policy to simplify the presentation of our system, enhance our ability to prove properties of the system, and simplify the development of regression for reasoning with the resulting axiomatizations:

Axiomatization Policy. *All actions are to be axiomatized as affecting only either the K fluent or other fluents.*

The policy ensures a sharp division between knowledge-producing actions and ordinary actions. Without this policy there is nothing to prevent us from having an action such as open the bag which causes the bag to be open and makes the agent aware of the content of the bag.[6] But this policy does not restrict the capabilities of the agents that we model as we can always follow an open action (which only causes the bag to be open) by a sense action (which causes the agent to know what the contents of the bag are). We can now speak of knowledge-producing actions and ordinary actions as two disjoint classes of actions.

The second issue is that even though our examples so far have involved knowledge of fluents, our approach correctly handles knowledge of sentences and open formula correctly. We can write formula such as

$$\textbf{Knows}\big(\forall x\ \text{BLUE}(x) \wedge \text{OBJECT}(x) \rightarrow \text{BIG}(x), s\big)$$

[6] Example suggested by an anonymous reviewer of the paper.

or

$$\exists x\ \mathbf{Knows}\big(\text{BLUE}(x) \wedge \text{OBJECT}(x) \wedge \text{BIG}(x), s\big)$$

Here we are talking about sentences and formula that do not involve the K fluent. The issue of knowledge about knowledge will be covered in Section 7.

5. Example

Consider the example of opening a safe whose combination is written on a piece of paper (adapted from Moore [27], but without the frame axioms). The preconditions for the action DIAL-COMB(x) (dialing the combination of the safe x and also pulling the handle) are:

$$\begin{aligned}&\text{POSS}\big(\text{DIAL-COMB}(x), s\big) \equiv \\ &\quad \text{SAFE}(x, s) \wedge \text{AT}(x, s) \wedge \mathbf{Kref}\big(\text{COMB}(x), s\big)\end{aligned} \tag{19}$$

The fluent AT(x, s) holds when the agent is located at the location of object x in situation s. The idea in sentence (19) is that for the dialing action to be possible, the object being dialed needs to be a safe, the agent needs to be at the safe, and the agent needs to know the combination of the safe. In this paper, we do not discuss the details of the connection between the agent's ability to open the safe by dialing the combination and the agent's knowledge of the combination of the safe.[7]

The successor state axioms for the fluents OPEN (i.e., something is open), SAFE, AT, and the functional fluents INFO and COMB are as follows:

$$\begin{aligned}&\text{OPEN}\big(x, \text{DO}(a, s)\big) \equiv \\ &\quad a = \text{DIAL-COMB}(x) \vee \big(\text{OPEN}(x, s) \wedge a \neq \text{LOCK}(x)\big)\end{aligned} \tag{20}$$

$$\text{SAFE}\big(x, \text{DO}(a, s)\big) \equiv \text{SAFE}(x, s) \tag{21}$$

$$\begin{aligned}&\text{AT}\big(x, \text{DO}(a, s)\big) \equiv \\ &\quad a = \text{MOVETO}(x) \vee \big(\text{AT}(x, s) \wedge \neg\exists y\ a = \text{MOVETO}(y)\big)\end{aligned} \tag{22}$$

$$\text{INFO}\big(x, \text{DO}(a, s)\big) = y \equiv \text{INFO}(x, s) = y \tag{23}$$

$$\text{COMB}\big(x, \text{DO}(a, s)\big) = y \equiv \text{COMB}(x, s) = y \tag{24}$$

The functional fluent INFO(x, s) is used to denote what is written on paper x.

The axiomatization of the initial state includes SAFE(SF, S_0), AT(SF, S_0), AT(PPR, S_0), and **Knows**(INFO(PPR) = COMB(SF), S_0). Note that the axiomatization does not entail POSS(DIAL-COMB(SF), S_0).

There is a knowledge-producing action READ(x), with the following action precondition axiom:

$$\text{POSS}\big(\text{READ}(x), s\big) \equiv \text{AT}(x, s) \tag{25}$$

[7] Indeed, a more refined axiomatization would allow the safe to be opened without knowing its combination, namely by dialing a number that (magically or luckily) happens to be the combination.

The SR axioms are as follows:

$$\text{SR}\big(\text{DIAL-COMB}(x), s\big) = r \equiv r = \text{“OK”} \tag{26}$$

$$\text{SR}\big(\text{READ}(x), s\big) = r \equiv r = \text{INFO}(x, s) \tag{27}$$

In other words, the effect of $\text{READ}(x)$ is to make $\text{INFO}(x)$ known.

Note that sentence (18), along with sentence (23), and sentence (27), and also the fact that the READ action is possible in S_0, ensure that the axiomatization entails

$$\exists x\ \textbf{Knows}\big(\text{INFO}(\text{PPR}) = x, \text{DO}\big(\text{READ}(\text{PPR}), S_0\big)\big) \tag{28}$$

Since the axiomatization of S_0 includes $\textbf{Knows}(\text{INFO}(\text{PPR}) = \text{COMB}(\text{SF}), S_0)$, sentences (23), (24), (27), and (18) ensure that

$$\textbf{Knows}\big(\text{INFO}(\text{PPR}) = \text{COMB}(\text{SF}), \text{DO}\big(\text{READ}(\text{PPR}), S_0\big)\big) \tag{29}$$

also holds. Therefore, the axiomatization entails

$$\textbf{Kref}\big(\text{COMB}(\text{SF}), \text{DO}\big(\text{READ}(\text{PPR}), S_0\big)\big) \tag{30}$$

by (28), (29) and the properties of equality.

Since the successor state axioms ensure that a READ action does not change AT and SAFE, it is the case that the axiomatization entails

$$\text{POSS}\big(\text{DIAL-COMB}(\text{SF}), \text{DO}\big(\text{READ}(\text{PPR}), S_0\big)\big).$$

Therefore, the sequence of actions is a possible sequence. Furthermore, the successor state axiom for OPEN (sentence (20)) ensures that

$$\text{OPEN}\big(\text{SF}, \text{DO}\big(\text{DIAL-COMB}(\text{SF}), \text{DO}\big(\text{READ}(\text{PPR}), S_0\big)\big)\big)$$

holds. In other words, after reading what is on the paper, the safe can be opened by dialing its combination.

6. Correctness of the solution

Once knowledge is introduced and the axiomatization of the effects of actions includes both actions that change the world and actions that change knowledge, we must address the analogue of the ordinary frame problem as discussed in Section 2. This amounts to demonstrating that our approach to the frame problem developed in the previous sections ensures that knowledge only changes as appropriate.

We prove five theorems demonstrating that our axiomatization does in fact ensure that knowledge only changes as appropriate. Next an example is given illustrating some of the subtlety in these results. Finally, we point out that our results readily generalize to knowledge of complex formulas.

6.1. Theorems

We now state and prove five theorems concerning change in knowledge. In the following, P will be used to represent an arbitrary literal, i.e., a fluent (including equality) or its negation.

The first theorem shows that knowledge-producing actions do not change the state of the world. The only fluent whose truth value is altered by a knowledge-producing action is K.

Theorem 1 (Knowledge-producing effects). *For all situations s, all fluents* P (*other than* K) *and knowledge-producing action terms α, if* $\mathrm{P}(s)$ *then* $\mathrm{P}(\mathrm{DO}(\alpha, s))$.

Proof. Immediate from having successor state axioms for each fluent and the axiomatization policy. □

It is also necessary to show that actions only affect knowledge in the appropriate way. The truth of the following theorem ensures that there are no unwanted increases in knowledge. Informally, nothing is learned about a fluent P by doing action α, as long as α does not affect P and in case α is a knowledge-producing action, α does not provide information about something known to be related to P. More formally, we have:

Theorem 2 (Default persistence of ignorance). *For an action α and a situation s, if* $\neg\mathbf{Knows}(\mathrm{P}, s)$ *holds and the axiomatization entails*

$$\forall s\ \mathrm{P}(s) \equiv \mathrm{P}\big(\mathrm{DO}(\alpha, s)\big)$$

and

$$\forall y\ \neg\mathbf{Knows}\Big(\big(\mathrm{POSS}(\alpha) \wedge \mathrm{SR}(\alpha) = y\big) \rightarrow \mathrm{P}, s\Big)$$

then

$$\neg\mathbf{Knows}\big(\mathrm{P}, \mathrm{DO}(\alpha, s)\big)$$

holds as well.

Proof. We are given

$$\forall y\ \neg\mathbf{Knows}\Big(\big(\mathrm{POSS}(\alpha) \wedge \mathrm{SR}(\alpha) = y\big) \rightarrow \mathrm{P}, s\Big).$$

This is an abbreviation for

$$\forall y\ \exists s'\ \mathrm{K}(s', s) \wedge \mathrm{POSS}(\alpha, s') \wedge \mathrm{SR}(\alpha, s') = y \wedge \neg\mathrm{P}(s').$$

Therefore

$$\exists s'\ \mathrm{K}(s', s) \wedge \mathrm{POSS}(\alpha, s') \wedge \mathrm{SR}(\alpha, s') = \mathrm{SR}(\alpha, s) \wedge \neg\mathrm{P}(s')$$

holds. By sentence (18), and the fact that $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s')$, and $\mathrm{POSS}(\alpha, s')$ hold, $\mathrm{K}(\mathrm{DO}(\alpha, s'), \mathrm{DO}(\alpha, s))$ must hold as well. The sentence $\neg\mathrm{P}(\mathrm{DO}(\alpha, s'))$ must hold since $\forall s\ \mathrm{P}(s) \equiv \mathrm{P}(\mathrm{DO}(\alpha, s))$ holds. So, there is an s'' such that $\mathrm{K}(s'', \mathrm{DO}(\alpha, s))$ and $\neg\mathrm{P}(s'')$ hold. Therefore $\neg\mathbf{Knows}(\mathrm{P}, \mathrm{DO}(a, s))$ holds. □

The next theorem shows that agents know the consequences of knowledge acquired through knowledge-producing actions. Informally, if α is a knowledge-producing action that determines whether or not a fluent F is true, and F is true in s, and

$$\mathbf{Knows}(\mathrm{F} \rightarrow \mathrm{P}, s)$$

holds then

$$\mathbf{Knows}\big(\mathrm{P}, \mathrm{DO}(\alpha, s)\big)$$

holds as well even though

$$\neg\mathbf{Knows}(\mathrm{P}, s)$$

is true.

Theorem 3 (Knowledge incorporation). *For a knowledge-producing action α, a fluent or the negation of a fluent* F*, a fluent or the negation of a fluent* P*, and a situation s, if the axiomatization entails*

$$\exists y\ \mathbf{Knows}\big(\mathrm{F} \equiv \mathrm{SR}(\alpha) = y, s\big)$$

and also

$$\mathrm{F}(s), \quad \mathrm{POSS}(\alpha, s),$$

and

$$\mathbf{Knows}(\mathrm{F} \rightarrow \mathrm{P}, s)$$

hold, then

$$\mathbf{Knows}\big(\mathrm{P}, \mathrm{DO}(\alpha, s)\big)$$

holds as well.

Proof. We are given that

$$\exists y\ \mathbf{Knows}\big(\mathrm{F} \equiv \mathrm{SR}(\alpha) = y, s\big)$$

is entailed. This is an abbreviation for

$$\exists y\ \forall s'\ \mathrm{K}(s', s) \rightarrow \mathrm{F}(s') \equiv \mathrm{SR}(\alpha, s') = y$$

Since K is reflexive (i.e., the axiomatization entails $\forall s\ \mathrm{K}(s, s)$), and $\mathrm{F}(s)$ holds, we conclude that $\mathrm{SR}(\alpha, s)$ is equal to the very same y. Then by sentence (18), the fact that $\mathrm{POSS}(\alpha, s)$ holds, and since our axiomatization policy ensures that

$$\forall s\ \mathrm{F}\big(\mathrm{DO}(\alpha, s)\big) \equiv \mathrm{F}(s)$$

holds, we conclude that for every $\mathrm{DO}(\alpha, s')$ such that $\mathrm{K}(\mathrm{DO}(\alpha, s'), \mathrm{DO}(\alpha, s))$ holds, $\mathrm{F}(\mathrm{DO}(\alpha, s'))$ holds and also both $\mathrm{F}(s')$ and $\mathrm{K}(s', s)$ hold as well.

Furthermore, since $\mathbf{Knows}(\mathrm{F} \rightarrow \mathrm{P}, s)$ holds, and is an abbreviation for

$$\forall s'\ \mathrm{K}(s', s) \wedge \mathrm{F}(s') \rightarrow \mathrm{P}(s')$$

we conclude that in every s' such that both $\mathrm{K}(s', s)$ and $\mathrm{F}(s')$ hold, $\mathrm{P}(s')$ must hold as well. Since by the axiomatization policy

$$\forall s\ \mathrm{P}\big(\mathrm{DO}(\alpha, s)\big) \equiv \mathrm{P}(s)$$

must be true, it is also the case that in every $\mathrm{DO}(\alpha, s')$ such that $\mathrm{K}(\mathrm{DO}(\alpha, s'), \mathrm{DO}(\alpha, s))$ holds, $\mathrm{P}(\mathrm{DO}(\alpha, s')$ holds as well. Therefore, the sentence **Knows**$(\mathrm{P}, \mathrm{DO}(a, s))$ must hold. □

Additionally, it is a property of this specification that agents never forget. Informally speaking, if the agent knows P at s, then P is also known at $\mathrm{DO}(\alpha, s)$ as long as the effect of α is not to make P false. We have the following theorem:

Theorem 4 (Memory). *For all fluents* P *and situations* s*, if* **Knows**(P, s) *holds then* **Knows**$(\mathrm{P}, \mathrm{DO}(\alpha, s))$ *holds as long as the axiomatization entails*

$$\forall s\ \mathrm{P}(s) \equiv \mathrm{P}\big(\mathrm{DO}(\alpha, s)\big)$$

Proof. For **Knows**(P, s) to be true, it must be the case by definition that $\forall s\ \mathrm{K}(s', s) \rightarrow \mathrm{P}(s')$ holds. By sentence (18), for all states s'' such that $\mathrm{K}(s'', \mathrm{DO}(\alpha, s))$ holds, it is the case that $s'' = \mathrm{DO}(\alpha, s')$ for some s' such that $\mathrm{K}(s', s)$ is true. Since $\mathrm{P}(s')$ is true for each of these s', and since the axioms entail

$$\forall s\ \mathrm{P}(s) \equiv \mathrm{P}\big(\mathrm{DO}(\alpha, s)\big)$$

we have that $\mathrm{P}(\mathrm{DO}(\alpha, s'))$ is true. Thus for any situation s'' such that $K(s'', \mathrm{DO}(\alpha, s))$ holds, $\mathrm{P}(s'')$ is true. Therefore **Knows**$(\mathrm{P}, \mathrm{DO}(\alpha, s))$ is true. □

Finally, agents know the effects of actions.

Theorem 5 (Knowledge of effects of actions). *If* α *is an ordinary* (*not a knowledge-producing*) *action, and if the axiomatization entails*

$$\forall s\ \phi[s] \rightarrow \mathrm{P}\big(\mathrm{DO}(\alpha, s)\big)$$

where ϕ *is an arbitrary formula with situation terms suppressed*[8] *and* P *is a fluent or its negation, then the following is also entailed*:

$$\mathbf{Knows}\big(\big(\mathrm{POSS}(\alpha) \wedge \phi\big), s\big) \rightarrow \mathbf{Knows}\big(\mathrm{P}, \mathrm{DO}(\alpha, s)\big)$$

Proof. The sentence

$$\mathbf{Knows}\big(\mathrm{P}, \mathrm{DO}(\alpha, s)\big)$$

is an abbreviation for

$$\forall s''\ \mathrm{K}\big(s'', \mathrm{DO}(\alpha, s)\big) \rightarrow P(s'')$$

Sentence (18) requires that each s'' be equal to $\mathrm{DO}(\alpha, s')$ for an s' such that both $\mathrm{K}(s', s)$ and $\mathrm{POSS}(\alpha, s')$ hold. If

$$\mathbf{Knows}\big(\big(\mathrm{POSS}(\alpha) \wedge \phi\big), s\big)$$

[8] When ϕ is an arbitrary sentence and s a situation term, then $\phi[s]$ is the sentence that results from adding an extra argument to every fluent of ϕ and inserting s into that argument position.

holds, then certainly $\phi[s']$ and $\text{POSS}(\alpha, s')$ must hold for each s' such that $\text{K}(s', s)$. Since the situation variable in

$$\forall s\ \phi[s] \rightarrow \text{P}\big(\text{DO}(\alpha, s)\big)$$

is universally quantified, $\text{P}(s'')$ must hold for each s'' such that $\text{K}(s'', \text{DO}(\alpha, s))$. Therefore

$$\textbf{Knows}\big(\big(\text{POSS}(\alpha) \wedge \phi\big), s\big) \rightarrow \textbf{Knows}\big(\text{P}, \text{DO}(\alpha, s)\big)$$

holds. □

Consider again the successor state axiom for BROKEN given in sentence (8). If $\textbf{Knows}(\neg\text{BROKEN}(\text{OBJ}_1), \text{S}_0)$ is true, then

$$\textbf{Knows}\big(\neg\text{BROKEN}(\text{OBJ}_1), \text{DO}\big(\text{DROP}(\text{OBJ}_2), \text{S}_0\big)\big)$$

must also be true. Also, note that if

$$\textbf{Knows}\big(\text{FRAGILE}(\text{OBJ}_2), \text{S}_0\big) \quad \text{and} \quad \textbf{Knows}\big(\text{POSS}\big(\text{DROP}(\text{OBJ}_2)\big), \text{S}_0\big)$$

are true, then

$$\textbf{Knows}\big(\text{BROKEN}(\text{OBJ}_2), \text{DO}\big(\text{DROP}(\text{OBJ}_2), \text{S}_0\big)\big)$$

must also be true.

6.2. Example

We give an example[9] below to illustrate some of the possibly surprising subtlety involved in our axiomatization; in particular Theorem 3. Agents know the consequences of what they sense given their total body of knowledge prior to the sensing.

Consider the case of a sensing action that enables the agent to determine which of two objects is larger. The name of the action is COMPARE. We have a fluent $\text{LG\_THAN}(x, y, s)$ to represent[10] x being larger than y in situation s.

We assume that there are no actions that change LG_THAN. So, the successor state axiom is as follows:

$$\text{LG\_THAN}\big(x, y, \text{DO}(a, s)\big) \equiv \text{LG\_THAN}(x, y, s)$$

The SR axiom for COMPARE is given below:

$$\begin{aligned}&\text{SR}\big(\text{COMPARE}(x, y), s\big) = r \equiv \\ &\quad \big(r = \text{“YES”} \wedge \text{LG\_THAN}(x, y, s)\big) \vee \big(r = \text{“NO”} \wedge \neg\text{LG\_THAN}(x, y, s)\big)\end{aligned} \tag{31}$$

It specifies that after doing the $\text{COMPARE}(x, y)$ action, the agent knows whether or not x is larger than y.

The axiomatization of the initial situation is as follows:

[9] The example was kindly suggested by Zenon Pylyshyn.

[10] An alternative approach would be to define LG_THAN in terms of a functional fluent SIZE, sorted to take numerical values.

$$\mathbf{Knows}\big(\text{LG\_THAN}(\text{OBJ}_1, \text{OBJ}_2), \text{S}_0\big)$$

$$\text{LG\_THAN}(\text{OBJ}_2, \text{OBJ}_3, \text{S}_0)$$

$$\mathbf{Knows}\big(\text{LG\_THAN}(x, y) \wedge \text{LG\_THAN}(y, z) \rightarrow \text{LG\_THAN}(x, z), \text{S}_0\big)$$

The agent knows that OBJ_1 is larger than OBJ_2. Additionally, it is the case that OBJ_2 is larger than OBJ_3, but the agent need not know that this is the case. Furthermore, the agent knows that LG_THAN is transitive.

For simplicity, we also ensure that the agent knows that no two objects are the same size.

$$\mathbf{Knows}\big(\text{LG\_THAN}(x, y) \rightarrow \neg\text{LG\_THAN}(y, x), \text{S}_0\big)$$

So, the following LG_THAN relations are compatible with the agent's knowledge:

1. $\text{LG\_THAN}(\text{OBJ}_1, \text{OBJ}_2, \text{S}_0)$, $\text{LG\_THAN}(\text{OBJ}_1, \text{OBJ}_3, \text{S}_0)$, $\text{LG\_THAN}(\text{OBJ}_2, \text{OBJ}_3, \text{S}_0)$
2. $\text{LG\_THAN}(\text{OBJ}_1, \text{OBJ}_2, \text{S}_0)$, $\text{LG\_THAN}(\text{OBJ}_3, \text{OBJ}_2, \text{S}_0)$, $\text{LG\_THAN}(\text{OBJ}_1, \text{OBJ}_3, \text{S}_0)$
3. $\text{LG\_THAN}(\text{OBJ}_1, \text{OBJ}_2, \text{S}_0)$, $\text{LG\_THAN}(\text{OBJ}_3, \text{OBJ}_2, \text{S}_0)$, $\text{LG\_THAN}(\text{OBJ}_3, \text{OBJ}_1, \text{S}_0)$

Now note that the axiomatization entails:

$$\mathbf{Knows}\big(\text{LG\_THAN}(\text{OBJ}_1, \text{OBJ}_3), \text{DO}\big(\text{COMPARE}(\text{OBJ}_2, \text{OBJ}_3), \text{S}_0\big)\big)$$

After doing the action $\text{COMPARE}(\text{OBJ}_2, \text{OBJ}_3)$, the agent now not only knows that OBJ_2 is larger than OBJ_3, but also that OBJ_1 is larger than OBJ_3. Only possibility 1 remains. Both 2 and 3 have been eliminated.

6.3. Generalization of the effects of knowledge-producing actions

In the discussion so far and in Theorems 1–5, we have assumed that the effect of *sense* type actions is to make true a formula consisting of **Kwhether** with the argument being a single fluent. But nothing hinged on this restriction.

The effect of a *sense* type action may be axiomatized to determine the truth value of a complex formula. For example, the effect of a SENSE-COMPLEX action performed by a robot [16,18] may be specified as follows:

$$\begin{aligned}\mathbf{Kwhether}\big(&\exists x\ \big(\text{OBJECT}(x) \wedge \text{HOLDING}(x)\\ &\wedge \text{OFSHAPE}(x, \text{SHAPE}_1)\big), \text{DO}(\text{SENSE-COMPLEX}, \text{S}_0)\big)\end{aligned} \tag{32}$$

We can readily define a SR axiom as follows:

$$\begin{aligned}&\text{SR}(\text{SENSE-COMPLEX}, s) = r \equiv\\ &\quad\big(r = \text{“YES”} \wedge \big(\exists x\ \text{OBJECT}(x, s) \wedge \text{HOLDING}(x, s)\\ &\quad\quad\wedge \text{OFSHAPE}(x, \text{SHAPE}_1, s)\big)\big) \vee\\ &\quad\big(r = \text{“NO”} \wedge \neg\big(\exists x\ \text{OBJECT}(x, s) \wedge \text{HOLDING}(x, s)\\ &\quad\quad\wedge \text{OFSHAPE}(x, \text{SHAPE}_1, s)\big)\big)\end{aligned} \tag{33}$$

Additionally, our demonstration that knowledge only changes as appropriate holds for knowledge of formulas as well. By simply replacing the literal P (and also F in the case of Theorem 3) with an arbitrary formula, Theorems 1–5 and their proofs generalize

immediately to formulas as well. So, the approach to the frame problem for knowledge-producing actions is correct for knowledge understood as the knowledge of formulas.

7. Knowledge of arbitrary formulas

Previously, we argued that allowing complex formulas as arguments to the **Knows** operator does not affect our theory at all as long as the formulas do not include the K fluent. Now, we relax this restriction and consider the case where the arguments to the **Knows** operator may be arbitrary formulas, in particular those that include the K fluent.

In other words, now we are considering the case of nested **Knows** operators. The situation argument of the operator is then understood contextually. If it is not the outermost operator, the situation argument is understood to be the first argument of the immediately dominating K atom. For example, (34) is understood as an abbreviation for (35).

$$\mathbf{Knows}\big(\mathbf{Knows}(\mathrm{P}), \mathrm{S}_0\big) \tag{34}$$

$$\forall s_1\ \mathrm{K}(s_1, \mathrm{S}_0) \rightarrow \big(\forall s_2\ \mathrm{K}(s_2, s_1) \rightarrow \mathrm{P}(s_2)\big) \tag{35}$$

The only remaining issue concerns requiring that the **Knows** operator conforms to the properties of a particular modal logic. For example, if the logic chosen is *S*4, then we want positive introspection (sentence (36)) to be a property of the logic.

$$\mathbf{Knows}(\phi, s) \rightarrow \mathbf{Knows}\big(\mathbf{Knows}(\phi), s\big) \tag{36}$$

Restrictions need to be placed on the K relation so that it correctly models the accessibility relation of a particular modal logic. The problem is to do this in a way that does not interfere with the successor state axioms for K, which must completely specify the K relation for non-initial situations. The solution is to axiomatize the restrictions for the initial situation and then verify that the restrictions are then obeyed at all situations.

The sort INIT is used to restrict variables to range only over S_0 and those situations accessible from S_0. It is necessary to stipulate that:

$$\mathrm{INIT}(\mathrm{S}_0)$$

$$\forall s, s_1\ \mathrm{INIT}(s_1) \rightarrow \big(\mathrm{K}(s, s_1) \rightarrow \mathrm{INIT}(s)\big)$$

$$\forall s, s_1\ \neg\mathrm{INIT}(s_1) \rightarrow \big(\mathrm{K}(s, s_1) \rightarrow \neg\mathrm{INIT}(s)\big)$$

$$\mathrm{INIT}(s) \rightarrow \neg\exists s'\ \big(s = \mathrm{DO}(a, s')\big)$$

We want to require that the situation S_0 is a member of the sort INIT, everything K-accessible from an INIT situation is also INIT, and that everything K-accessible from a a situation that is not INIT is also not INIT. Also it is necessary to require that none of the situations that result from the occurrence of an action are INIT.

Given the decision that we are to use a particular modal logic of knowledge, it is necessary to axiomatize the corresponding restrictions that need to be placed on the K relation. These are listed below[11] and are merely first-order representations of the

[11] $\forall s$: INIT φ is an abbreviation for $\forall s\ \mathrm{INIT}(s) \rightarrow \varphi$.

conditions on the accessibility relations for the standard modal logics of knowledge discussed in the literature [3,4,12,13]. The reflexive restriction is always added as we want a modal logic of knowledge. Some subset of the other restrictions are then added to semantically define a particular modal logic.

Reflexive $\forall s_1{:}\,\text{INIT}\ \text{K}(s_1, s_1)$

Euclidian $\forall s_1{:}\,\text{INIT}, s_2{:}\text{INIT}, s_3{:}\text{INIT}$
$\text{K}(s_2, s_1) \wedge \text{K}(s_3, s_1) \rightarrow \text{K}(s_3, s_2)$

Symmetric $\forall s_1{:}\,\text{INIT}, s_2{:}\text{INIT}\ \text{K}(s_2, s_1) \rightarrow \text{K}(s_1, s_2)$

Transitive $\forall s_1{:}\,\text{INIT}, s_2{:}\text{INIT}, s_3{:}\text{INIT}$
$\text{K}(s_2, s_1) \wedge \text{K}(s_3, s_2) \rightarrow \text{K}(s_3, s_1)$

To model the logic *S*4, for example, one would need to include the axioms for both reflexivity and transitivity.

The next step is to prove that if the K relation over the initial situations satisfies a particular restriction *R*, that restriction *R* will also hold over the other situations as well.

Theorem 6. *If the* K *relation on the set of initial situations is restricted to conform to the reflexive condition along with some subset of the symmetric, transitive and Euclidean properties, then the* K *relation at every situation, resulting from the execution of a sequence of possible actions* (*as defined by* POSS)*, will satisfy the same set of properties.*

Proof. For each of the restrictions,[12] it is only necessary to prove that if the restriction holds at s then it must also hold at $\text{DO}(\alpha, s)$ for any action α as long as $\text{POSS}(\alpha, s)$ is true. Since every ground situation term is constructed out of S_0 and a finite number of DO function symbols and action terms, and since the restriction holds at S_0, the restriction must hold at every situation that results form a possible sequence of actions.[13]

- *Reflexive*. Assume that $\text{K}(s, s)$ holds for situation s. Then we need to show that $\text{K}(\text{DO}(\alpha, s), \text{DO}(\alpha, s))$ must hold for all actions α such that $\text{POSS}(\alpha, s)$ is true. By the form of the successor state axiom for K (sentence (18)), and the fact that for all s $\text{SR}(\alpha, s) = \text{SR}(\alpha, s)$ holds, and also that $\text{POSS}(\alpha, s) \equiv \text{POSS}(\alpha, s)$ holds, and the assumption $\text{K}(s, s)$, $\text{K}(\text{DO}(\alpha, s), \text{DO}(\alpha, s))$ must be true.
- *Symmetric*. Assume that $\forall s'\ \text{K}(s', s) \rightarrow \text{K}(s, s')$ holds for s. Then we must show that $\forall s''\ \text{K}(s'', \text{DO}(\alpha, s)) \rightarrow \text{K}(\text{DO}(\alpha, s), s'')$ holds for $\text{DO}(\alpha, s)$ for all α such that $\text{POSS}(\alpha, s)$ is true. For $\forall s''\ \text{K}(s'', \text{DO}(\alpha, s)) \rightarrow \text{K}(\text{DO}(\alpha, s), s'')$ to be false it must

[12] As an example of a restriction that is not guaranteed to hold at every level, even if it does hold over the set of initial situations, consider the condition of being serial:

$$\forall s\ \exists s'\ \text{K}(s', s)$$

Note that assuming the restriction holds at s, it is not guaranteed to hold at $\text{DO}(a, s)$, since it is possible that $\text{SR}(a, s) \neq \text{SR}(a, s')$.

[13] Alternatively, we can appeal to the induction principle for the situation calculus described in [21,34].

be the case that for some s'', and α, $\mathrm{K}(s'', \mathrm{DO}(\alpha, s))$ is true, but $\mathrm{K}(\mathrm{DO}(\alpha, s), s'')$ is false. Note that by the successor state axiom for K (sentence (18)), the only way for $\mathrm{K}(s'', \mathrm{DO}(\alpha, s))$ to be true is for s'' to be equal to $\mathrm{DO}(\alpha, s')$ for some s', for $\mathrm{K}(s', s)$ to be true, and for $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s')$ to be true. Also note that both $\mathrm{POSS}(\alpha, s)$ and $\mathrm{POSS}(\alpha, s')$ must hold. By assumption, if $\mathrm{K}(s', s)$ is true, then $\mathrm{K}(s, s')$ must hold as well. Then by the successor state axiom for K (sentence (18)) and the symmetry of $=$, it must also be the case that $\mathrm{K}(\mathrm{DO}(\alpha, s), s'')$ is true. Therefore $\forall s''\ \mathrm{K}(s'', \mathrm{DO}(\alpha, s)) \rightarrow \mathrm{K}(\mathrm{DO}(\alpha, s), s'')$ must hold.

- *Transitive*. Assume that

$$\forall s_1, s_2\ \mathrm{K}(s_1, s) \wedge \mathrm{K}(s_2, s_1) \rightarrow \mathrm{K}(s_2, s)$$

holds at s. Then we must show that

$$\forall s_3, s_4\ \mathrm{K}\big(s_3, \mathrm{DO}(\alpha, s)\big) \wedge \mathrm{K}(s_4, s_3) \rightarrow \mathrm{K}\big(s_4, \mathrm{DO}(\alpha, s)\big)$$

holds at $\mathrm{DO}(\alpha, s)$ for every α such that $\mathrm{POSS}(\alpha, s)$ is true. For $\forall s_3\, s_4\ \mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, s_3) \rightarrow \mathrm{K}(s_4, \mathrm{DO}(\alpha, s))$ to be false, it must be the case that for some s_3 and s_4, $\mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, s_3)$ is true, but $\mathrm{K}(s_4, \mathrm{DO}(\alpha, s))$ is false. Note that by the successor state axiom for K (sentence (18)), the only way for $\mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, s_3)$ to be true is for s_3 to be equal to $\mathrm{DO}(\alpha, s_3^*)$ for some s_3^* and for $\mathrm{K}(s_3^*, s)$ to be true, and for s_4 to be equal to $\mathrm{DO}(\alpha, s_4^*)$ for some s_4^* and for $\mathrm{K}(s_4^*, s_3^*)$ to be true, and for $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s_3^*)$ and $\mathrm{SR}(\alpha, s_3^*) = \mathrm{SR}(\alpha, s_4^*)$ to be true. Also note that $\mathrm{POSS}(\alpha, s)$, $\mathrm{POSS}(\alpha, s_3^*)$, and $\mathrm{POSS}(\alpha, s_4^*)$ must hold. Then it must also, be the case that $\mathrm{K}(s_4^*, s)$ is true by assumption. Note that (since $=$ is transitive) if $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s_3^*)$ and $\mathrm{SR}(\alpha, s_3^*) = \mathrm{SR}(\alpha, s_4^*)$, then $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s_4^*)$. Then by the successor state axiom for K (sentence (18)), it must also be the case that $\mathrm{K}(\mathrm{DO}(\alpha, s_4^*), \mathrm{DO}(\alpha, s))$ is true. Therefore $\forall s_3 s_4\ \mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, s_3) \rightarrow \mathrm{K}(s_4, \mathrm{DO}(\alpha, s))$ must hold.

- *Euclidean*. Assume that

$$\forall s_1, s_2\ \big(\mathrm{K}(s_1, s) \wedge \mathrm{K}(s_2, s)\big) \rightarrow \mathrm{K}(s_2, s_1)$$

holds at s. Then it is necessary to show that

$$\forall s_3, s_4\ \big(\mathrm{K}\big(s_3, \mathrm{DO}(\alpha, s)\big) \wedge \mathrm{K}\big(s_4, \mathrm{DO}(\alpha, s)\big)\big) \rightarrow \mathrm{K}(s_4, s_3)$$

holds at $\mathrm{DO}(\alpha, s)$ for each α such that $\mathrm{POSS}(\alpha, s)$ is true.

For $\forall s_3 s_4\ \mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, \mathrm{DO}(\alpha, s)) \rightarrow \mathrm{K}(s_4, s_3)$ to be false, it must be the case that for some α, s_3 and s_4, $\mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, \mathrm{DO}(\alpha, s))$ is true, but $\mathrm{K}(s_4, s_3)$ is false. Note that by the successor state axiom for K (sentence (18)), the only way for $\mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, \mathrm{DO}(\alpha, s))$ to be true is for s_3 to be equal to $\mathrm{DO}(\alpha, s_3^*)$ for some s_3^* and for $\mathrm{K}(s_3^*, s)$ to be true, and for s_4 to be equal to $\mathrm{DO}(\alpha, s_4^*)$ for some s_4^* and for $\mathrm{K}(s_4^*, s)$ to be true, and for $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s_3^*)$ and $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s_4^*)$ to be true. Also note that $\mathrm{POSS}(\alpha, s)$, $\mathrm{POSS}(\alpha, s_3^*)$, and $\mathrm{POSS}(\alpha, s_4^*)$ must hold. Then it must also be the case that $\mathrm{K}(s_4^*, s_3^*)$ is true by assumption. Note that (since $=$ is in effect Euclidean) if $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s_3^*)$ and $\mathrm{SR}(\alpha, s) = \mathrm{SR}(\alpha, s_4^*)$, then $\mathrm{SR}(\alpha, s_3^*) = \mathrm{SR}(\alpha, s_4^*)$. Then by sentence (18), it must also be the case that $\mathrm{K}(\mathrm{DO}(\alpha, s_4^*), \mathrm{DO}(\alpha, s_3^*))$ is true. Therefore the sentence $\forall s_3\ s_4\ \mathrm{K}(s_3, \mathrm{DO}(\alpha, s)) \wedge \mathrm{K}(s_4, \mathrm{DO}(\alpha, s)) \rightarrow \mathrm{K}(s_4, s_3)$ must hold. □

The significance of this theorem is that if the K relation at the initial situation is defined as satisfying certain conditions, then the K relation at all situations reachable by "executing actions" starting in the initial situation, also satisfy those properties. So, if we decide to use, for example, the logic $S4$ to model knowledge, we can go ahead and stipulate that the K relation at the initial situation is reflexive and transitive. Then we are guaranteed that the relation at all reachable situations will also satisfy those properties and our model of knowledge will remain $S4$, without danger of conflicting with the successor state axioms.

8. Reasoning

Given the representation of actions and their effects, we would like to have a method for addressing the *projection problem* [33]. This is the question of determining whether or not some sentence G is true in the situation resulting from the execution of an action sequence $\alpha_1, \ldots, \alpha_n$ of ground action terms. This question is represented as the query

$$\mathcal{F} \models G\big(\mathrm{DO}\big([\alpha_1, \ldots, \alpha_n], \mathrm{S}_0\big)\big),$$

where $\mathcal{F}$ is the axiomatization of actions and their effects, and the initial situation. The expression $\mathrm{DO}([a_1, \ldots, a_n], s)$ is a compact notation for the situation term

$$\mathrm{DO}\big(a_n, \mathrm{DO}\big(a_{n-1}, \ldots, \mathrm{DO}(a_1, s)\ldots\big)\big),$$

which denotes the state resulting from performing the action a_1, followed by $a_2, \ldots$, followed by a_n, beginning in situation s.

One method of answering the query is to translate the axiomatization of the initial situation and the query G (both of which contain the modal operator **Knows**) into first-order logic using the well known method implicit in much of the discussion here and used by Moore [27,28]. Then any first-order theorem proving method can be used to query a particular axiomatization. It would be much more efficient to use a method designed for performing automated deduction in modal logics, but sentence (18) has no such representation in a modal logic of knowledge.[14] Note that the equivalences in the successor state axioms would expand into a very large number of clauses yielding a large search space if resolution theorem proving were used to reason about the effects of actions. Therefore, the utilization of a specialized method for reasoning with the successor-state axioms is important from an efficiency standpoint whether or not knowledge-producing actions are involved.

Here, regression operators are utilized to address the projection problem. Regression is also used to determine whether or not the sequence of actions $\alpha_1, \ldots, \alpha_n$ is an executable sequence of actions resulting in a *legal* situation. Then a modal theorem proving method can be used to determine whether or not the result of regression is entailed by the axioms of the initial state.

[14] In fact, the results reported in [7,40] indicate that even modal deduction methods that can be understood as reasoning with the K literals representing the accessibility relation, cannot be easily modified to handle a sentence like (18).

We first develop a full set of regression steps, building upon the work of Reiter. Next, we illustrate the use of regression to address the projection problem. Finally, we illustrate the use of regression to determine the legality of a sequence of actions.

8.1. Regression

Reiter [32] develops a form of *regression* to reduce reasoning about future situations to reasoning about the initial situation. The basic idea of regression is that a formula G' is the regression of G over action a if and only if G' is the weakest condition such that if G' is true before a, then G will be true after a.

In this section, a regression operator is developed for knowledge-producing actions and applied to the problem of determining whether or not a particular plan satisfies a particular property. So given a plan, expressed as a ground situation term (i.e., a term built on S_0 with the function *do* and ground action terms) s_{gr}, the question is whether the axiomatization of the domain $\mathcal{F}$ entails $G(s_{gr})$ where G is an arbitrary sentence including modal operators. Under these circumstances, the successor state axioms (including (18)) are used only to regress the formula $G(s_{gr})$. The result of the regression is a formula in ordinary modal logic, i.e., a formula without action terms and where the only situation term is S_0. Then an ordinary modal theorem proving method (e.g., those discussed in [7,40]) may be used to determine whether or not the regressed formula holds. In what follows, it is assumed that the formulas do not use the fluent K except as abbreviated by **Knows**.

The regression operator $\mathcal{R}$ is defined relative to a set of successor state axioms Θ. Parts (i), (ii)a, b, c, (iii), and (iv) of the definition of the regression operator $\mathcal{R}_\Theta$ concern ordinary (i.e., not knowledge-producing) actions [29,32]. Note that step (ii)d is concerned with the regression of SR. It is assumed that the function symbol SR will only be found initially in the SR axioms and in the successor state axiom for K, but it is introduced by part (vi) (to be discussed shortly) of the regression definition. Also, note that we require that functional fluents only occur as arguments to the equality literal. This restriction does not lead to any loss in generality as a functional fluent as an argument to some other predicate symbol can be eliminated by introducing an existentially quantified variable in that position and an equality symbol setting the functional fluent equal to the variable. The first four parts of the definition of the regression operator are given below:

(i) When A is a non-fluent atom, including equality atoms without functional fluents as arguments; or when A is a fluent atom, or **Knows** operator, whose situation argument is the situation constant S_0, $\mathcal{R}_\Theta[A] = A$.

(ii) There are four cases here. Case (a) covers ordinary (non-functional) fluents. Case (b) covers equality literals with at least one functional fluent as an argument. Case (c) covers the regression of POSS literals, and case (d) covers the regression of equality literals involving the SR functional fluent.

(a) When F is a fluent (other than K) whose successor state axiom in Θ is

$$\left[\mathrm{F}\left(x_1, \ldots, x_n, \mathrm{DO}(a, s)\right) \equiv \Phi_F\right]$$

then

$$\mathcal{R}_\Theta\left[\mathrm{F}\left(t_1, \ldots, t_n, \mathrm{DO}(\alpha, \sigma)\right)\right] = \Phi_{\mathrm{F}}|^{x_1,\ldots,x_n,a,s}_{t_1,\ldots,t_n,\alpha,\sigma}$$

In other words, the atom $\mathrm{F}(t_1, \ldots, t_n, \mathrm{DO}(\alpha, \sigma))$ is replaced by the appropriate instance of the right-hand side of the equivalence in F's successor state axiom. This instance is created by substituting $t_1, \ldots, t_n, \alpha, \sigma$ for $x_1, \ldots, x_n, a, s$ in the right-hand side of the equivalence.

(b) When the item to be regressed is an equality literal with an argument being the functional fluent F whose successor state axiom in Θ is

$$\left[\mathrm{F}\left(x_1, \ldots, x_n, \mathrm{DO}(\alpha, s)\right) = y \equiv \Phi_{\mathrm{F}}\right]$$

then

$$\mathcal{R}_{\Theta}\left[\mathrm{F}\left(t_1, \ldots, t_n, \mathrm{DO}(\alpha, \sigma)\right) = t_{n+1}\right] = \Phi_{\mathrm{F}}|^{x_1,\ldots,x_n,a,s,y}_{t_1,\ldots,t_n,\alpha,\sigma,t_{n+1}}$$

and

$$\mathcal{R}_{\Theta}\left[t_{n+1} = \mathrm{F}\left(t_1, \ldots, t_n, \mathrm{DO}(\alpha, \sigma)\right)\right] = \Phi_{\mathrm{F}}|^{x_1,\ldots,x_n,a,s,y}_{t_1,\ldots,t_n,\alpha,\sigma,t_{n+1}}$$

In other words, the equality atom with $\mathrm{F}(t_1, \ldots, t_n, \mathrm{DO}(\alpha, \sigma))$ as an argument is replaced by the appropriate instance of the right-hand side of the equivalence in F's successor state axiom. This instance is created by substituting $t_1, \ldots, t_n, \alpha, \sigma, t_{n+1}$ for $x_1, \ldots, x_n, a, s, y$ in the right-hand side of the equivalence.

(c) When the item to be regressed is a $\mathrm{POSS}(a, s)$ literal with the action precondition axiom of the form

$$\mathrm{POSS}\left(\alpha(\vec{x}), s\right) \equiv \Pi_{\alpha}(x_1, \ldots, x_n, s) \tag{37}$$

then

$$\mathcal{R}_{\Theta}\left[\mathrm{POSS}\left(\alpha(t_1, \ldots, t_n), \sigma\right)\right] = \mathcal{R}_{\Theta}\left[\Pi_{\alpha}(x_1, \ldots, x_n, s)|^{x_1,\ldots,x_n,s}_{t_1,\ldots,t_n,\sigma}\right]$$

In other words, the atom $\mathrm{POSS}(\alpha(t_1, \ldots, t_n), \sigma)$ is replaced by the regression of the appropriate instance of the right-hand side of the equivalence in α's action precondition axiom. This instance is created by substituting $t_1, \ldots, t_n, \sigma$ for $x_1, \ldots, x_n, s$ in the right-hand side of the equivalence.

(d) When the item to be regressed is an equality literal with an argument being an SR function with a sensing result axiom of the form:

$$\mathrm{SR}\left(\alpha(\vec{x}), s\right) = r \equiv \phi_{\alpha}(\vec{x}, r, s) \tag{38}$$

then

$$\mathcal{R}_{\Theta}\left[\mathrm{SR}(t_1, \ldots, t_n, \sigma) = t_{n+1}\right] = \mathcal{R}_{\Theta}\left[\phi_{\alpha}|^{x_1,\ldots,x_n,a,s,r}_{t_1,\ldots,t_n,\alpha,\sigma,t_{n+1}}\right]$$

and

$$\mathcal{R}_{\Theta}\left[t_{n+1} = \mathrm{SR}(t_1, \ldots, t_n, \sigma)\right] = \mathcal{R}_{\Theta}\left[\phi_{\alpha}|^{x_1,\ldots,x_n,a,s,r}_{t_1,\ldots,t_n,\alpha,\sigma,t_{n+1}}\right]$$

In other words, an equality atom with $\mathrm{SR}(\alpha(t_1, \ldots, t_n), \sigma)$ as an argument is replaced by the regression of the appropriate instance of the right-hand side of the equivalence in α's sensing result axiom. This instance is created by substituting $t_1, \ldots, t_n, \sigma, t_{n+1}$ for $x_1, \ldots, x_n, s, r$ in the right-hand side of the equivalence.

(iii) Whenever W is a formula, $\mathcal{R}_\Theta[\neg W] = \neg\mathcal{R}_\Theta[W]$, $\mathcal{R}_\Theta[(\forall v)W] = (\forall v)\mathcal{R}_\Theta[W]$, $\mathcal{R}_\Theta[(\exists v)W_1] = (\exists v)\mathcal{R}_\Theta[W_1]$.

(iv) Whenever W_1 and W_2 are formulas, $\mathcal{R}_\Theta[W_1 \wedge W_2] = \mathcal{R}_\Theta[W_1] \wedge \mathcal{R}_\Theta[W_1]$, $\mathcal{R}_\Theta[W_1 \vee W_2] = \mathcal{R}_\Theta[W_1] \vee \mathcal{R}_\Theta[W_1]$, $\mathcal{R}_\Theta[W_1 \rightarrow W_2] = \mathcal{R}_\Theta[W_1] \rightarrow \mathcal{R}_\Theta[W_1]$.

Additional steps are needed to extend the regression operator to knowledge-producing actions. An additional definition is needed for the specification to follow. The result of the operation[15] φ^{-1} is φ, but with the removal of the last argument position from all the fluents in φ.

Step (v) covers the case of regressing the **Knows** operator through a non-knowledge-producing action. Step (vi) covers the case of regressing the **Knows** operator through a knowledge producing action. In the definitions below, s' is a new situation variable.

(v) Whenever a is not a knowledge-producing action,

$$\mathcal{R}_\Theta\big[\mathbf{Knows}\big(\mathrm{W}, \mathrm{DO}(a, s)\big)\big] = \mathbf{Knows}\big(\mathrm{POSS}(a) \rightarrow \mathcal{R}_\Theta\big[\mathrm{W}[\mathrm{DO}(a, s')]\big]^{-1}, s\big).$$

(vi)

$$\begin{aligned}&\mathcal{R}_\Theta\big[\mathbf{Knows}\big(W, \mathrm{DO}(\mathrm{SENSE}_i, s)\big)\big] = \\ &\quad \exists y\ \mathrm{SR}(\mathrm{SENSE}_i, s) = y \wedge \\ &\qquad \mathbf{Knows}\big(\big(\mathrm{POSS}(a) \wedge \mathrm{SR}(\mathrm{SENSE}_i) = y\big) \rightarrow \mathcal{R}_\Theta\big[W[\mathrm{DO}(\mathrm{SENSE}_i)]\big]^{-1}, s\big)\end{aligned}$$

In the following theorem,[16] $\mathcal{F}$ is the axiomatization of the domain including $\mathcal{F}_{ss}$, the successor state axioms. The notation $\mathcal{R}^*_\Theta(\varphi)$ is used to indicate that the regression operator is applied repeatedly until further applications leave the formula unchanged.

Theorem 7. *For any ground situation term* s_{gr}

$$\mathcal{F} \models G(s_{gr}) \quad \textit{iff} \quad \mathcal{F} - \mathcal{F}_{ss} \models \mathcal{R}^*_\Theta\big[G(s_{gr})\big]$$

Proof. It suffices to show that the process of regression preserves logical equivalence given the axiomatization $\mathcal{F}$.

$$\mathcal{F} \models G(s_{gr}) \equiv \mathcal{R}^*_\Theta\big[G(s_{gr})\big]$$

This is done by showing that each step preserves logical equivalence. The process must terminate as every step removes the outer DO from the situation terms and the number of DO function symbols making up any such term is finite. As each step preserves equivalence, the whole process results in an equivalent formula. Since after regression terminates the sentence G does not contain any action terms (i.e., the only situation

[15] Recall the reverse operation. When φ is an arbitrary sentence and s a situation term, then $\varphi[s]$ is the sentence that results from adding an extra argument to every fluent of φ and inserting s into that argument position.

[16] We assume along with [31,35] that the functional fluent consistency property holds. This property ensures that the conditions defining a functional fluent's value in the next situation $\mathrm{DO}(\alpha, s)$ define a unique value for the fluent. If this condition did not hold, there would be a source of inconsistency in the successor state axioms and we could not remove the axioms without making the theory consistent.

term is S_0), the successor state axioms are no longer needed to determine whether or not $\mathcal{F} \models \mathcal{R}^*_{\Theta}[G(s_{gr})]$. Therefore $\mathcal{F} \models \mathcal{R}^*_{\Theta}[G(s_{gr})]$ if and only if $\mathcal{F} - \mathcal{F}_{ss} \models \mathcal{R}^*_{\Theta}[G(s_{gr})]$.

To prove that each step preserves logical equivalence, it suffices to show the following:

$$\mathcal{F} \models \forall a, s\ G\big(\text{DO}(a, s)\big) \equiv \mathcal{R}_{\Theta}\big[G\big(\text{DO}(a, s)\big)\big].$$

The proof is by induction on the size of the sentence G.

The proofs of equivalence for the first four steps of the regression operator $\mathcal{R}_{\Theta}$ are relatively simple.

(i) Immediate.
(ii) By the form of the successor state axioms.
(iii) Follows from the definition of negation and the quantifiers, and the inductive hypothesis.
(iv) From the definition of the connectives and the inductive hypothesis.

The proofs of equivalence for steps (v), and (vi) are as follows:

(v) $\mathbf{Knows}\big(W, \text{DO}(a, s)\big)$

by the definition of **Knows**

$$\forall s''\ \text{K}\big(s'', \text{DO}(a, s)\big) \rightarrow W[s'']$$

by the successor state axiom for K (sentence (18)), and the fact that the axiomatization entails $\forall s, s'\ \text{SR}(a, s) = \text{SR}(a, s')$ and also the inductive hypothesis

$$\forall s'\ \big(\text{K}(s', s) \wedge \text{POSS}(a, s')\big) \rightarrow \mathcal{R}_{\Theta}\big[W[\text{DO}(a, s')]\big]$$

by the definition of **Knows**

$$\mathbf{Knows}\big(\text{POSS}(a) \rightarrow \mathcal{R}_{\Theta}\big[W[\text{DO}(a, s')]\big]^{-1}, s\big)$$

(vi) $\mathbf{Knows}\big(W, \text{DO}(\text{SENSE}_i, s)\big)$

by the definition of **Knows**

$$\forall s''\ \text{K}\big(s'', \text{DO}(\text{SENSE}_i, s)\big) \rightarrow W[s'']$$

by the successor state axiom for K (sentence (18)), and also and the inductive hypothesis

$$\forall s'\ \big(\text{K}(s', s) \wedge \text{POSS}(\text{SENSE}_i, s') \wedge \text{SR}(\text{SENSE}_i, s) = \text{SR}(\text{SENSE}_i, s')\big) \rightarrow \mathcal{R}_{\Theta}\big[W[\text{DO}(\text{SENSE}_i, s')]\big]$$

by the definition of equality and the existential quantifier

$$\forall s'\ \big(\text{K}(s', s) \wedge \text{POSS}(\text{SENSE}_i, s') \wedge \exists y\ \text{SR}(\text{SENSE}_i, s) = y \wedge \text{SR}(\text{SENSE}_i, s') = y\big) \rightarrow\ \mathcal{R}_{\Theta}\big[W[\text{DO}(\text{SENSE}_i, s')]\big]$$

by the definition of the connectives and quantifiers

$$\forall y\ \mathrm{SR}(\mathrm{SENSE}_i, s) = y \rightarrow \\ \forall s'\ \big(\mathrm{K}(s', s) \wedge \mathrm{POSS}(\mathrm{SENSE}_i, s') \wedge \mathrm{SR}(\mathrm{SENSE}_i, s') = y\big) \rightarrow \\ \mathcal{R}_{\Theta}\big[W[\mathrm{DO}(\mathrm{SENSE}_i, s')]\big]$$

by the definition of the connectives, quantifiers, and the fact that there can only be one denotation of $\mathrm{SR}(\mathrm{SENSE}_i, s)$

$$\exists y\ \mathrm{SR}(\mathrm{SENSE}_i, s) = y\ \wedge\ \forall s'\ \big(\mathrm{K}(s', s) \rightarrow \\ \mathrm{POSS}(\mathrm{SENSE}_i, s')\ \wedge\ \mathrm{SR}(\mathrm{SENSE}_i, s') = y\big) \rightarrow \\ \mathcal{R}_{\Theta}\big[W[\mathrm{DO}(\mathrm{SENSE}_i, s')]\big]$$

by the definitions of **Knows**

$$\exists y\ \mathrm{SR}(\mathrm{SENSE}_i, s) = y \wedge \\ \mathbf{Knows}\big(\big(\mathrm{POSS}(\mathrm{SENSE}_i) \wedge \mathrm{SR}(\mathrm{SENSE}_i) = y\big) \\ \rightarrow \mathcal{R}_{\Theta}\big[W[\mathrm{DO}(\mathrm{SENSE}_i, s')]\big]^{-1}, s\big) \qquad \square$$

The result means that to test if some sentence G is true after executing a plan, it is only necessary to first regress $G(s_{gr})$, where s_{gr} is the plan expressed as a situation term, using the successor state axioms. This is accomplished by repeatedly passing the regression operator through the formula until the only situation term is S_0. Then the successor state axioms (including (18)) are no longer needed. At that point an ordinary modal logic theorem proving method can be utilized to perform the test to determine whether or not $\mathcal{F} - \mathcal{F}_{ss} \models \mathcal{R}^*_{\Theta}[G(s_{gr})]$.

8.2. Example: Litmus paper test

Consider the following example adapted from [28] (but without the frame axioms). The task is to show that after an agent performs a litmus paper test on an acidic solution, the agent will know that the solution is acidic. The litmus paper turns red if and only if the solution is acidic. The axiomatization includes $\mathrm{ACID}(\mathrm{S}_0)$. The actions are TEST_1 and $\mathrm{SENSE}_\mathrm{R}$. As the action preconditions are all TRUE, the predicate POSS is ignored in the presentation here. The successor state axioms for RED and ACID are given below:

$$\mathrm{RED}\big(\mathrm{DO}(a, s)\big) \equiv \big(\mathrm{ACID}(s) \wedge a = \mathrm{TEST}_1\big) \vee \big(\mathrm{RED}(s) \wedge a \neq \mathrm{TEST}_1\big) \tag{39}$$

$$\mathrm{ACID}\big(\mathrm{DO}(a, s)\big) \equiv \mathrm{ACID}(s) \tag{40}$$

The SR axiom for $\mathrm{SENSE}_\mathrm{R}$ is given below:

$$\mathrm{SR}(\mathrm{SENSE}_\mathrm{R}, s) = r \equiv \big(r = \text{“YES”} \wedge \mathrm{RED}(s)\big) \\ \vee \big(r = \text{“NO”} \wedge \neg\mathrm{RED}(s)\big) \tag{41}$$

The formula to be initially regressed is

$$\mathbf{Knows}\big(\mathrm{ACID}, \mathrm{DO}\big(\mathrm{SENSE}_\mathrm{R}, \mathrm{DO}(\mathrm{TEST}_1, \mathrm{S}_0)\big)\big) \tag{42}$$

Step (vi) of the definition of $\mathcal{R}$ is used with (42) to yield (43).

$$\exists y\ \mathrm{SR}\big(\mathrm{SENSE}_\mathrm{R},\ \mathrm{DO}(\mathrm{TEST}_1, \mathrm{S}_0)\big) = y \wedge \\ \mathbf{Knows}\big(\mathrm{SR}(\mathrm{SENSE}_\mathrm{R}) = y \rightarrow \mathrm{ACID},\ \mathrm{DO}(\mathrm{TEST}_1, \mathrm{S}_0)\big) \tag{43}$$

Next steps (v) and (ii) yield:

$$\exists y\,\big(y = \text{“YES”} \wedge \text{ACID}(\text{S}_0)\big) \vee \big(y = \text{“NO”} \wedge \neg\text{ACID}(\text{S}_0)\big) \wedge \mathbf{Knows}\Big(\big((y = \text{“YES”} \wedge \text{ACID}) \vee (y = \text{“NO”} \wedge \neg\text{ACID})\big) \rightarrow \text{ACID}, \text{S}_0\Big) \tag{44}$$

Note that by step (ii)d $\text{SR}(\text{SENSE}_\text{R}, \text{DO}(\text{TEST}_1, \text{S}_0)) = y$ first expands into

$$\big(y = \text{“YES”} \wedge \text{RED}\big(\text{DO}(\text{TEST}_1, \text{S}_0)\big)\big) \vee \big(y = \text{“NO”} \wedge \neg\text{ACID}\big(\text{DO}(\text{TEST}_1, \text{S}_0)\big)\big)$$

and then is regressed again. The atom $\text{RED}(\text{DO}(\text{TEST}_1, \text{S}_0))$ regresses to $\text{ACID}(\text{S}_0)$ and $\neg\text{RED}(\text{DO}(\text{TEST}_1, \text{S}_0))$ regresses to $\neg\text{ACID}(\text{S}_0)$, by step (ii) with sentence (39). Additionally, $\mathbf{Knows}(\text{SR}(\text{SENSE}_\text{R}) = y \rightarrow \text{ACID}, \text{DO}(\text{TEST}_1, \text{S}_0))$ is regressed by steps (ii), (iii), (iv), and (v) with sentences (39), (40), and (41).

Given that $\text{ACID}(\text{S}_0)$ holds, simplification of (44) then yields (45).

$$\mathbf{Knows}\big((y = \text{“YES”} \wedge \text{ACID}) \rightarrow \text{ACID}, \text{S}_0\big) \tag{45}$$

Sentence (45) is clearly valid and so (42) is entailed by the original theory. Note that (45) can be rewritten as a sentence in an ordinary modal logic because the only situation term is S_0.

Now, consider the safe opening example given earlier. We wish to prove that

$$\text{OPEN}\big(\text{SF}, \text{DO}\big(\text{DIAL-COMB}(\text{SF}), \text{DO}(\text{READ}(\text{PPR}), \text{S}_0)\big)\big) \tag{46}$$

is entailed by the axiomatization given in Section 5. Sentence (46) is regressed to (47) by step (ii) with sentence (20).

$$\text{DIAL-COMB}(\text{SF}) = \text{DIAL-COMB}(\text{SF}) \vee \big(\text{OPEN}\big(\text{SF}, \text{DO}(\text{READ}(\text{PPR}), \text{S}_0)\big) \wedge \text{DIAL-COMB}(\text{SF}) \neq \text{LOCK}(\text{SF})\big) \tag{47}$$

At this point it can be seen that the regressed formula is entailed by the axioms since one of the disjuncts $\text{DIAL-COMB}(\text{SF}) = \text{DIAL-COMB}(\text{SF})$ is clearly entailed by the theory. The crucial part of this particular problem lies in determining that the DIAL-COMB action is possible in $\text{DO}(\text{READ}(\text{PPR}), \text{S}_0)$. This is the topic of the next section.

8.3. Legality testing

In order to determine whether or not a particular sequence of actions results in a state in which a particular sentence is true, it is also necessary to show that each step in the sequence of actions is executable or possible. Following Reiter [35], we define a *legal action sequence*. Consider a sequence of ground action terms $\alpha_1, \ldots, \alpha_n$. This sequence is *legal* if and only if, beginning in the initial situation S_0, each action α_i in the sequence is possible in the state resulting from performing the actions $\alpha_1, \ldots, \alpha_{i-1}$. The situation term

$$\text{DO}\big(\alpha_m, \text{DO}(\alpha_{m-1}, \ldots, \text{DO}(\alpha_1, \text{S}_0)\ldots)\big)$$

is a *legal situation* if and only if $[\alpha_1, \ldots, \alpha_m]$ is a legal action sequence.

We need a method of testing the legality of action sequences. Suppose that there are n actions and the action precondition axioms are as follows:

$$\begin{aligned}(\forall \vec{x_1})\textsc{Poss}\big(\mathrm{A}_1(\vec{x_1}), s\big) &\equiv \Pi_{\mathrm{A}_1}(\vec{x}_1, s),\\ &\vdots\\ (\forall \vec{x_n})\textsc{Poss}\big(\mathrm{A}_n(\vec{x_n}), s\big) &\equiv \Pi_{\mathrm{A}_n}(\vec{x}_n, s)\end{aligned}$$

We are using the notation $\Pi_{\mathrm{A_i}}(\vec{x}_i, s)$ to represent the right-hand sides of the action precondition axioms as introduced in sentence (1). Given a particular ground action term $\alpha_i(\vec{t})$ from the sequence, we need to pick out the proper instance of the corresponding $\Pi_{\mathrm{A_j}}$. The notation $\Pi_{(\alpha_i)}$ is used to indicate the $\Pi_{\mathrm{A_j}}[\vec{x} \mapsto \vec{t}]$ such that $\alpha_i = \mathrm{A}_j(\vec{x})[\vec{x} \mapsto \vec{t}]$. Then the sequence $[\alpha_1, \ldots, \alpha_m]$, where each α_i is a ground action term, is a legal action sequence if and only if

$$\begin{aligned}\mathcal{F} \models \Pi_{(\alpha_1)}[s \mapsto \mathrm{S}_0] \wedge \Pi_{(\alpha_2)}\big[s \mapsto \textsc{do}(\alpha_1, \mathrm{S}_0)\big] \wedge \cdots\\ \wedge \Pi_{(\alpha_m)}\big[s \mapsto \textsc{do}([\alpha_1, \ldots, \alpha_{m-1}], \mathrm{S}_0)\big]\end{aligned} \tag{48}$$

by the definition of a legal action sequence.

By the correctness of regression, we can conclude that the sequence $[\alpha_1, \ldots, \alpha_n]$ is a legal action sequence if and only if

$$\begin{aligned}\mathcal{F} - \mathcal{F}_{ss} \models \Pi_{(\alpha_1)}[s \mapsto \mathrm{S}_0] \wedge \mathcal{R}^*_{\mathcal{O}}\big[\Pi_{(\alpha_2)}[s \mapsto \textsc{do}(\alpha_1, \mathrm{S}_0)]\big] \wedge \cdots\\ \wedge \mathcal{R}^*_{\mathcal{O}}\big[\Pi_{(\alpha_m)}[s \mapsto \textsc{do}([\alpha_1, \ldots, \alpha_{m-1}], \mathrm{S}_0)]\big]\end{aligned}$$

Returning to the safe opening example, consider testing the legality of the following situation:

$$\textsc{do}\big(\textsc{dial-comb}(\textsc{sf}), \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0)\big) \tag{49}$$

This is answered by determining whether or not the axiomatization entails sentence (50), which is formed on the basis of (48), with the preconditions of DIAL-COMB and READ as axiomatized with sentences (19) and (25).

$$\begin{aligned}&\textsc{At}(\textsc{ppr}, \mathrm{S}_0) \wedge\\ &\quad \textsc{Safe}\big(\textsc{sf}, \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0)\big) \wedge \textsc{At}\big(\textsc{sf}, \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0)\big) \wedge\\ &\quad \textbf{Kref}\big(\textsc{comb}(\textsc{sf}), \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0)\big)\end{aligned} \tag{50}$$

Note that $\textsc{Safe}(\textsc{sf}, \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0))$ regresses to $\textsc{Safe}(\textsc{sf}, \mathrm{S}_0)$ by step (ii) of the regression operator and sentence (21). Also, $\textsc{At}(\textsc{sf}, \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0))$ regresses to

$$\big(\textsc{read}(\textsc{ppr}) = \textsc{moveto}(\textsc{sf}) \vee \big(\textsc{At}(\textsc{sf}, \mathrm{S}_0) \wedge \textsc{read}(\textsc{ppr}) \neq \textsc{moveto}(\textsc{sf})\big)\big)$$

by step (ii) and sentence (22). Furthermore, $\textbf{Kref}(\textsc{comb}(\textsc{sf}), \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0))$ is an abbreviation for $\exists x\ \textbf{Knows}(\textsc{comb}(\textsc{sf}) = x, \textsc{do}(\textsc{read}(\textsc{ppr}), \mathrm{S}_0))$, which regresses to

$$\begin{aligned}&\exists x\ \exists y\ \textsc{sr}\big(\textsc{read}(\textsc{ppr}), \mathrm{S}_0\big) = y \wedge\\ &\quad \textbf{Knows}\big((\textsc{At}(\textsc{ppr}) \wedge \textsc{sr}(\textsc{read}(\textsc{ppr})) = y \rightarrow \textsc{comb}(\textsc{sf}) = x, \mathrm{S}_0\big)\end{aligned} \tag{51}$$

by step (vi), and then to

$$\begin{aligned}&\exists x\ \exists y\ y = \textsc{info}(\textsc{ppr}, \mathrm{S}_0) \wedge\\ &\quad \textbf{Knows}\big((\textsc{At}(\textsc{ppr}) \wedge y = \textsc{info}(\textsc{ppr})) \rightarrow \textsc{comb}(\textsc{sf}) = x, \mathrm{S}_0\big)\end{aligned} \tag{52}$$

by step (ii)d with sentence (27). Therefore sentence (50) regresses to the following sentence:

$$\begin{aligned}&\textsc{At}(\textsc{ppr}, s_0) \wedge \textsc{Safe}(\textsc{sf}, s_0) \wedge \\ &\quad \big(\textsc{read}(\textsc{ppr}) = \textsc{moveto}(\textsc{sf}) \vee \big(\textsc{At}(\textsc{sf}, s_0) \wedge \textsc{read}(\textsc{ppr}) \neq \textsc{moveto}(\textsc{sf})\big)\big) \wedge \\ &\quad \exists x\, \exists y\ y = \textsc{info}(\textsc{ppr}, s_0) \wedge \\ &\quad \mathbf{Knows}\big(\big(\textsc{At}(\textsc{ppr}) \wedge y = \textsc{info}(\textsc{ppr})\big) \rightarrow \textsc{comb}(\textsc{sf}) = x, s_0\big) \end{aligned} \tag{53}$$

It can be readily determined that the axiomatization of the initial situation entails this sentence. The axiomatization includes $\textsc{At}(\textsc{ppr}, s_0)$, $\textsc{Safe}(\textsc{sf}, s_0)$, and $\textsc{At}(\textsc{sf}, s_0)$. Since the axiomatization also includes $\mathbf{Knows}(\textsc{info}(\textsc{ppr}) = \textsc{comb}(\textsc{sf}), s_0)$,

$$\begin{aligned}&\exists x\, \exists y\ y = \textsc{info}(\textsc{ppr}, s_0) \wedge \\ &\quad \mathbf{Knows}\big(\big(\textsc{At}(\textsc{ppr}) \wedge y = \textsc{info}(\textsc{ppr})\big) \rightarrow \textsc{comb}(\textsc{sf}) = x, s_0\big) \end{aligned} \tag{54}$$

is entailed as well. Informally, if

$$\exists y\ \mathbf{Knows}\big(\textsc{At}(\textsc{ppr}) \wedge y = \textsc{info}(\textsc{ppr}), s_0\big)$$

holds, then certainly

$$\exists y\ \mathbf{Knows}\big(y = \textsc{info}(\textsc{ppr}), s_0\big)$$

holds as well. Since

$$\mathbf{Knows}\big(\textsc{info}(\textsc{ppr}) = \textsc{comb}(\textsc{sf}), s_0\big)$$

holds,

$$\exists x\ \mathbf{Knows}\big(\textsc{comb}(\textsc{sf}) = x, s_0\big)$$

must hold too.

9. An extended example: The Omelet problem

We illustrate the approach with a problem that initially appeared in [37] and has been recently popularized in a somewhat altered form as a problem for AI planning by David Poole.

> Imagine that we have a robot working as a chef. Its task is to make a 3 egg omelet from a set of eggs some of which may be bad. None of the eggs in the omelet should be bad. The robot has two bowls. It can only see if an egg is bad if it has been broken into a bowl. It can throw out the contents of a bowl and also pour the contents of one bowl into another.
>
> A limited number of eggs may be assumed. Additionally we can add the statement that there are at least three good eggs.

In this section, we first provide a basic axiomatization of the problem. Next, we introduce the constructs of the Golog agent programming language. Finally, a Golog program is given for the Omelet problem.

9.1. The axiomatization

Our agent will have at its disposal both a SMALL_BOWL and a LARGE_BOWL. Furthermore it is assumed that another container BASKET is available and contains some number of eggs.

The agent needs to be able to break an egg into a bowl. After breaking the egg, the agent will no longer be holding the egg, the egg will be broken, and in the bowl. The agent must also have the capability of pouring the contents of one bowl into another. After the pouring action, the eggs will no longer be in the first bowl, but will be in the second. Also available is the action of throwing out the contents of a bowl. The contents will then no longer be in the bowl. The agent needs to be able to fetch an egg from the basket. Finally, we must endow the agent with the capability of inspecting a bowl to see if there are any bad eggs in it. The goal of the agent is to have three eggs in the large bowl that are not bad.

We provide the following terms denoting actions:

1. BREAK_INTO(*bowl*)
2. POUR(*bowl*1, *bowl*2)
3. THROW_OUT(*bowl*)
4. INSPECT(*bowl*)
5. FETCH(*e*, *container*)

The following fluents are needed:

1. IN(*egg*, *bowl*, *s*)
2. BROKEN(*egg*, *s*)
3. HOLDING(*egg*, *s*)
4. NUMBER_EGGS(*bowl*, *s*)
5. BAD(*egg*, *s*)

and the following *Non-Fluents*:

1. EGG(x)

Note that BAD needs to be a fluent even though there are no actions that change whether or not an egg is bad, because we specifically do not want our agent to know whether or not every egg is bad. On the other, EGG(x) is a non-fluent since we both do not have actions that change whether an object into an egg or a non-egg, and we are willing to allow the agent to know whether or not an object is an egg without sensing.

Our robot has only one arm. It is therefore, for example, not capable of pouring the contents of one bowl into another if it is already holding an egg. The robot can only determine if an egg is bad if it is broken and in a container. These restrictions are captured in our axiomatization of POSS for the robot's repertoire of actions as given below:

$$\begin{aligned}&\text{POSS}\big(\text{BREAK\_INTO}(bowl), s\big) \equiv \\ &\quad \exists egg\, \neg\text{BROKEN}(egg, s) \wedge \text{HOLDING}(egg, s)\end{aligned} \tag{55}$$

$$\text{POSS}\big(\text{FETCH}(e, con), s\big) \equiv \text{IN}(e, con, s) \wedge \neg\exists e_1\ \text{HOLDING}(e_1, s) \tag{56}$$

$$\begin{aligned}&\text{POSS}\big(\text{POUR}(b_1, b_2), s\big) \equiv\\ &\quad \neg\exists e\ \text{HOLDING}(e, s) \wedge \text{NUMBER\_EGGS}(b_1, s) \geqslant 0\end{aligned} \tag{57}$$

$$\begin{aligned}&\text{POSS}\big(\text{INSPECT}(b), s\big)\ \equiv\\ &\quad \exists e\ \text{EGG}(e) \wedge \text{IN}(e, b, s) \wedge \text{BROKEN}(e)\end{aligned} \tag{58}$$

The following are the successor-state axioms[17] for the fluents:

$$\begin{aligned}&\text{BROKEN}\big(e, do(a, s)\big) \equiv\\ &\quad \big(\text{HOLDING}(e, s) \wedge \exists b\ a = \text{BREAK\_INTO}(b)\big) \vee \text{BROKEN}(e, s)\end{aligned} \tag{59}$$

$$\begin{aligned}&\text{NUMBER\_EGGS}\big(b, do(a, s)\big) = n \equiv\\ &\quad \big(\text{NUMBER\_EGGS}(b, s) = n - 1 \wedge a = \text{BREAK\_INTO}(b)\big) \vee\\ &\quad \big(\text{NUMBER\_EGGS}(b, s) = i \wedge \exists b_1\ a = \text{POUR}(b_1, b) \wedge\\ &\qquad \text{NUMBER\_EGGS}(b_1, s) = j \wedge n = i + j\big) \vee\\ &\quad \big(n = 0 \wedge \big(a = \text{THROW\_OUT}(b) \vee \exists b_1\ a = \text{POUR}(b, b_1)\big)\big) \vee\\ &\quad \big(\text{NUMBER\_EGGS}(b, s) = n \wedge \neg\big(a = \text{BREAK\_INTO}(b) \vee\\ &\qquad \big(\exists b_1\ a = \text{POUR}(b_1, b) \vee a = \text{POUR}(b, b_1)\big)\\ &\qquad\quad \vee a = \text{THROW\_OUT}(b)\big)\big)\end{aligned} \tag{60}$$

$$\begin{aligned}&\text{IN}\big(e, b_1, do(a, s)\big) \equiv\\ &\quad \big(\text{HOLDING}(e, s) \wedge a = \text{BREAK\_INTO}(b_1)\big) \vee\\ &\quad \big(\exists b_2\ a = \text{POUR}(b_2, b) \wedge \text{IN}(e, b_2, s)\big) \vee\\ &\quad \big(\text{IN}(e, b_1, s) \wedge \neg\big(a = \text{THROW\_OUT}(b_1) \vee\\ &\qquad\quad \exists b_2\ a = \text{POUR}(b_1, b_2)\big)\big)\end{aligned} \tag{61}$$

$$\begin{aligned}&\text{HOLDING}\big(e, do(a, s)\big) \equiv\\ &\quad \exists c\ a = \text{FETCH}(b_1, c) \vee\\ &\quad \text{HOLDING}(e, s) \wedge \neg\exists b\ a = \text{BREAK\_INTO}(b)\end{aligned} \tag{62}$$

$$\text{BAD}\big(e, do(a, s)\big) \equiv \text{BAD}(e, s) \tag{63}$$

The following is the SR axiom for INSPECT.

$$\begin{aligned}&\text{SR}(\text{INSPECT}, s) = r \equiv\\ &\quad \big(r = \text{“BAD”} \wedge \text{BAD}(s)\big)\ \vee\ \big(r = \text{“GOOD”} \wedge \neg\text{BAD}(s)\big)\end{aligned} \tag{64}$$

We also need:

$$\text{SR}(\text{BREAK\_INTO}, s) = r \equiv r = \text{“OK”} \tag{65}$$

$$\text{SR}(\text{POUR}, s) = r \equiv r = \text{“OK”} \tag{66}$$

$$\text{SR}(\text{THROW\_OUT}, s) = r \equiv r = \text{“OK”} \tag{67}$$

$$\text{SR}(\text{FETCH}, s) = r \equiv r = \text{“OK”} \tag{68}$$

[17] We assume that the integers and the various arithmetic operations used here have either been axiomatized or built in as interpreted symbols in the fashion of constraint logic programming.

9.2. Golog

Golog [19] is a high-level agent programming language built on the situation calculus along with the approach to the frame problem due to Reiter. It utilizes a notation for complex actions or programs. The construct **Do** is a macro defined in terms of the more primitive situation calculus constructs. The formula $\mathbf{Do}(\delta, s, s')$ holds if the situation s' is a terminating situation for the complex action δ starting in situation s.

The following constructs are available:

- $\delta_1; \delta_2$—sequences;
- $\delta_1|\delta_2$—nondeterministic choice of actions;
- **if** ϕ **then** δ_1 **else** δ_2—conditionals;
- **while** ϕ **do** δ—while loops;
- $(\Pi x)\delta$—nondeterministic choice of parameters;
- recursive procedures.

The **Do** macro is defined to properly expand these various constructs. For example consider:

$$\mathbf{Do}\big([\mathrm{A};\mathrm{B}], s, s'\big) \stackrel{\text{def}}{=} \exists s^* \ \mathbf{Do}(\mathrm{A}, s, s*) \wedge \mathbf{Do}(\mathrm{B}, s^*, s')$$

Full details are given in [19]. After $\mathbf{Do}(\delta, s, s')$ is executed by the Golog interpreter, s' is bound to a situation term which represents a possible sequence of primitive situation calculus actions which can result from a particular run of δ.

In the process of expanding the program δ, the Golog interpreter utilizes regression to determine the truth of fluents at various points. In the presentation here, we are assuming that the interpreter of [19] has been modified to both incorporate the appropriate mechanisms to regress **Knows** as discussed in this paper, and so that it calls a modal theorem prover to test whether or not the axiomatization of the initial situation entails the regressed formula.

9.3. The omelet program

Consider the candidate Golog encoding of the omelet problem given below:

```
While ¬NUMBER_EGGS(LARGE_BOWL) = 3
    (Πe) FETCH(e, BASKET);
         BREAK_INTO(SMALL_BOWL);
         if BAD(SMALL_BOWL)
           then THROW_OUT(SMALL_BOWL)
           else POUR(SMALL_BOWL, LARGE_BOWL);
```

In line 2 of the program, the agent (or interpreter) picks up an arbitrary egg and then breaks it into the small bowl. The conditional statement (line 4) requires the agent to determine whether the egg in the bowl is bad or not. If the egg in the bowl is bad, the agent must throw out the contents of the bowl (line 5). Otherwise, it must pour the contents into the large bowl

(line 6). How is it possible for the agent to know whether or not any particular egg is bad? Without such information, the conditional statement simply can not be executed. If this information is given initially (somehow the agent knows for each particular egg whether or not it is bad), the problem would be very unrealistic. In addition to having to know whether or not each egg is bad, the agent would have to have some way to identify each particular egg.

The type of program we need is a *knowledge-based program* in the sense of [6,35] since the code must make use of **Knows**. Therefore, we adopt he following Golog omelet program.

While ¬**Knows**(NUMBER_EGGS(LARGE_BOWL) = 3)
  (Πe) FETCH(e, BASKET);
    BREAK_INTO(SMALL_BOWL);
    INSPECT(SMALL_BOWL);
    if Knows(BAD(SMALL_BOWL))
      then THROW_OUT(SMALL_BOWL)
      else POUR(SMALL_BOWL, LARGE_BOWL);

Now, the presence of the inspect action (line 4) guarantees that when the agent executes the condition (line 5), it will know whether or not the egg is bad. The conditional can then be executed. Since it is the knowledge of the agent that is important, all conditions (line 1 and line 5) are within the **Knows** operator. For a condition to be executable, what is important is not whether or not it is true, but rather that the agent either knows that it is true or knows that it is false.

We can simulate runs of such a program by specifying an initial state of the world. For example:

EGG(EGG1) EGG(EGG2) EGG(EGG3) EGG(EGG4)
EGG(EGG5) EGG(EGG6) EGG(EGG7) EGG(EGG8)
¬BROKEN(EGG1, S_0) ¬BROKEN(EGG2, S_0)
¬BROKEN(EGG3, S_0) ¬BROKEN(EGG4, S_0)
¬BROKEN(EGG5, S_0) ¬BROKEN(EGG6, S_0)
¬BROKEN(EGG7, S_0) ¬BROKEN(EGG8, S_0)
¬BAD(EGG1, S_0) BAD(EGG2, S_0) ¬BAD(EGG3, S_0) BAD(EGG4, S_0)
BAD(EGG5, S_0) BAD(EGG6, S_0) ¬BAD(EGG7, S_0) ¬BAD(EGG8, S_0)
¬∃e HOLDING(e, S_0)
IN(EGG1, BASKET, S_0) IN(EGG2, BASKET, S_0) IN(EGG3, BASKET, S_0)
IN(EGG4, BASKET, S_0) IN(EGG5, BASKET, S_0) IN(EGG6, BASKET, S_0)
IN(EGG7, BASKET, S_0) IN(EGG8, BASKET, S_0)
¬∃e IN(e, SMALL_BOWL, S_0) ¬∃e IN(e, LARGE_BOWL, S_0)
NUMBER_EGGS(LARGE_BOWL, S_0) = 0
NUMBER_EGGS(SMALL_BOWL, S_0) = 0
NUMBER_EGGS(BASKET, S_0) = 8

We have specified that eight eggs are available. They are all in the basket and none of them are broken. We have also specified which of the eggs are bad and which are not. There are a total of 4 bad eggs. The agent is not holding anything and both of the bowls are empty.

Note that the result of the execution of the Golog program is a situation term at which it is true that the large bowl contains three good eggs. In other words our axiomatization entails **Knows**(NUMBER_EGGS(LARGE_BOWL) $= 3, s'$) where the s' satisfies **Do**$(\delta, \mathrm{S}_0, s')$. The sequence of steps encoded in the situation term s' represents a possible sequence of actions that will result in the goal. We are in effect simulating a run of the program given a particular initial situation in which there are a certain number of eggs and it is specified which ones are good and which ones are bad. For example, the following sequence of actions is the result of a possible run of the program:

$$\begin{aligned}&[\text{FETCH}(\text{EGG}3, \text{BASKET}), \text{BREAK\_INTO}(\text{SMALL\_BOWL}),\\ &\text{INSPECT}(\text{SMALL\_BOWL}), \text{POUR}(\text{SMALL\_BOWL}, \text{LARGE\_BOWL}),\\ &\text{FETCH}(\text{EGG}2, \text{BASKET}), \text{BREAK\_INTO}(\text{SMALL\_BOWL}),\\ &\text{INSPECT}(\text{SMALL\_BOWL}), \text{THROW\_OUT}(\text{SMALL\_BOWL}),\\ &\text{FETCH}(\text{EGG}8, \text{BASKET}), \text{BREAK\_INTO}(\text{SMALL\_BOWL}),\\ &\text{INSPECT}(\text{SMALL\_BOWL}), \text{POUR}(\text{SMALL\_BOWL}, \text{LARGE\_BOWL}),\\ &\text{FETCH}(\text{EGG}7, \text{BASKET}), \text{BREAK\_INTO}(\text{SMALL\_BOWL}),\\ &\text{INSPECT}(\text{SMALL\_BOWL}), \text{POUR}(\text{SMALL\_BOWL}, \text{LARGE\_BOWL})]\end{aligned}$$

The same Golog program works with other initial states of the world as long as we have at least three good eggs.

Here, we have just taken the simplest view of combining Golog with knowledge and knowledge-producing actions. There are many issues to be dealt with and references to the literature on the topic are given in the next section.

10. Conclusion

10.1. Summary

This paper has proposed a method for handling the frame problem for knowledge-producing actions. Since the work builds upon Reiter's approach to the frame problem, the results can be incorporated into the agent programming languages Golog and ConGolog.

A number of properties of the specification were established. These properties are in effect analogues of the frame problem for changes in knowledge. The properties are that knowledge-producing actions do not affect fluents other than the knowledge fluent, actions that are not knowledge-producing only affect the knowledge fluent as appropriate, and agents know the effects of their actions. In addition, *memory* emerges as a side-effect: if something is known in a certain situation, it remains known at successor situations, unless something relevant has changed.

Also, the issue of automatically reasoning with such an axiomatization was addressed. In particular, a form of *regression* examined by Reiter for reducing reasoning about future situations to reasoning about the initial situation has been extended to now cover

knowledge-producing actions. Additionally, the result of regression can be used with an ordinary modal theorem proving method to address the projection problem.

10.2. Related work

In [39], the approach developed here is extended in a preliminary fashion to cover the case where the knowledge prerequisites and effects of actions can be *indexical* knowledge rather than *objective* knowledge (the case considered here). Following [16,18], this was done by making situations a composite of agents, times and worlds. In [44], the indexical knowledge of time is addressed in the context of allowing concurrent actions. In [25], the framework developed in this paper is utilized as the basis for a formal theory of testing.

In [42], the issue of *belief* rather than knowledge is addressed. The results presented in this paper required that the accessibility relation be reflexive. Note that in the case of a knowledge-producing action a that causes P to be known at $\mathrm{DO}(a, s)$, there must be a situation s' such that $\mathrm{K}(s', s)$, and $\mathrm{P}(s')$. But in the case of a belief-producing action, there is no guarantee that such a situation s' exist. In fact, if the agent falsely believes P and then does an accurate sensing of the truth of P, there will then be no accessible situations. What is needed is a form of belief revision. This is why the results do not directly extend to modal logics without a reflexive accessibility relation. The work reported in [42], incorporates into a framework similar to that developed here, the machinery to handle belief revision. But the issue of the analogue of the frame problem in the context of belief, i.e., ensuring that belief only changes as appropriate, remains an open question. Another important topic for further work is extending regression to the case of belief.

A probabilistic notion of belief designed to handle noisy signals from multiple sensors is developed in [1]. The topic of *only-knowing* is considered within the situation calculus in [14]. An account of the ability of an agent to execute a Golog program is given in [17]. In all of this work, the automation of reasoning, possibly through adapting regression, is an important area for future research.

There has been some work on incorporating knowledge and knowledge-producing actions into other action logics, in particular the language $\mathcal{A}$ [2,22]. Here the approach to the frame problem underlying the language $\mathcal{A}$ (essentially equivalent to a propositional version of the successor state axioms used in this paper) is extended to handle changes in knowledge given the presence of sensing. Implementation is handled in [22] by translating the axiomatization into epistemic logic programs, while in [2] a form of regression is considered. Additionally, knowledge and knowledge-producing actions have been added to the fluent calculus [43].

A number of authors have proposed alternatives to the notion of possible worlds for the representation of knowledge within the situation calculus. Petrick and Levesque [30] consider *knowledge fluents* and Funge [8] utilizes *interval valued epistemic fluents*. In a related effort, Reiter [36] investigates the circumstances under which knowledge can be reduced to provability.

There has also been work on the integration of Golog/Congolog with sensing and knowledge [9,10,15,35], with the goal of controlling a robot or software agent. An important issue is how to combine the *off-line* reasoning about the effects of actions with

the necessity that sensing be *on-line* since the result of sensing can only be determined by executing the action.

Acknowledgements

Our work on the situation calculus and the frame problem has been carried out in collaboration with Yves Lespérance, Fangzhen Lin, and Ray Reiter. We thank them for many useful discussions, and for comments on earlier versions of this paper. Additionally, we thank Leo Bertossi, Joe Halpern, Neelakantan Kartha, Sheila McIlraith, Bill Millar, Steven Shapiro, Stephen Zimmerbaum, and the anonymous reviewers for helpful comments on earlier versions of this paper. This research was funded in part by the National Sciences and Engineering Research Council of Canada (NSERC), and the Institute for Robotics and Intelligent Systems. The first author was an NSERC International Postdoctoral Fellow from 1992 through 1994 during which a large portion of this work was completed. The first author also acknowledges support from the New Jersey Institute of Technology under SBR grant 421250, the National Science Foundation (NSF) under grants SES-9819116 and CISE-9818309, and also from the New Jersey Commission on Science and Technology.

References

[1] F. Bacchus, J. Halpern, H.J. Levesque, Reasoning about noisy sensors and effectors in the situation calculus, Artificial Intelligence 111 (1999) 171–208.
[2] C. Baral, T. Son, Formalizing sensing actions—A transition function based approach, Artificial Intelligence 125 (2001) 19–91.
[3] R. Bull, K. Segerberg, Basic modal logic, in: D. Gabbay, F. Guenther (Eds.), Handbook of Philosophical Logic, Vol. II, Chapter 1, D. Reidel, Dordrecht, 1984, pp. 1–88.
[4] B.F. Chellas, Modal Logic: An Introduction, Cambridge University Press, Cambridge, 1980.
[5] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, M. Williamson, An approach to planning with incomplete information, in: B. Nebel, C. Rich, W. Swartout (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference, Cambridge, MA, 1992, pp. 115–125.
[6] R. Fagin, J.Y. Halpern, Y.O. Moses, M.Y. Vardi, Reasoning about Knowledge, MIT Press, Cambridge, MA, 1995.
[7] A. Frisch, R. Scherl, A general framework for modal deduction, in: J.A. Allen, R. Fikes, E. Sandewall (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, Morgan Kaufmann, San Mateo, CA, 1991, pp. 196–207.
[8] J. Funge, Representing knowledge within the situation calculus using interval-valued epistemic fluents, J. Reliable Comput. 5 (1) (1999).
[9] G. De Giacomo, H.J. Levesque, Projecting using regression and sensors, in: Proc. IJCAI-99, Stockholm, Sweden, 1999, pp. 160–165.
[10] G. De Giacomo, H.J. Levesque, An incremental interpreter for high-level programs with sensing, in: Logical Foundations for Cognitive Agents: Contributions in honor of Ray Reiter, Springer, Berlin, 1999, pp. 86–102.
[11] A.R. Haas, The case for domain-specific frame axioms, in: F.M. Brown (Ed.), The Frame Problem in Artificial Intelligence. Proceedings of the 1987 Workshop, Morgan Kaufmann, San Mateo, CA, 1987, pp. 343–348.
[12] G.E. Hughes, M.J. Cresswell, An Introduction to Modal Logic, Methuen, London, 1968.
[13] S. Kripke, Semantical considerations on modal logic, Acta Philos. Fenn. 16 (1963) 83–94.

[14] G. Lakemeyer, H.J. Levesque, AOL: A logic of acting, sensing, knowing and only-knowing, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR-98), Morgan Kaufmann, San Mateo, CA, 1998, pp. 316–327.
[15] G. Lakemeyer, On sensing and off-line interpreting in GOLOG, in: Logical Foundations for Cognitive Agents: Contributions in honor of Ray Reiter, Springer, Berlin, 1999, pp. 173–189.
[16] Y. Lespérance, H.J. Levesque, Indexical knowledge in robot plans, in: Proc. AAAI-90, Boston, MA, 1990, pp. 1030–1037.
[17] Y. Lespérance, H.J. Levesque, F. Lin, R.B. Scherl, Ability and knowing how in the situation calculus, Studia Logica 66 (1) (2000) 165–186.
[18] Y. Lespérance, A formal theory of indexical knowledge and action, PhD Thesis, University of Toronto, January 1991.
[19] H. Levesque, R. Reiter, Y. Lespérance, F. Lin, R.B. Scherl, GOLOG: A logic programming language for dynamic domains, J. Logic Programming 31 (1997) 59–83.
[20] H. Levesque, What is planning in the presence of sensing?, in: Proc. AAAI-96, Portland, OR, 1996, pp. 1139–1146.
[21] F. Lin, R. Reiter, State constraints revisited, J. Logic Comput. 4 (5) (1994) 655–678.
[22] J. Lobo, G. Mendez, S. Taylor, Knowledge and the action description language $\mathcal{A}$, J. Logic Programming 1 (2) (2001) 129–184.
[23] J. McCarthy, P. Hayes, Some philosophical problems from the standpoint of artificial intelligence, in: B. Meltzer, D. Michie (Eds.), Machine Intelligence 4, Edinburgh University Press, Edinburgh, UK, 1969, pp. 463–502.
[24] J. McCarthy, Programs with common sense, in: M. Minsky (Ed.), Semantic Information Processing, MIT Press, Cambridge, MA, 1968, pp. 403–418 [Chapter 7].
[25] S. McIlraith, R.B. Scherl, What sensing tells us: Towards a formal theory of testing for dynamical systems, in: Proc. AAAI-00, Austin, TX, 2000, pp. 483–490.
[26] S. McIlraith, Integrating actions and state constraints: A closed-form solution to the ramification problem (sometimes), Artificial Intelligence 116 (2000) 87–121.
[27] R.C. Moore, Reasoning about knowledge and action, Technical Note 191, SRI International, Menlo Park, CA, October 1980.
[28] R.C. Moore, A formal theory of knowledge and action, in: J.R. Hobbs, R.C. Moore (Eds.), Formal Theories of the Commonsense World, Ablex, Norwood, NJ, 1985, pp. 319–358.
[29] E.P.D. Pednault, ADL: Exploring the middle ground between STRIPS and the situation calculus, in: R.J. Brachman, H. Levesque, R. Reiter (Eds.), Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, San Mateo, CA, 1989, pp. 324–332.
[30] R. Petrick, H. Levesque, Knowledge equivalence in combined action theories, in: D. Fensel, F. Giunchiglia, D. McGuinness, M. Williams (Eds.), Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR2002), Morgan Kaufmann, San Mateo, CA, 2002, pp. 303–314.
[31] F. Pirri, R. Reiter, Some contributions to the metatheory of the situation calculus, J. ACM 46 (3) (1999) 261–325.
[32] R. Reiter, 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: Papers in Honor of John McCarthy, Academic Press, San Diego, CA, 1991, pp. 359–380.
[33] R. Reiter, The projection problem in the situation calculus: A soundness and completeness result, with an application to database updates, in: Proceedings of the First International Conference on AI Planning Systems, College Park, MD, 1992, pp. 198–203.
[34] R. Reiter, Proving properties of states in the situation calculus, Artificial Intelligence 64 (1993) 337–351.
[35] R. Reiter, Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems, MIT Press, Cambridge, MA, 2001.
[36] R. Reiter, On knowledge-based programming with sensing in the situation calculus, ACM Trans. Comput. Logic (TOCL) 2 (4) (2001) 433–457.
[37] L.J. Savage, The Foundations of Statistics, Wiley, New York, 1954.
[38] R.B. Scherl, H.J. Levesque, The frame problem and knowledge producing actions, in: Proc. AAAI-93, Washington, DC, 1993, pp. 689–695.

[39] R. Scherl, H. Levesque, Y. Lespérance, The situation calculus with sensing and indexical knowledge, in: Proceedings of BISFAI'95: The Fourth Bar-Ilan Symposium on Foundations of Artificial Intelligence, Ramat Gan and Jerusalem, Israel, 1995, pp. 86–95.
[40] R. Scherl, A constraint logic approach to automated modal deduction, PhD Thesis, University of Illinois, 1992.
[41] L.K. Schubert, Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions, in: H.E. Kyberg, R.P. Loui, G.N. Carlson (Eds.), Knowledge Representation and Defeasible Reasoning, Kluwer Academic, Boston, MA, 1990, pp. 23–67.
[42] S. Shapiro, M. Pagnucco, Y. Lespérance, H.J. Levesque, Iterated belief change in the situation calculus, in: Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR-2000), Morgan Kaufmann, San Mateo, CA, 2000, pp. 527–538.
[43] M. Thielscher, Representing the knowledge of a robot, in: A. Cohn, F. Giunchiglia, B. Selman (Eds.), Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), Morgan Kaufmann, San Mateo, CA, 2000, pp. 109–120.
[44] S. Zimmerbaum, R. Scherl, Knowledge, time, and concurrency in the situation calculus, in: C. Castelfranchi, Y. Lespérance (Eds.), Intelligent Agents VII: Proceedings of the 2000 Workshop on Agent Theories, Architectures, and Languages (ATAL-2000), in: Lecture Notes in Artificial Intelligence, Vol. 1986, Springer, Berlin, 2001, pp. 31–45.