The Situation Calculus with Sensing and Indexical Knowledge*

Richard B. Scherl†
Department of Computer and Information Science
New Jersey Institute of Technology
Newark, New Jersey 07102
email: scherl@vienna.njit.edu

Hector J. Levesque‡
Department of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 1A4
email: hector@cs.toronto.edu

Yves Lespérance
Department of Computer Science
University of Toronto
Toronto, Ontario
Canada M5S 1A4
email: lesperan@cs.toronto.edu

1 Introduction

Objective knowledge alone is unsuitable for the formalization of action.

- It is not necessary for action. Suppose I don't know where an object is in absolute terms, or where I am, but I do know that the object is 2 feet ahead.
- It is not sufficient for action. Suppose I know the precise location of an object (e.g. at (2,-3)), but I do not know where I am.
- It not produced by perceptual actions. Perceptual actions cannot tell you the location of an object; at best they tell you its location relative to you (2 feet ahead, or within a radius of 6 feet, etc.).

*This research received financial support from the Information Technology Research Center (Ontario, Canada), the Institute for Robotics and Intelligent Systems (Canada), and the Natural Science and Engineering Research Council (Canada)

†Most of the work was performed while a National Sciences and Engineering Research Council of Canada International Postdoctoral Fellow

‡Fellow of the Canadian Institute for Advanced Research

What matters in these examples is not the *objective* locations of objects, but their locations *relative* to the robot. We need to be able to express *indexical* information, i.e. information about the world relative to a context—without having to identify these objectively. Examples are me, my location, my orientation, the current time, and the object in my sights. The importance of such indexical representations in the area of robotics has been pointed out in [Agre and Chapman, 1990].

This paper develops a version of the situation calculus that incorporates the notion of indexical knowledge. It is part of a larger endeavor, the *cognitive robotics* project, which uses the situation calculus as the basis of both a higher level programming language for agents, and a logical theory of the integration of reasoning, action, and perception [Lespérance *et al.*, 1994; Lespérance *et al.*, 1995].

With the representational framework developed here, the knowledge prerequisites and effects of actions can be specified in terms of indexical rather than objective knowledge. The problem includes how to represent information such as the knowledge of relative positions and relative effects of actions, how to model the lack of knowledge of one's own identity, and how to model knowledge of time.

These issues are considered in a modal framework in [Lespérance and Levesque, 1995]. Here, these notions are incorporated into the situation calculus. This is done in such a way as to utilize the solution to the frame problem for knowledge producing actions developed in [Scherl and Levesque, 1993] where only objective knowledge and a single agent are considered. Additionally, a computationally attractive method is developed for answering queries of whether or not a particular sentence is true in the state resulting from the execution of a particular sequence of actions.

2 The Situation Calculus and the Frame Problem

The situation calculus 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 $do(\alpha, s)$. 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, $\textsc{Broken}(x, s)$ 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 $\textsc{pos}(\textsc{bill}, s)$, which denotes the position of BILL in situation s.

One of the longstanding problems connected with the situation calculus has been the frame problem, i.e., the need to add *frame axioms* that specify when fluents remain unchanged. Reiter [1991] (generalizing the work of Haas, Schubert, and Pednault) proposed a solution to the frame problem. In this presentation, the formulation found in [Reiter, 1991] is generalized to a multi-agent context.

A simple example domain will be used throughout this paper. There are some number of

robots who move about on a two-dimensional grid. A relational fluent (HOLDING) is needed to indicate whether or not a particular robot is holding an object and functional fluents are needed to denote the position (POS) of robots (and other objects) and the orientation (ORI) of the robots. The idea here is to imagine that these robots can perform a limited number of primitive actions that are inherently indexical. These (following [Lespérance and Levesque, 1995]) are picking up an object, putting down an object, moving forward one step, turning 90° to the left, turning 90° to the right, and sensing whether an object with certain properties is at his location.

To treat multiple agents, we will take the first argument of *do* to be a pair consisting of an agent and an action type. The variable a below ranges over such pairs. The function AGENT(a) is used to map a to the agent component and the function TYPE(a) is used to map a to the action type component. Action types are represented as strings of characters. For our example, the action types are "pickup", "putdown", "forward", "left", and "right". This differs from our earlier work [Scherl and Levesque, 1993] and the work of Reiter [1991], in which the variable a ranged over terms denoting actions.

The core of the solution to the frame problem rests on providing for each fluent a successor state axiom of the form given below[1].

Successor State Axiom

$$F(do(a,s)) \equiv \gamma_F^+(a,s) \vee (F(s) \wedge \neg\gamma_F^-(a,s)) \tag{1}$$

Here $\gamma_F^+(a,s)$ represents the conditions under which the truth value of F will change from negative to positive and $\gamma_F^-(a,s)$ represents the conditions under which the truth value of F will change from positive to negative. Similar successor state axioms may be written for functional fluents. Reiter[1991] shows how to derive a set of *successor state axioms* of the form given in 1 from the usual axiomatization in terms of positive effect and negative effect axioms, along with unique name axioms and a completeness assumption.

The successor state axioms for the fluents HOLDING, POS, and ORI are given below.

$$\begin{aligned}
&\text{HOLDING}(agt, x, do(a,s)) \equiv \\
&\quad (\text{OBJECT}(x,s) \wedge \text{POS}(agt,s) = \text{POS}(x,s) \wedge \neg\exists y \text{HOLDING}(agt,y,s) \wedge \\
&\qquad \text{TYPE}(a) = \text{"pickup"} \wedge \text{AGENT}(a) = agt) \vee \\
&\quad (\text{HOLDING}(agt,x,s) \wedge \\
&\qquad (\text{TYPE}(a) \neq \text{"putdown"} \vee \text{AGENT}(a) \neq agt))
\end{aligned} \tag{2}$$

This axiom states that the only way for the fluent HOLDING to be true of actor agt and object x in situation $do(a,s)$, is if in situation s either agt is located in the same position as the object x, is not holding anything, and the action a is the execution of a "pickup" type

[1] Unlike the presentation in [Reiter, 1991; Scherl and Levesque, 1993], here we do not use action precondition axioms. The use of such axioms and the *Poss* predicate is perfectly consistent with the approach here, but unnecessary for the types of examples that we are considering.

This discussion assumes (following [Reiter, 1991]) that there are no ramifications, i.e., indirect effects of actions. The assumption that there are no state constraints in the axiomatization of the domain will be made throughout this paper. In [Lin and Reiter, 1994], 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.

action by agt; or agt is holding x in situation s and either the agent of action a is an actor other than agt, or the action type of a is a type other than "putdown."

$$\begin{aligned}
&\textsc{pos}(x, do(a,s)) = l \equiv \\
&\quad (x = \textsc{agent}(a) \vee (\textsc{Object}(x) \wedge \textsc{Holding}(\textsc{agent}(a), x, s))) \\
&\qquad \wedge \textsc{type}(a) = \text{"forward"} \wedge \\
&\qquad\quad \textsc{pos}(\textsc{agent}(a,s)) = (l - (\langle 1, 0\rangle \times \textsc{rot}(\textsc{ori}(\textsc{agent}(a), s))))) \vee \\
&\quad (\textsc{pos}(x,s) = l \wedge (\textsc{type}(a) \neq \text{"forward"} \vee \\
&\qquad\quad (\textsc{agent}(a) \neq x \wedge \neg \textsc{Holding}(\textsc{agent}(a), x, s)))
\end{aligned} \tag{3}$$

The following is the definition of ROT:

$$\textsc{rot}(\theta) \stackrel{\text{def}}{=} \left\langle \begin{array}{cc} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{array} \right\rangle$$

This axiom states that for POS of an object x in situation $do(a,s)$ to be equal to l, it must be the case in situation s that either x is a robot or something being held by a robot, x was located 1 unit behind l and the action a is an execution of action type "forward" by x or the robot holding x; or x was located at l and the action a is not the execution of the action type "forward" by x or some robot holding x.

$$\begin{aligned}
&\textsc{ori}(x, do(a,s)) = \theta \equiv \\
&\quad (x = \textsc{agent}(a) \wedge \textsc{type}(a) = \text{ "left"} \wedge \textsc{ori}(x,s) = \textsc{mod}_{2\pi}(\theta - \pi/2)) \vee \\
&\quad (x = \textsc{agent}(a) \wedge \textsc{type}(a) = \text{"right"} \wedge \textsc{ori}(x,s) = \textsc{mod}_{2\pi}(\theta + \pi/2)) \vee \\
&\quad (\textsc{ori}(x,s) = \theta \wedge (\textsc{type}(a) \neq \text{"left"} \wedge \textsc{type}(a) \neq \text{"right"}) \vee \\
&\qquad\quad \textsc{agent}(a) \neq x)
\end{aligned} \tag{4}$$

This axiom states that for the orientation of robot x to be θ in situation $do(a,s)$ it must have been the case in situation s that either the orientation was $\theta + \pi/2$ and a is the execution of action type "left" by x; or the orientation was $\theta - \pi/2$ and a is the execution of action type "right" by x; or the orientation was θ and a is not the execution of "left" or "right" by x.

We have developed a set of macros for specifying complex actions that are abbreviations for formulas in the situation calculus. Space precludes their presentation here, but it is fully developed in the longer version of this paper. Moving to a particular objective location or relative location can then be specified as a complex action in this macro language.

3 Indexical Knowledge

The approach we take to formalizing indexical knowledge is to adapt the possible-world model of indexical knowledge of [Lespérance and Levesque, 1995] to the situation calculus, much as Moore[1985] adapted the possible-world model of knowledge to the situation calculus.

It will be assumed here that agents are aware of all actions, i.e., every action occurrence is common knowledge. But, consistent with the indexical logic of knowledge and action, although actors know that the agent of a particular action is denoted by a particular term, they will not necessarily know who that agent is. An actor may know that he himself

performed an action, without necessarily knowing who he is. Similarly, an actor may know that the block was moved by the person on his left, without necessarily knowing who the person on his left is.

A 4-place relation $K(agt', s', agt, s)$ is introduced. This is understood as indicating that it is compatible with agt's knowledge in s that the current situation is s' and he is agt'. The predicate K is treated the same way one would any other fluent.

We use the notation $\mathbf{Knows}(A, P(\mathsf{now}), s)$ (read as A knows P in situation s) as an abbreviation for a formula that uses K. For example:

$$\mathbf{Knows}(\text{ROB}, \text{BROKEN}(y, \mathsf{now}), s) \stackrel{\text{def}}{=} \forall agt', s'\ K(agt', s', \text{ROB}, s) \rightarrow \text{BROKEN}(y, s').$$

Note that this notation substitutes the appropriate situation argument for now on expansion. This notation can be generalized inductively to arbitrary formulas so that, for example[2]:

$$\begin{aligned}&\exists x\, \mathbf{Knows}(\text{ROB}, \exists y[\text{NEXTTO}(x, y, \mathsf{now}) \wedge \neg \mathbf{Knows}(\text{ROBERTA}, \text{BROKEN}(y, \mathsf{now}), \mathsf{now})], s)\\ &\stackrel{\text{def}}{=} \exists x \forall agt', s'\ K(agt', s', \text{ROB}, s) \rightarrow \exists y[\text{NEXTO}(x, y, s') \wedge\\ &\qquad \neg \forall\, agt'', s'' K(agt'', s'', \text{ROBERTA}, s') \rightarrow \text{BROKEN}(y, s'')].\end{aligned}$$

We also introduce the notation self as illustrated in the following example:

$$\begin{aligned}&\mathbf{Knows}(\text{ROB}, \text{NEXTTO}(\mathsf{self}, y, \mathsf{now}), s) \stackrel{\text{def}}{=}\\ &\quad \forall\, agt', s'\ K(agt', s', \text{ROB}, s) \rightarrow \text{NEXTTO}(agt', y, s').\end{aligned}$$

The convention is that self is replaced by the leftmost argument of K, when it (self) occurs within a literal not in the scope of another K. Consider the following example:

$$\begin{aligned}&\mathbf{Knows}(\text{ROB}, \exists y[\text{NEXTTO}(\mathsf{self}, y, \mathsf{now}) \wedge \neg \mathbf{Knows}(\mathsf{self}, \text{NEXTTO}(\mathsf{self}, y, \mathsf{now}), \mathsf{now})], s)\\ &\stackrel{\text{def}}{=} \forall agt', s'\ K(agt', s', \text{ROB}, s) \rightarrow \exists y[\text{NEXTTO}(agt', y, s') \wedge\\ &\qquad \neg \forall agt'', s'' K(agt'', s'', agt', s') \rightarrow \text{NEXTTO}(agt'', y, s'')].\end{aligned}$$

The term self can only be used within the scope of a **Knows** operator.

Turning now to the specification of the effects of knowledge-producing actions, consider actions whose effect is to make known the truth value of some formula[3]. We might have a "sense$_\alpha$" action for a sentence α, such that after doing a "sense$_\alpha$", the agent comes to know whether α holds. We introduce the notation $\mathbf{Kwhether}(agt, \alpha, s)$ as an abbreviation for a formula indicating that the truth value of sentence α is known:

$$\mathbf{Kwhether}(agt, \alpha, s) \stackrel{\text{def}}{=} \mathbf{Knows}(agt, \alpha, s) \vee \mathbf{Knows}(agt, \neg\alpha, s).$$

By adding a fluent $\text{TIME}(s)$ representing the absolute time of a situation, we can also handle objective knowledge about time, indexical knowledge about time, and knowledge of the relation between indexical and objective notions of time (e.g. knowing that it is now 3:00). Full details are in the extended version of this paper.

[2]The predicate NEXTTO can be defined in terms of POS.

[3]The results reported here can easily be extended, following [Scherl and Levesque, 1993], to incorporate a "read" type action that makes known the denotation of a term.

4 Solving the Frame Problem for Knowledge

The extension of Reiter's [1991] solution to the frame problem to the situation calculus with indexical knowledge rests on the specification of a successor state axiom (following [Scherl and Levesque, 1993]) for the four-place K relation. 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 first be introduced by stages. Consider the simple situation where there are only two actors and each refers to the other with the term $\text{OTHER}(\mathsf{self}, s)$[4]. It is assumed that OTHER is axiomatized in such a way that $\text{OTHER}(agt, s) \neq agt$. First we will consider what effect ordinary actions will have on the K relation and then we will give a specification for the case where there is only a single knowledge-producing action. In the discussion to follow, for convenience the notation $\langle agt, type \rangle$ is used to represent pairs of agents and action types.

For non-knowledge-producing action types (e.g. "pickup"), the specification (based on Moore [1985]) is as follows:

$$\begin{aligned} & K(agt'', s'', agt', do(\langle agt, \text{"pickup"} \rangle, s)) \equiv \\ & \quad \exists s' \, K(agt'', s', agt', s) \wedge s'' = do(a^*, s') \wedge \exists a^* \, \text{TYPE}(a^*) = \text{"pickup"} \wedge \\ & \quad\quad (agt' = agt \rightarrow \text{AGENT}(a^*) = agt'') \\ & \quad\quad\quad\quad \wedge \\ & \quad\quad (agt' = \text{OTHER}(agt, s) \rightarrow \text{AGENT}(a^*) = \text{OTHER}(agt'', s')) \end{aligned} \tag{5}$$

The axiom ensures that if $agt' = agt$, the only change in knowledge that occurs in moving from s to $do(\langle agt, \text{"pickup"} \rangle, s)$ is the knowledge that the action type "pickup" has been performed by itself. But if agt' is not the agent of the action, then the only change in knowledge that occurs in moving from s to $do(\langle agt, \text{"pickup"} \rangle, s)$ is the knowledge that the action type "pickup" has been performed by the other robot. Note that this does not require the agent to know who he is or who the other agent is.

Now consider the simple case of a knowledge-producing action type "sense$_P$" that determines whether or not the fluent P is true (following Moore [1985]).

$$\begin{aligned} & K(agt'', s'', agt'', do(\langle agt, \text{ "sense}_P\text{"}, s \rangle)) \equiv \\ & \quad \exists s' \, K(agt'', s', agt', s) \wedge s'' = do(a^*, s') \wedge \exists a^* \, \text{TYPE}(a^*) = \text{"sense}_P\text{"} \wedge \\ & \quad\quad (agt' = agt \rightarrow (\text{AGENT}(a^*) = agt'' \wedge P(s) \equiv P(s'))) \\ & \quad\quad\quad\quad \wedge \\ & \quad\quad (agt' = \text{OTHER}(agt, s) \rightarrow \text{AGENT}(a^*) = \text{OTHER}(agt'', s)) \end{aligned} \tag{6}$$

The idea here is that in the case where agt' is the agent of the action, after performing the action agt' not only knows that the action type "sense$_P$" has been performed, and that he has performed it (as above), but also the truth value of the predicate P. On the other hand, iff $agt' \neq agt$ then in moving from s to $do(\langle agt, \text{"sense}_P\text{"} \rangle, s)$, agt' knows only that the action type "sense$_P$" has been performed, and OTHER has performed it, but he does not know the truth value of the predicate P.

[4]The simplest option is chosen here. Issues with regard to naming agents are discussed in [Lespérance and Levesque, 1995].

$$
\begin{array}{l}
Poss(a,s) \rightarrow [K(agt'',s'',agt',do(a,s)) \equiv \\
\quad \exists s'\ K(agt'',s',agt',s) \wedge s'' = do(a^*,s') \wedge \exists a^*\ \textsc{type}(a^*) = \textsc{type}(a) \wedge \\
\quad\quad (agt' = \textsc{agent}(a) \rightarrow (\textsc{agent}(a^*) = agt'' \wedge \\
\quad\quad\quad (\textsc{type}(a) = \mathcal{A}_1 \rightarrow \varphi_1) \\
\quad\quad\quad\quad\quad\quad \wedge \\
\quad\quad\quad\quad\quad\quad \vdots \\
\quad\quad\quad\quad\quad\quad \wedge \\
\quad\quad\quad (\textsc{type}(a) = \mathcal{A}_n \rightarrow \varphi_n)) \wedge \\
\quad\quad (agt' = \textsc{other}_1(\textsc{agent}(a),s) \rightarrow \textsc{agent}(a^*) = \textsc{other}_1(agt'',s')) \\
\quad\quad\quad\quad\quad\quad \wedge \\
\quad\quad\quad\quad\quad\quad \vdots \\
\quad\quad\quad\quad\quad\quad \wedge \\
\quad\quad (agt' = \textsc{other}_{\mathrm{M}}(\textsc{agent}(a),s) \rightarrow \textsc{agent}(a^*) = \textsc{other}_{\mathrm{M}}(agt'',s'))
\end{array}
$$

Figure 1: Successor State Axiom for K

Observe that the successor state axiom for P guarantees that P is true at $do(\langle agt, \text{“sense}_P\text{”}\rangle, s)$ iff P is true at s, and similarly for s' and $do(\langle agt, \text{“sense}_P\text{”}\rangle, s')$. Therefore, P has the same truth value in all worlds s'' such that $K(agt'', s'', agt', do(\langle agt, \text{“sense}_P\text{”}\rangle, s))$, and so $\mathbf{Kwhether}(P, do(\langle agt, \text{“sense}_P\text{”}\rangle, s))$ is true.

In general, there may be many knowledge-producing actions. Associated with each knowledge-producing action $\mathcal{A}_i$ is a formula of the form $\psi_i(s) \equiv \psi_i(s')$, where ψ_i is a sentence. Assume that there are n knowledge-producing actions $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and therefore n associated sentences $\psi_1, \ldots, \psi_n$. Additionally, other than the agent of the action a, there are n other actors–$\textsc{other}_1(agt, s) \ldots \textsc{other}_n(agt, s)$. The general form of the successor state axiom for K is given in Figure 1.

The only remaining issue concerns requiring that the **Knows** operator conform to the properties of a particular modal logic. Restrictions need to be placed on the K relation so that it correctly models the accessibility relation of a particular modal logic. This issue is discussed in [Scherl and Levesque, 1993]. The solution is to utilize a sort *Init* to restrict variables to range only over S_0 and those situations accessible from S_0. Then the K relation is appropriately restricted over these initial situations. The needed restrictions (following [Lespérance and Levesque, 1995]) are reflexivity and transitivity. Full details on the axiomatization of these restrictions is found in the extended version of this paper.

5 Properties of the Solution

Note that even though the logic is a variant of $S4$, Sentence 7 is not a theorem of the logic.

$$\mathbf{Knows}(\textsc{rob}, \mathrm{P}(\mathsf{now})) \rightarrow \mathbf{Knows}(\textsc{rob}, \mathbf{Knows}(\textsc{rob}, \mathrm{P}(\mathsf{now}), \mathsf{now}), S_0) \tag{7}$$

On the other hand 8 is a theorem.

$$\mathbf{Knows}(\textsc{rob}, \mathrm{P}(\mathsf{now})) \rightarrow \mathbf{Knows}(\textsc{rob}, \mathbf{Knows}(\mathsf{self}, \mathrm{P}(\mathsf{now}), \mathsf{now}), S_0) \tag{8}$$

It is not the case that if ROB knows P he then knows that ROB knows P, but rather he knows that *he* knows P.

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.

Theorem 1 (Default Persistence of Ignorance) *If* $\neg$**Knows**(AGT, P(now), s) *then* $\neg$**Knows**(AGT, P(now), $do(a, s)$), *as long as*

1. *P is a fluent whose successor state axiom specifies that it is not changed by the action* TYPE(a). *More formally,* $\forall s,\ \alpha$ TYPE(a) = TYPE(α) $\rightarrow$ $P(s) \equiv P(do(\alpha, s))$

2. *It is not the case that both* AGT = AGENT(a) *and* TYPE(a) *is a knowledge-producing action with the corresponding* φ *being of the form* $F(s) \leftrightarrow F(s')$ *such that either* $F(s)$ *and* **Knows**(AGT, F(now) $\rightarrow$ P(now), s) *or* $\neg F(s)$ *and* **Knows**(AGT, $\neg F$(now) $\rightarrow$ P(now), s)

Finally, it is a property of this specification that agents never forget.

Theorem 2 (Memory) *For all fluents* P *and situations* s, *if* **Knows**(AGT, P(now), s) *then* **Knows**(AGT, P(now), $do(a, s)$) *as long as* P *is a fluent whose successor state axiom specifies that it is not changed by the action* TYPE(a).

Now consider the following sentences:

$$\textbf{Knows}(\text{ROB}, [\exists x \text{OBJ}(x) \wedge \text{POS}(x, \mathsf{now}) = \text{POS}(\mathsf{self}, \mathsf{now}) \wedge \neg\exists y \text{HOLDING}(y, \mathsf{now})], S_0) \tag{9}$$

$$\exists p\, p = \text{POS}(\mathsf{self}, S_0) \wedge \textbf{Knows}(\text{ROB}, [\exists x \text{OBJ}(x) \wedge \text{POS}(x, \mathsf{now}) = p \wedge \neg\exists y \text{HOLDING}(y, \mathsf{now})], S_0) \tag{10}$$

$$\textbf{Knows}(\text{ROB}, \exists y \text{HOLDING}(y, \mathsf{now}), do(\langle \text{ROB}, \text{“pickup”} \rangle, S_0)) \tag{11}$$

Our axiomatization entails that:
$\models$ (9) $\rightarrow$ (11) If ROB knows that there is an object where he is, then after doing pickup, he will be holding something.
$\not\models$ (10) $\rightarrow$ (11) If ROB knows that there is an object at some location and that location happens to be where he is, then he need not know that he would be holding anything after doing a pickup action.

Additional properties are found in the extended version of the paper.

6 Reasoning

Given the representation of actions and their effects, we would like to have a method for addressing the *projection problem*. This is the question of determining whether or not some sentence G is true in the situation resulting from the execution of a sequence of ground action terms.

Following [Reiter, 1991; Scherl and Levesque, 1993] *regression* is used to reduce reasoning about future situations to reasoning about the initial situation. 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. Under these circumstances, the successor state axioms are only used to regress the formula $G(s_{gr})$. The result of the regression is a formula in a modal logic of indexical knowledge—i.e. a formula where the only situation term is S_0.

We have the following theorem, where $\mathcal{F}$ is the axiomatization of the domain including $\mathcal{F}_{ss}$, the successor state axioms. The theorem shows that reasoning with the successor state axioms need only be done with regression. 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 3 *For any ground situation term* s_{gr}

$$\mathcal{F} \models G(s_{gr}) \quad \leftrightarrow \quad \mathcal{F} - \mathcal{F}_{ss} \models \mathcal{R}^*_\Theta G(s_{gr})$$

Both the axiomatization of the initial situation and the regressed formula are expressions in a logic of indexical knowledge. The entailment test can be performed either by translating the sentences into first-order logic and using an ordinary theorem prover or more efficiently by using a theorem proving method for the indexical logic of knowledge. We have developed such a method based on the framework in [Frisch and Scherl, 1991]. Full details appear in the longer version of this paper.

The steps of the regression operator for fluents other than knowledge are the same as those found in [Reiter, 1991; Scherl and Levesque, 1993]. Steps **v** and **vi**, for knowledge, are given below. Two definitions are needed for the specification to follow. When φ is an arbitrary sentence and s a situation term, then $\varphi[s]$ is the sentence that results from substituting s for now throughout φ, but not within the scope of a **Knows** operator. The reverse operation φ^{-1} is the replacement of those situation variables that are not within the scope of a **Knows** operator by now.

In the definitions below, s' is a new situation variable.

v. Whenever a is not a knowledge-producing action,
$\mathcal{R}_\Theta[\mathbf{Knows}(agt, W, do(a, s))] = \mathbf{Knows}(agt, \mathcal{R}_\Theta[W[do(a, s')]]^{-1}, s)$.

vi. Whenever a is a knowledge producing action and α_i is the corresponding sentence in the successor state axiom for K

$$\begin{aligned}
&\mathcal{R}_\Theta[\mathbf{Knows}(agt, W, do(a, s))] = \\
&(\text{AGENT}(a) = agt \wedge ((\psi_i(s) \rightarrow \mathbf{Knows}(agt, \psi_i \rightarrow \mathcal{R}_\Theta[W[do(a, s')]]^{-1}, s)) \\
&\qquad\qquad \wedge (\neg\psi_i(s) \rightarrow \mathbf{Knows}(agt, \neg\psi_i \rightarrow \mathcal{R}_\Theta[W[do(a, s')]]^{-1}, s))) \\
&\vee (\text{AGENT}(a) \neq agt \wedge \mathbf{Knows}(agt, \mathcal{R}_\Theta[W[do(a, s')]]^{-1}, s))
\end{aligned}$$

Here the method is only illustrated with the example of Sentence 11. It is regressed to the following:

$$\begin{aligned}
\mathbf{Knows}(\text{ROB}, \exists y\, \text{OBJ}(y) \wedge \mathsf{pos}(\mathsf{self}, \mathsf{now}) = \text{POS}(y, \mathsf{now}) \\
\wedge \neg\exists x \text{HOLDING}(x, \mathsf{now})), s)
\end{aligned} \tag{12}$$

It can be verified that that Sentence 9 $\models$ 12, while Sentence 10 $\not\models$ 12.

7 Summary and Future Work

Knowledge producing actions such as sensing typically yield indexical knowledge, which is exactly what agents require to know what actions to perform to achieve their goals. We have extended the situation calculus with knowledge of [Scherl and Levesque, 1993] to deal with this feature of commonsense reasoning. In doing so, we have preserved the solution to the frame problem and adapted the regression method for reasoning about the effects of action to cope with indexical knowledge.

In this paper the assumption was made that all agents are aware of all actions carried out by other agents. Relaxing this assumption is being addressed in our current research. Other topics are developing methods for synthesizing plans that include sensing actions, as well as applications in robot programming.

References

[Agre and Chapman, 1990] Agre, P. E. and Chapman, D. 1990. What are plans for? *Robotics and Autonomous Systems* 6:17–34.

[Frisch and Scherl, 1991] Frisch, Alan and Scherl, Richard 1991. A general framework for modal deduction. In Allen, J.A.; Fikes, R.; and Sandewall, E., editors 1991, *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, San Mateo,CA : Morgan Kaufmann. 196–207.

[Lespérance and Levesque, 1995] Lespérance, Yves and Levesque, Hector 1995. Indexical knowledge and robot action—a logical account. *Artificial Intelligence* 73(1-2):69–115.

[Lespérance *et al.*, 1994] Lespérance, Yves; Levesque, Hector; Lin, Fangzhen; Marcu, Daniel; Reiter, Ray; and Scherl, Richard 1994. A logical approach to high-level robot programming — a progress report. Appears in *Control of the Physical World by Intelligent Systems*, Working Notes of the 1994 AAAI Fall Symposium, New Orleans, LA.

[Lespérance *et al.*, 1995] Lespérance, Yves; Levesque, Hector J.; Lin, F.; Marcu, Daniel; Reiter, Raymond; and Scherl, Richard B. 1995. Foundations of a logical approach to agent programming. To appear in *Proceedings of the IJCAI-95 Workshop on Agent Theories, Architectures, and Languages.*

[Lin and Reiter, 1994] Lin, Fangzhen and Reiter, Raymond 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678.

[Moore, 1985] Moore, R.C. 1985. A formal theory of knowledge and action. In Hobbs, J.R. and Moore, R.C., editors 1985, *Formal Theories of the Commonsense World.* Ablex, Norwood, NJ. 319–358.

[Reiter, 1991] Reiter, Raymond 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, Vladimir, editor 1991, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy.* Academic Press, San Diego, CA. 359–380.

[Scherl and Levesque, 1993] Scherl, Richard B. and Levesque, Hector J. 1993. The frame problem and knowledge producing actions. In *Proceedings, Eleventh National Conference on Artificial Intelligence.* 689–695.