State Constraints Revisited[1]

Fangzhen Lin and **Ray Reiter**[2]

Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
email: fl@ai.toronto.edu reiter@ai.toronto.edu

March 10, 1994

[1]To appear in the *Journal of Logic and Computation. Special Issue on Actions and Processes.* 1994.

[2]Fellow of the Canadian Institute for Advanced Research

Abstract

We pursue the perspective of Reiter that in the situation calculus one can formalize primitive, determinate actions with axioms which, among others, include two disjoint sets: a set of *successor state axioms* and a set of *action precondition axioms*. We posed ourselves the problem of automatically generating successor state axioms, given only a set of effect axioms and a set of state constraints. This is a special version of what has been traditionally called the *ramification problem*. To our surprise, we found that there are state constraints whose role is not to yield indirect effects of actions. Rather, *they are implicit axioms about action preconditions*. As such, they are intimately related to the classical *qualification problem*. We also discovered that other kinds of state constraints arise; these are related to the formalization of strategic or control information. This paper is devoted to describing our results along these lines, focusing on ramification and qualification state constraints. More specifically, we propose a two step procedure for determining an axiomatization which monotonically solves our versions of the ramification and qualification problems. We justify the first step semantically by appealing to a suitable minimization policy. Step two we justify by simple Clark predicate completion.

1 Introduction

We pursue the perspective of Reiter ([20]), that in the situation calculus one can formalize primitive,[1] determinate actions with axioms which, among others, include two disjoint sets: a set of *successor state axioms* and a set of *action precondition axioms*. Reiter shows how to obtain these successor state axioms by purely syntactic manipulations of the effect axioms (or "causal rules", as they are sometimes called). He does not, however, address the ramification problem; his approach fails in the presence of state constraints. In an attempt to remedy this, we posed ourselves the problem of automatically generating successor state axioms, given a set of effect axioms and a set of state constraints. To our surprise, we found that there are state constraints whose role is not to yield indirect effects of actions. Rather, *they are implicit axioms about action preconditions*. Such constraints relate to the qualification problem in much the same way as indirect effects do to the ramification problem. We later learned that the same observation had been made by Ginsberg and Smith [5]. We also discovered that other kinds of state constraints arise; these are related to the formalization of strategic or control information. This paper is devoted to describing our results along these lines, focusing on those constraints relevant to indirect effects and action preconditions. In particular, we propose a two step procedure for determining an axiomatization which monotonically solves our versions of the ramification and qualification problems. We justify the first step semantically by appealing to a suitable minimization policy. Step two we justify by simple Clark predicate completion.

[1] Roughly speaking, an action is primitive if it is not defined in terms of other actions. See Lespérance, Levesque, Lin, Reiter, and Scherl (forthcoming) for an approach to complex actions in the situation calculus.

2 The Problem

We illustrate with an example. In this paper, all free variables in a formula are considered to be universally quantified from the outside. Consider the painting blocks world in which we have the action $paint(x, y)$ which paints block x with color y, as represented by the following *effect axiom*:

$$Poss(paint(x, y), s) \supset color(x, y, do(paint(x, y), s)), \tag{1}$$

where for any action a, $Poss(a, s)$ means that action a is possible in s. A robot can perform $paint(x, y)$ only if it is close to x, and has the necessary paint:

$$Poss(paint(x, y), s) \supset nearby(x, s) \wedge haspaint(y, s). \tag{2}$$

The only state constraint is that a block can have just one color:

$$color(x, y_1, s) \wedge color(x, y_2, s) \supset y_1 = y_2. \tag{3}$$

It is well known that in addition to the above axioms, we need so-called *frame axioms*. In this case, for the property *color* we have:

$$\begin{aligned} Poss(paint(x, y), s) \supset [(\forall x', y').x \neq x' \supset \\ color(x', y', do(paint(x, y), s)) \equiv color(x', y', s)]. \end{aligned} \tag{4}$$

However, these axioms are still not enough since axiom (2) only allows us to infer when it is not possible to do $paint(x, y)$. In order to act, we need a guarantee of when it is possible to do *paint*. Normally, no absolute guarantee will exist. For example, in order to do $paint(x, y)$, the block x must be clear, the paint must not be frozen, etc. This dilemma is an instance of what has traditionally been called the *qualification problem* (McCarthy [12]). The assumption we need for solving this problem for this particular example is that every condition that prevents a robot from executing $paint(x, y)$ is implied by the axioms. Thus in this case, we have, by simple predicate completion (Clark [2]):

$$Poss(paint(x, y), s) \equiv nearby(x, s) \wedge haspaint(y, s). \tag{5}$$

One might expect that formally, this can be done by minimizing $\neg Poss$, as in (Lifschitz [8]). Unfortunately, in the general case, the minimization is not straightforward, especially in the presence of the frame problem. In particular, which do we solve first: the frame problem or the qualification problem? The answer in [8] is that we should solve the qualification problem first. However, if the axioms about *Poss* all have the form (2), which was implicitly assumed in [8], the order does not really matter. In general, we shall assume that one first has to have the necessary frame axioms before one can compute *Poss*. We shall see the motivation behind this assumption later. The same assumption is implicitly made in (Ginsberg and Smith [5]).

The qualification problem gets complicated when not all axioms about *Poss* are explicitly given as in (2). Traditionally, state constraints are considered to yield indirect effects of actions, and are the sources of the *ramification problem* (Finger [4]). However, as we shall see, *many state constraints are in fact implicit axioms*

about Poss. These state constraints complicate the qualification problem in much the same way as indirect effects do for the frame problem.

Imagine an ancient kingdom where yellow is reserved for the emperor. Thus for a robot, the world respects the rule:

$$\neg color(x, yellow, s). \tag{6}$$

As a result, a robot has to make sure that initially, no block is yellow, and he is not allowed to paint any block yellow:[2]

$$\neg Poss(paint(x, yellow), s),$$

which is simply a logical consequence of the effect axiom (1) and the constraint (6). Other such state constraints are not as straightforward, and are more fun. Imagine a more lenient emperor who tolerates a single yellow block but no more:

$$color(x_1, yellow, s) \wedge color(x_2, yellow, s) \supset x_1 = x_2. \tag{7}$$

The poor robot this time has to make sure that initially not more than one block can be yellow, and if there is already a yellow block, he cannot paint another block yellow without first painting the yellow block a different color:

$$(\exists x')(color(x', yellow, s) \wedge x \neq x') \supset \neg Poss(paint(x, yellow), s). \tag{8}$$

Notice that this time, the necessary condition for executing $paint(x, y)$ cannot be deduced from the effect axiom (1) and the state constraint (7) only. To derive it, we must use the frame axiom (4) as well. This is why we need to have determined the frame axioms before computing *Poss*.

This example illustrates the importance of distinguishing indirect-effect yielding state constraints such as (3) and other state constraints such as (7). Syntactically, the constraints (3) and (7) are very similar. But their *pragmatic* roles are very different. If, for example, we were to use (7) to compute the indirect effect of $paint(x, y)$, we would conclude as a logical consequence of (1) and (7) that painting a block yellow will "cause" an existing yellow block to change its color:

$$Poss(paint(x, yellow), s) \supset$$
$$(color(x', yellow, s) \wedge x \neq x') \supset \neg color(x', yellow, do(paint(x, yellow), s)).$$

On the other hand, if (3) were not taken into account in generating the frame axiom, then from the effect axiom (1), by the law of inertia, we have

$$Poss(paint(x, y), s) \supset [y' \neq y \supset$$
$$color(x', y', do(paint(x, y), s)) \equiv color(x', y', s)].$$

Now if we were to use (3) to compute *Poss*, then from it and the above frame axiom, we would conclude, incorrectly, that it is impossible to paint any block a different color:

$$y \neq y' \wedge color(x, y', s) \supset \neg Poss(paint(x, y), s).$$

[2]But yellow paint may still be available for the purpose of, say, making orange paint.

We conclude that the state constraints (3) and (7) play different roles in our axiomatization. (3) contributes to the ramifications of *paint*, while (7) contributes new information about *Poss*. Moreover, reversing their roles leads to counterintuitive results.

In the following, we shall call those indirect-effect yielding state constraints *ramification constraints*, and those yielding action preconditions *qualification constraints*. As in (Ginsberg and Smith [5]), we shall explicitly distinguish these two kinds of constraints. In summary, ramification constraints cause complications, traditionally called the ramification problem, to the frame problem. Qualification constraints cause a symmetric problem to the qualification problem. In the next section we present our version of the situation calculus. We then describe our approach to solving these problems.

3 The Discrete Situation Calculus

It has become clear to us that any serious study of the situation calculus requires foundational principles, analogous in many ways to the Peano foundational axioms for number theory. While it is not obvious what counts as an appropriate axiomatization of the "Peano situation calculus", we have found the following axioms to be intuitively appealing and useful in many ways. Formulating a "one and true" set of foundational axioms and exploring their meta-mathematical properties remains an ongoing goal of our research program.

As usual, we shall use a many-sorted language for the situation calculus. The two domain independent sorts are *situation* and *action*. We emphasize that sort *action* is for primitive actions. There is a unique situation constant symbol, S_0, denoting the initial situation. It is like the number 0 in Peano arithmetic. Unlike Peano arithmetic which has a unique successor function, we have a family of successor functions modeled by the binary function $do : action \times situation \rightarrow situation$.

In place of the Peano axioms, we have the following:

$$S_0 \neq do(a, s), \tag{9}$$

$$do(a_1, s_1) = do(a_2, s_2) \supset (a_1 = a_2 \wedge s_1 = s_2), \tag{10}$$

$$(\forall P)[P(S_0) \wedge (\forall a, s)(P(s) \supset P(do(a, s))) \supset (\forall s)P(s)]. \tag{11}$$

The first two axioms are unique names assumptions. They eliminate finite cycles, and merging. The last axiom is second order induction. It amounts to the domain closure axiom that every situation has to be obtained from the initial one by repeatly applying the function *do*. For a discussion of the use of induction in the situation calculus, see (Reiter [22]).

In reality, an action is not always executable in every situation. To formalize this, we use a binary predicate $Poss(a, s)$. We write $s < s'$ if s' can be obtained from s by a sequence of executable actions. Inductively, we have:

$$\neg s < S_0, \tag{12}$$

$$s < do(a, s') \equiv (Poss(a, s') \wedge s \leq s'), \tag{13}$$

where $s \leq s'$ is shorthand for $s < s' \vee s = s'$.

In the following, we shall denote by Σ the set of axioms we have so far, including the ones for $<$. It is clear that because of the second-order induction axiom, the situation domain of any model of Σ must be isomorphic to the smallest set $\mathcal{S}$ satisfying:

1. $S_0 \in \mathcal{S}$.
2. If $S \in \mathcal{S}$, and $A \in \mathcal{A}$, then $do(A, S) \in \mathcal{S}$, where $\mathcal{A}$ is the domain of actions in the model.

That is, Σ is *categorical* for the sort *situation*. These axioms have their origin in (Reiter [19, 22]). Similar axioms are used in (Pinto and Reiter [18]). The following proposition summarizes some simple consequences of Σ. All proofs are given in the appendix.

Proposition 1

Transitivity:	$(s_1 < s_2 \wedge s_2 < s_3) \supset s_1 < s_3$.
Anti-reflexivity:	$\neg s < s$.
Unique names:	$s_1 < s_2 \supset s_1 \neq s_2$.
Induction on $<$:	$(\forall P)[P(S_0) \wedge (\forall a, s)(P(s) \wedge Poss(a, s) \supset P(do(a, s))) \supset$ $(\forall s)(S_0 \leq s \supset P(s))]$.

In addition to the domain independent sorts, there may be other domain dependent sorts. In the sequel, we shall assume a fixed additional sort *object* for objects in the domain of interest. We shall use x, y, and their primed and subscripted versions to denote variables ranging over the domain of *object*. An (n+1)-ary, $n \geq 0$, *fluent* will be a predicate of arity: $object^n \times situation$. An n-ary action *prototype* will be a function of arity: $object^n \rightarrow action$. In the following, we shall assume a set of *unique names axioms*, denoted by $\mathcal{D}_{una}$, for actions. These axioms have the forms:

$$f(x_1, ..., x_n) = f(y_1, ..., y_n) \supset (x_1 = y_1 \wedge \cdots \wedge x_n = y_n),$$

$$f(x_1, ..., x_m) \neq g(y_1, ..., y_n),$$

where f and g are different action prototypes.

4 The Ramification Problem

This section deals with the frame problem in the presence of ramification constraints. Several solutions have been proposed in the literature (cf. Baker [1], Lin and Shoham [11], and others.), most of them based on circumscription (McCarthy [13]). There seems to be a consensus that these different minimization semantics turn out to be equivalent for determinate actions (cf. Costello [3], Kartha [7]).

This section defines precisely the minimization policy used in this paper. It is based on the one in (Lin and Shoham [11]). There are however some differences. First, in (Lin and Shoham [11]), the minimization is done with respect to a fixed action. The minimization here is for all actions simultaneously. Secondly, the axioms

in (Lin and Shoham [11]) do not deal with $Poss$ predicate. Reflecting our decision to solve the frame problem first, our minimization policy shall fix $Poss$.

To motivate our minimization policy, consider a language with a single action, and only two states: the initial one S_i, and the resulting state after the action is performed S_r (cf. Lifschitz [9, 10]). For every fluent F, we replace F by two predicates F_i and F_r. For every tuple $(x_1, ..., x_n)$, $F_i(x_1, ..., x_n)$ stands for $F(x_1, ..., x_n, S_i)$, and $F_r(x_1, ..., x_n)$ for $F(x_1, ..., x_n, S_r)$.

Now consider the action *load* that loads the gun. Suppose we have two unary fluents *loaded* and *alive*. These two fluents yield the following four propositions: $loaded_i$ (the gun is initially loaded), $loaded_r$ (the gun is loaded after performing *load*), $alive_i$ (Fred is alive initially), and $alive_r$ (Fred is alive after performing *load*). Let T consist of the following axioms:

$$loaded_r,$$
$$ab1 \equiv \neg(loaded_i \equiv loaded_r),$$
$$ab2 \equiv \neg(alive_i \equiv alive_r).$$

Intuitively, we want to minimize $ab1$ and $ab2$ in T to formalize the law of inertia. Now the question is which predicates should be fixed, and which should be allowed to vary. According to McCarthy's original proposal, all predicates are allowed to vary, and we conclude, counter-intuitively, that

$$\neg ab1 \wedge \neg ab2,$$

which implies $loaded_i$ (the gun is loaded initially.). A little thought reveals that we should fix the initial situation, i.e. $loaded_i$ and $alive_i$, and allow the resulting situation to vary. Formally, if we minimize $ab1$ and $ab2$ in T, with $loaded_i$ and $alive_i$ fixed, we'll conclude correctly that

$$(ab1 \equiv \neg loaded_i) \wedge \neg ab2.$$

Now if we use, as we normally do, the situation calculus to formalize actions whose effects depend only on the starting situation, i.e., the past and the future are irrelevant, then it is sufficient to capture the effects of an action with respect to two generic starting and resulting situations. As we have seen, this should be done by minimizing changes with the starting situation fixed. This is exactly what the following minimization policy does.

Recall that our language has three sorts: *situation*, *action*, and *object*. In the following, we shall divide a variable assignment σ into three parts: σ_s, σ_a, and σ_o. The first assigns variables of sort *situation*, the second variables of sort *action*, and the third variables of sort *object*.

Definition 2 A model M of a theory W is *minimal* iff there is not another model M' of W, and a variable assignment to situations σ_s such that

1. M and M' have the same universe.
2. M and M' differ only in their interpretations of fluents. In particular, they interpret $Poss$ the same.

3. For any assignment σ_o, and any fluent $F(x_1, ..., x_n, s)$,

$$M, \sigma_s, \sigma_o \models F(x_1, ..., x_n, s)$$

iff

$$M', \sigma_s, \sigma_o \models F(x_1, ..., x_n, s).$$

4. For any assignments σ_a and σ_o, and any fluent $F(x_1, ..., x_n, s)$, if

$$M, \sigma_s, \sigma_a, \sigma_o \models Poss(a, s) \wedge \neg ab(a, F(x_1, ..., x_n, s)),$$

then

$$M', \sigma_s, \sigma_a, \sigma_o \models \neg ab(a, F(x_1, ..., x_n, s)),$$

where $ab(a, F(x_1, ..., x_n, s))$ is a shorthand for

$$\neg F(x_1, ..., x_n, s) \equiv F(x_1, ..., x_n, do(a, s)).$$

5. There are two assignments σ_a and σ_o, and a fluent $F(x_1, ..., x_n, s)$ such that

$$M, \sigma_s, \sigma_a, \sigma_o \models Poss(a, s) \wedge ab(a, F(x_1, ..., x_n, s))$$

but

$$M', \sigma_s, \sigma_a, \sigma_o \models \neg ab(a, F(x_1, ..., x_n, s)).$$

4.1 A Circumscriptive Specification for the Ramification Problem

With the above minimization policy in hand, we are now in a position to provide what we take to be a semantic specification of what counts as a solution to the ramification problem. To do so, we must precisely characterize the axioms to be minimized. In general, we suppose that, in addition to the foundational axioms Σ and the unique names axioms for actions $\mathcal{D}_{una}$, we are given the following two sets of axioms:

1. A set $\mathcal{D}_{ef}$ of effect axioms of the form

$$Poss(a, s) \supset [\Psi^+ \supset F(x_1, ..., x_n, do(a, s))],$$

and of the form

$$Poss(a, s) \supset [\Psi^- \supset \neg F(x_1, ..., x_n, do(a, s))],$$

where F is an (n+1)-ary fluent, and Ψ^+ and Ψ^- are *simple state formulas* whose free variables are among $a, s, x_1, ..., x_n$. A simple state formula is one which does not mention $Poss$, $<$, or any situation term other than a unique situation variable.

2. A set $\mathcal{D}_{ram}$ of ramification constraints of the form:

$$RC(s),$$

where RC is a simple state formula with a unique free variable s.

With these axioms in hand, we identify a solution to the ramification problem with those minimal models of $\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ram} \cup \mathcal{D}_{ef}$.

In the next section, we give an independently motivated syntactic procedure which, given $\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ram} \cup \mathcal{D}_{ef}$, computes a certain monotonic theory and we show that, under suitable circumstances, this monotonic theory is logically equivalent to the above minimal model nonmonotonic theory.

5 How to Compute Successor State Axioms (Sometimes)

In [20] Reiter shows how to address the frame problem in the absence of state constraints. The idea there was to use the given effect axioms to compute a set of *explanation closure axioms* (Schubert [23]). On the assumption that the given effect axioms completely describe the causal laws of the domain being axiomatized, these explanation closure axioms can be seen intuitively to have the force of frame axioms. Moreover, under a natural consistency assumption on these axioms, the effect and explanation closure axioms can be shown to be logically equivalent to a set of *successor state axioms*. These have a particularly simple and computationally useful form, as we shall describe below. In other words, *when all the effect axioms are in hand*, [20] provides a conceptually and computationally simple solution to the frame problem. Unfortunately, this solution to the frame problem no longer applies in the presence of ramification state constraints. The reason is that this solution relies on the prior availability of all the effect axioms; these must be explicitly in hand before the explanation closure, and hence the successor state axioms, can be computed. When state constraints are present, it might be possible to derive new effect axioms from the old effect axioms together with the constraints. In other words, one can no longer be certain that all the effect axioms will be explicitly in hand for the purpose of determining successor state axioms. It is this possibility of *implicit* effect axioms that makes the frame problem so difficult when there are state constraints present. If we could make all these implicit effect axioms explicit (through deduction) using the given effect axioms and ramification constraints, we could then compute the successor state axioms, thereby solving the frame problem. This is the intuition behind the approach to the frame problem of this section.

Given an axiomatization $\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ram} \cup \mathcal{D}_{ef}$ as described above, our task is to generate a successor state axiom (Reiter [20]) for each fluent $F(x_1, ..., x_n, s)$:

$$Poss(a, s) \supset F(x_1, ..., x_n, do(a, s)) \equiv \Phi_F, \tag{14}$$

where Φ_F is a simple state formula whose free variables are among $s, a, x_1, ..., x_n$. Having explicit successor state axioms is desirable for two reasons. First, as shown by Reiter [20], successor state axioms are computationally appealing in that they allow regression (Waldinger [25], Pednault [16], and others), a property we shall shortly exploit. Secondly, it is clear that having a set of successor state axioms completely formalizes the effects of actions on the fluents.

The discussion at the beginning of the section suggests that for every (n+1)-ary fluent $F(x_1, ..., x_n, s)$, we should determine all positive effect axioms for F derivable

from the state constraints and the effect axioms (including, of course, the effect axiom for F of $\mathcal{D}_{ef}$ itself):

$$\begin{aligned}
&\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ef} \cup \mathcal{D}_{ram} \models \\
&\quad (\forall a, s, x_1, ..., x_n).Poss(a, s) \supset [\varphi_1(x_1, ..., x_n, a, s) \supset F(x_1, ..., x_n, do(a, s))], \\
&\quad \vdots \\
&\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ef} \cup \mathcal{D}_{ram} \models \\
&\quad (\forall a, s, x_1, ..., x_n).Poss(a, s) \supset [\varphi_k(x_1, ..., x_n, a, s) \supset F(x_1, ..., x_n, do(a, s))],
\end{aligned}$$

where $\varphi_1, ..., \varphi_k$ are simple state formulas. When we are sure that we have got them all, we collect them into a single effect axiom:

$$\begin{aligned}
&\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ef} \cup \mathcal{D}_{ram} \models \\
&\quad (\forall a, s, x_1, ..., x_n).Poss(a, s) \supset [\Psi_F \supset F(x_1, ..., x_n, do(a, s))],
\end{aligned} \tag{15}$$

where $\Psi_F(x_1, ..., x_n, a, s)$ is a simple state formulas equivalent to $\varphi_1 \vee \cdots \vee \varphi_k$. Similarly, we find a single negative effect axiom for F:

$$\begin{aligned}
&\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ef} \cup \mathcal{D}_{ram} \models \\
&\quad (\forall a, s, x_1, ..., x_n).Poss(a, s) \supset [\Psi_{\neg F} \supset \neg F(x_1, ..., x_n, do(a, s))].
\end{aligned} \tag{16}$$

Now if Ψ_F ($\Psi_{\neg F}$) is indeed the most general condition under which F will be true (false) in situation $do(a, s)$, then the argument given in (Reiter [20]) yields the following successor state axiom for F:

$$Poss(a, s) \supset F(x_1, ..., x_n, do(a, s)) \equiv [\Psi_F \vee (F(x_1, ..., x_n, s) \wedge \neg\Psi_{\neg F})].$$

But how are we to know when we have determined all the positive and negative effect axioms for F or, what amounts to the same thing, when Ψ_F ($\Psi_{\neg F}$) is indeed the most general condition under which F will be made true (false)? The following theorem gives sufficient conditions under which the successor state axioms determined in this way are indeed correct with respect to the minimal model semantics of Section 4.

Theorem 3 Let $\mathcal{D}_{ss}$ be a set of the successor state axioms, one for each fluent, of the form:

$$Poss(a, s) \supset F(x_1, ..., x_n, do(a, s)) \equiv [\Psi_F \vee (F(x_1, ..., x_n, s) \wedge \neg\Psi_{\neg F})]. \tag{17}$$

where $F(x_1, ..., x_n, s)$ is an (n+1)-ary fluent, and Ψ_F and $\Psi_{\neg F}$ satisfy (15) and (16), respectively. Suppose further that:

1. The following *consistency condition* (Reiter [20]) holds for each F:

$$\mathcal{D}_{una} \models (\forall s, a, x_1, ..., x_n)\neg(\Psi_F \wedge \Psi_{\neg F}). \tag{18}$$

2. The ramification constraints relativised to states accessible from S_0 are derivable from the successor state axioms:

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \{RC(S_0)\} \models (\forall s).S_0 \leq s \supset RC(s) \tag{19}$$

for every $(\forall s)RC(s) \in \mathcal{D}_{ram}$.

Then a first-order structure is a minimal model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ram} \cup \mathcal{D}_{ef}$$

iff it is a model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}^{-}_{ram} \cup \mathcal{D}^{S_0}_{ram},$$

where $\mathcal{D}^{-}_{ram}$ is the set of ramification constraints relativised to states which are not accessible from S_0:

$$\mathcal{D}^{-}_{ram} = \{(\forall s).S_0 \not\leq s \supset RC(s) \mid (\forall s)RC(s) \in \mathcal{D}_{ram}\},$$

and $\mathcal{D}^{S_0}_{ram}$ is the set of ramification constraints restricted to the initial state:

$$\mathcal{D}^{S_0}_{ram} = \{RC(S_0) \mid (\forall s)RC(s) \in \mathcal{D}_{ram}\}.$$

1. The theorem says that under conditions (18) and (19), the nonmonotonic theory $\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ram} \cup \mathcal{D}_{ef}$ (according to the minimization policy of Section 4) is captured by the monotonic theory

   $$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}^{-}_{ram} \cup \mathcal{D}^{S_0}_{ram}.$$

2. More generally, it can be shown that for any initial state description $\mathcal{D}_{S_0}$, if $\mathcal{D}_{S_0} \models \mathcal{D}^{S_0}_{ram}$, and the conditions of the theorem are satisfied, then the nonmonotonic theory

   $$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ram} \cup \mathcal{D}_{ef} \cup \mathcal{D}_{S_0}$$

   is equivalent to the monotonic one

   $$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}^{-}_{ram} \cup \mathcal{D}_{S_0}.$$

3. Notice that $\mathcal{D}^{-}_{ram}$ must be included in the monotonic theory because the successor state axioms characterize action effects only when the action is possible. In practise, this turns out to be of no consequence because normally, the monotonic theory will be used to derive entailments which are relativised only to situations accessible from S_0. For example, for AI planning applications, one is concerned with deriving entailments of the form $(\exists s).S_0 \leq s \wedge G(s)$ where $G(s)$ defines a goal state (Green [6], Reiter [20]). In database applications (Reiter [21]), one is concerned with querying a database following an update transaction sequence $\mathbf{T}$. Such queries have the form $Q(do(\mathbf{T}, S_0))$ where $do(\mathbf{T}, S_0)$ denotes that database state resulting from performing the sequence of updates $\mathbf{T}$ starting in the initial database state S_0. Normally, the state $do(\mathbf{T}, S_0)$ will be known to be accessible from S_0, which is to say, the preconditions for "executing" the transaction sequence $\mathbf{T}$ will be known to be true (Reiter [21] calls such states *legal* states). For both these settings (planning problems, database query evaluation), the entailments of interest are relativised only to situations accessible from S_0. It should be clear that for the purpose of deriving entailments of this kind, the sentences in $\mathcal{D}^{-}_{ram}$ have no role to play:

   For the purpose of deriving entailments which are relativised only to situations accessible from S_0, $\mathcal{D}^{-}_{ram}$ can be ignored; it is sufficient to use the monotonic theory $\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{S_0}$.

We shall consider the general case of the above theorem in greater detail in a moment. First, we treat an important special case. If $\mathcal{D}_{ram} = \emptyset$, then the elementary syntactic manipulations in (Reiter [20]) are provably correct with respect to our minimal model semantics:[3]

Corollary 4 [Correctness of Successor State Axioms in the Absence of State Constraints (Reiter [20])] Suppose $\mathcal{D}_{ram} = \emptyset$, and without loss of generality, for each fluent F, suppose that $\mathcal{D}_{ef}$ contains exactly one positive effect axiom for F as given by (15), and exactly one negative effect axiom for F as given by (16). Then the successor state axiom defined in (Reiter [20]) for F is exactly (17). Moreover, if the consistency condition (18) holds for each F, then a first-order structure is a model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss}$$

iff it is a minimal model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ef}.$$

We now return to the general case of Theorem 3. The theorem suggests the following approach to the derivation of successor state axioms when given ramification constraints $\mathcal{D}_{ram}$ and effect axioms $\mathcal{D}_{ef}$:

1. Using $\mathcal{D}_{ram}$ and $\mathcal{D}_{ef}$, derive a set of effect axioms of the forms (15) and (16), and compute the corresponding successor state axioms (17).

2. Stop deriving new effect axioms whenever it has been verified that the entailments (18) and (19) hold. In this case, if $\mathcal{D}_{ss}$ are the corresponding successor state axioms, then the equivalent monotonic theory is

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ram}^{-} \cup \mathcal{D}_{ram}^{S_0}.$$

Clearly, this procedure is not guaranteed to terminate. Pinto [17] studies some simple classes of state constraints for which a set of successor state axioms can be effectively computed. Here, we shall only consider some examples. Before doing so, we prove a lemma to facilitate the proof of condition (19), which will normally require induction on situations (Reiter [22]), using Proposition 1. The lemma we are about to present "precomputes" the induction proof, reducing the theorem proving task (19) to a very simple entailment using only unique names axioms for actions.

To formulate the lemma, we first require the concept of the regression of a formula Ψ: $\mathcal{R}_{\mathcal{D}_{ss}}[\Psi]$, the *regression* of Ψ under the set of successor state axioms $\mathcal{D}_{ss}$ (cf. Waldinger [25], Pednault [16], and Reiter [20]), is the result of substituting $\Phi_F(t_1, ..., t_n, \alpha, \sigma)$ for every subformula $F(t_1, ..., t_n, do(\alpha, \sigma))$ mentioned by Ψ, where this substitution is performed for every fluent F mentioned by Ψ. Here, $\Phi_F(x_1, ..., x_n, a, s)$ is as in the successor state axiom (14). We shall write $\mathcal{R}_{\mathcal{D}_{ss}}[\Psi]$ as $\mathcal{R}[\Psi]$ when there is no possibility of confusion. For instance, if we have the following successor state axiom:

$$Poss(a, s) \supset [F(do(a, s)) \equiv (\neg F(s) \wedge a = flip) \vee (F(s) \wedge a \neq flip)],$$

[3] This was shown independently for propositional fluents by Steven Shapiro [24].

then the regression of $F(do(flip, S_0))$ is

$$(\neg F(S_0) \wedge flip = flip) \vee (F(S_0) \wedge flip \neq flip),$$

and the regression of $(\forall s)(\exists a).F(s) \supset \neg F(do(a, s))$ is

$$(\forall s)(\exists a).F(s) \supset \neg[(\neg F(s) \wedge a = flip) \vee (F(s) \wedge a \neq flip)].$$

Notice that regression is a purely syntactic manipulation, and may not preserve logical properties. For instance, the regressions of the two logically equivalent formulas, $(\forall s).F(s)$ and $(\forall s, a).F(s) \wedge F(do(a, s))$, are generally not equivalent.

Lemma 5 For any ramification state constraint $RC(s)$,

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \{RC(S_0)\} \models (\forall s).S_0 \leq s \supset RC(s)$$

iff

$$\mathcal{D}_{una} \models (\forall s, a).RC(s) \wedge Poss(a, s) \supset \mathcal{R}[RC(do(a, s))].$$

Example 6 We now derive a set of successor state axioms for our painting example. First, we rewrite the causal rule (1) as

$$Poss(a, s) \supset a = paint(x, y) \supset color(x, y, do(a, s)). \tag{20}$$

Now rewrite the ramification constraint (3) as

$$color(x, y, s) \supset y \neq y' \supset \neg color(x, y', s).$$

From these two axioms, we infer

$$\begin{array}{l} Poss(a, s) \supset \\ \quad (\exists y').(a = paint(x, y') \wedge y \neq y') \supset \neg color(x, y, do(a, s)). \end{array}$$

Using this axiom and the causal rule (20), with Ψ_{color} as

$$a = paint(x, y),$$

and $\Psi_{\neg color}$ as

$$(\exists y').a = paint(x, y') \wedge y \neq y',$$

we get the successor state axiom

$$\begin{array}{l} Poss(a, s) \supset \{color(x, y, do(a, s)) \equiv \\ \quad a = paint(x, y) \vee [color(x, y, s) \wedge \neg(\exists y')(a = paint(x, y') \wedge y \neq y')]\}. \end{array}$$

Since there are no effect axioms about $nearby(x, s)$, we let Ψ_{nearby} and $\Psi_{\neg nearby}$ be $False$. Thus we get

$$\begin{array}{l} Poss(a, s) \supset \\ \quad nearby(x, do(a, s)) \equiv [False \vee (nearby(x, s) \wedge \neg False)], \end{array}$$

which is equivalent to

$$Poss(a, s) \supset nearby(x, do(a, s)) \equiv nearby(x, s).$$

Similarly for *haspaint*, we have

$$Poss(a, s) \supset haspaint(x, do(a, s)) \equiv haspaint(x, s).$$

It remains to check conditions (18) and (19) of Theorem 3 to verify that the set of successor state axioms obtained is the right one:

1. *Consistency conditions:*

Unique names axioms for actions entail $\neg(\Psi_{color} \wedge \neg\Psi_{\neg color})$. $\neg(\Psi_{nearby} \wedge \neg\Psi_{\neg nearby})$ and $\neg(\Psi_{haspaint} \wedge \neg\Psi_{\neg haspaint})$ are identically *True*.

2. *Deriving the ramification constraints:*

There is only one state constraint; let us denote it by $(\forall s)RC(s)$, where $RC(s)$ is

$$(\forall x, y, y').color(x, y, s) \wedge color(x, y', s) \supset y = y'.$$

By Lemma 5, we need to show that

$$\mathcal{D}_{una} \cup \{RC(s) \wedge Poss(a, s)\} \cup \{\mathcal{R}[color(x, y, do(a, s)) \wedge color(x, y', do(a, s))]\} \models y = y'.$$

Using the above successor state axiom for *color*, which can be equivalently written as

$$\begin{aligned} &Poss(a, s) \supset \{color(x, y, do(a, s)) \equiv \\ &\quad a = paint(x, y) \vee [color(x, y, s) \wedge (\forall y'')a \neq paint(x, y'')]\}, \end{aligned}$$

we need to show that

$$\begin{aligned} &\mathcal{D}_{una} \cup \{RC(s), Poss(a, s), \\ &\quad a = paint(x, y) \vee [color(x, y, s) \wedge (\forall y'')a \neq paint(x, y'')], \\ &\quad a = paint(x, y') \vee [color(x, y', s) \wedge (\forall y'')a \neq paint(x, y'')]\} \\ &\quad\quad \models y = y'. \end{aligned}$$

This can be proved by cases, as follows:

- $\mathcal{D}_{una} \cup \{RC(s), Poss(a, s), a = paint(x, y) \wedge a = paint(x, y')\} \models y = y'$. This follows from the unique names axioms for actions.
- $\mathcal{D}_{una} \cup \{RC(s), Poss(a, s), a = paint(x, y) \wedge color(x, y', s) \wedge (\forall y'')a \neq paint(x, y'')\} \models y = y'$. This holds since the premises are inconsistent.
- $\mathcal{D}_{una} \cup \{RC(s), Poss(a, s), a = paint(x, y') \wedge color(x, y, s) \wedge (\forall y'')a \neq paint(x, y'')\} \models y = y'$. This holds again by the inconsistency of the premises.
- $\mathcal{D}_{una} \cup \{RC(s), Poss(a, s), color(x, y, s) \wedge color(x, y', s) \wedge (\forall y'')a \neq paint(x, y'')\} \models y = y'$. This follows from $RC(s)$ and $color(x, y, s) \wedge color(x, y', s)$.

In general, the proof-theoretic procedure is not complete simply because there may be multiple minimal models. Even if the minimal model is unique, it may not be captured by a set of successor state axioms. In fact, it may not even be captured by any first-order theory.

Example 7 Let $P(x, y, s)$ and $R(x, y, s)$ be ternary fluents, and A an action constant. Consider $\mathcal{D} = (\mathcal{D}_{ef}, \mathcal{D}_{ram})$, where

1. $\mathcal{D}_{ef}$ is
$$Poss(A, s) \supset [P(x, y, s) \supset R(x, y, do(A, s))].$$
2. $\mathcal{D}_{ram}$ is
$$R(x, x, s),$$
$$R(x, y, s) \supset R(y, x, s),$$
$$[R(x, y, s) \wedge R(y, z, s)] \supset R(x, z, s).$$

There cannot be a first-order theory capturing the minimal model of $\mathcal{D}$. Otherwise, the theory together with the following first-order sentence

$$(\forall a, s).Poss(a, s) \wedge (\forall x, y).\neg R(x, y, S_0)$$

will entail the transitive closure of $P(x, y, S_0)$, which is well-known not to be first-order definable.

6 Determining Action Precondition Axioms

This section concerns the qualification problem, and is to a large degree independent of the last section. We assume that by whatever means, we have succeeded in obtaining a set of successor state axioms $\mathcal{D}_{ss}$. Symmetric to our treatment of the ramification problem, we assume that we are given the following two sets:

1. A set $\mathcal{D}_{nec}$ of direct necessary conditions for $Poss$ of the form:
$$Poss(A(x_1, ..., x_n), s) \supset \Pi_A,$$
where $A(x_1, ..., x_n)$ is an n-ary action prototype, and Π_A is a simple state formula whose free variables are among $s, x_1, ..., x_n$.
2. A set $\mathcal{D}_{qual}$ of qualification constraints of the form:
$$QC(s),$$
where QC is a simple state formula with a unique free variable s.

Our goal is to generate a set of *action precondition axioms*, one for each action prototype, of the form:

$$Poss(A(x_1, ..., x_n), s) \equiv \Theta_A, \tag{21}$$

where Θ_A is a simple state formula whose free variables are among $s, x_1, ..., x_n$.

If $(\forall s).QC(s)$ is in $\mathcal{D}_{qual}$, that is, a qualification constraint, then it is clear that for each action $A(x_1, ..., x_n)$:

$$\mathcal{D}_{ss} \models (\forall x_1, ..., x_n, s).Poss(A(x_1, ..., x_n), s) \supset \\ (\mathcal{R}[QC(do(A(x_1, ..., x_n), s))] \equiv QC(do(A(x_1, ..., x_n), s))),$$

where $\mathcal{R}$ is the regression operator as defined in section 5. Therefore we have

$$\mathcal{D}_{ss} \cup \mathcal{D}_{qual} \models (\forall x_1, ..., x_n, s).Poss(A(x_1, ..., x_n), s) \supset \mathcal{R}[QC(do(A(x_1, ..., x_n), s))].$$

The formula $\mathcal{R}[QC(do(A(x_1, ..., x_n), s))]$ as defined is usually unwieldy. In most cases, it will need to be simplified by using $\mathcal{D}_{qual}$ and the unique names axioms in $\mathcal{D}_{una}$, as we shall see later in an example. Let Π_{QC} be a simple state formula such that

$$\mathcal{D}_{una} \cup \mathcal{D}_{qual} \models (\forall x_1, ..., x_n, s).\mathcal{R}[QC(do(A(x_1, ..., x_n), s))] \equiv \Pi_{QC}.$$

Then

$$\mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{qual} \models (\forall x_1, ..., x_n, s).Poss(A(x_1, ..., x_n), s) \supset \Pi_{QC}.$$

Now, without loss of generality, suppose that the only necessary condition axiom for A in $\mathcal{D}_{nec}$ is:

$$Poss(A(x_1, ..., x_n), s) \supset \Pi_A.$$

Then we obtain the following action precondition axiom by predicate completion (Clark [2]) of $Poss$:

$$Poss(A(x_1, ..., x_n), s) \equiv \Pi_A \wedge \bigwedge \Pi_{QC}, \tag{22}$$

where the big conjunction ranges over the state constraints in $\mathcal{D}_{qual}$. Let $\mathcal{D}_{pre}$ be the set of action precondition axioms thus obtained.

Theorem 8 Let $\mathcal{D}_{ss}$, $\mathcal{D}_{qual}$ and $\mathcal{D}_{pre}$ be as given above. Let $\mathcal{D}_{cls}$ be the following domain closure axiom for actions:

$$(\forall a)((\exists \vec{x})a = A_1(\vec{x}) \vee \cdots \vee (\exists \vec{y})a = A_n(\vec{y})),$$

where $A_1, ..., A_n$ are action prototypes. For every qualification constraint $(\forall s)QC(s) \in \mathcal{D}_{qual}$,

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{pre} \cup \mathcal{D}_{cls} \models QC(S_0) \supset (\forall s).S_0 \leq s \supset QC(s).$$

Proof Use the principle of induction on $<$ in Proposition 1

Corollary 9

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{pre} \cup \mathcal{D}_{cls} \cup \mathcal{D}^{-}_{qual} \cup \mathcal{D}^{S_0}_{qual} \models \mathcal{D}_{qual},$$

where $\mathcal{D}^{-}_{qual}$ is the set of qualification constraints relativised to states which are not accessible from S_0:

$$\mathcal{D}^{-}_{qual} = \{(\forall s).S_0 \not\leq s \supset QC(s) \mid (\forall s)QC(s) \in \mathcal{D}_{qual}\},$$

and $\mathcal{D}^{S_0}_{qual}$ is the set of qualification constraints restricted to the initial state:

$$\mathcal{D}^{S_0}_{qual} = \{QC(S_0) \mid (\forall s)QC(s) \in \mathcal{D}_{qual}\}.$$

Why is this corollary of interest? It tells us that provided the initial state S_0 satisfies all the qualification constraints $(\forall s)QC(s)$, i.e. that $QC(S_0)$ is entailed by $\mathcal{D}_{S_0}$, the axioms specifying the initial state, then these constraints are entailed by the background theory

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{pre} \cup \mathcal{D}_{cls} \cup \mathcal{D}^{-}_{qual} \cup \mathcal{D}_{S_0}.$$

This means that the ramification constraints can now be discarded; their effects have been "compiled" into the action precondition axioms (22). This highlights the difference between our solution to the qualification problem and that of (Ginsberg and Smith [5]). In [5], action qualifications are never stored, and have to be computed each time an action is attempted. Finally, notice that the background axioms for Corollary 9 include $\mathcal{D}^{-}_{qual}$, the original qualification constraints relativised to states inaccessible from S_0. Recall that precisely the same situation arose in our earlier treatment of ramification constraints (Theorem 3), where $\mathcal{D}^{-}_{ram}$ had to be included in the monotonic theory equivalent to the nonmonotonic minimal model semantics. We observed there that whenever the theory is used to derive entailments which are relativised only to situations accessible from S_0 (the normal case), the axioms $\mathcal{D}^{-}_{ram}$ will play no role and can be discarded. Exactly the same observation applies in the case of Corollary 9:

For the purpose of deriving entailments which are relativised only to situations accessible from S_0, $\mathcal{D}^{-}_{qual}$ can be ignored; it is sufficient to use the background axioms

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{pre} \cup \mathcal{D}_{cls} \cup \mathcal{D}_{S_0}.$$

Example 10 We now derive action preconditions for our painting example. $\mathcal{D}_{nec}$ consists of axiom (2):

$$Poss(paint(x, y), s) \supset nearby(x, s) \wedge haspaint(y, s).$$

We shall consider the world where not more than one yellow block is allowed; thus $\mathcal{D}_{qual}$ consists of (7). Write the qualification constraint as $(\forall s)QC$. Then $QC(s)$ is

$$(\forall x_1, x_2).color(x_1, yellow, s) \wedge color(x_2, yellow, s) \supset x_1 = x_2.$$

Let $\mathcal{D}_{ss}$ be the set of successor state axioms generated by Example 6. By the unique names axiom for actions:

$$paint(x_1, y_1) = paint(x_2, y_2) \supset (x_1 = x_2 \wedge y_1 = y_2),$$

$\mathcal{R}[QC(do(paint(x, y), s))]$ can be simplified to

$$\begin{aligned}(\forall x_1, x_2).\ \{&[(x = x_1 \wedge y = yellow) \vee \\ &\quad (color(x_1, yellow, s) \wedge \neg(\exists y')(x = x_1 \wedge y = y' \wedge y' \neq yellow))] \wedge \\ &[(x = x_2 \wedge y = yellow) \vee \\ &\quad (color(x_2, yellow, s) \wedge \neg(\exists y')(x = x_2 \wedge y = y' \wedge y' \neq yellow))]\} \supset \\ &x_1 = x_2.\end{aligned}$$

By using $\mathcal{D}_{qual}$, this can be further simplified to

$$\begin{aligned}(\forall x_1, x_2).\ [&(x = x_1 \wedge y = yellow \wedge color(x_2, yellow, s)) \vee \\ &(x = x_2 \wedge y = yellow \wedge color(x_1, yellow, s))] \supset \\ &x_1 = x_2,\end{aligned}$$

which is equivalent to

$$(\forall x_1).(color(x_1, yellow, s) \wedge y = yellow \supset x = x_1).$$

Therefore we can let $\Pi_{QC}(x, y, s)$ be the above formula, and obtain the following action precondition axiom which entails (8):

$$\begin{aligned}(\forall x, y, s).\ \ &Poss(paint(x, y), s) \equiv [nearby(x, s) \wedge haspaint(y, s) \wedge \\ &(\forall x_1).(color(x_1, yellow, s) \wedge y = yellow \supset x = x_1)].\end{aligned}$$

Notice that our procedure for computing action precondition axioms, and Theorem 8 above, explicitly relies on regression, and hence on the prior availability of a set of successor state axioms $\mathcal{D}_{ss}$. We assume that these have been computed using the methods of Section 5. This having been done, we apply the method of Section 6 to compile the qualification constraints. The resulting set of sentences

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{pre} \cup \mathcal{D}_{cls} \cup \mathcal{D}_{S_0}.$$

is what we take to be the monotonic theory "solving" the frame and ramification problems.

7 Examples of Qualification Constraints

Given a theory including a set of successor state axioms, the generation of action precondition axioms is straightforward using regression, as we have seen in the last section. This result, although simple, has many potential applications.

For every action, there may be some conditions for this action to be carried out physically. For example, in order to move a block, the block must be clear, and in order to turn off a light, the robot must be near the switch. As we observed earlier, this kind of precondition should be encoded directly as a necessary action precondition axiom. Still, there are cases when even such conditions are best formalized as state constraints. For example, in the block world, no block can be put on top of itself:[4]

$$\neg On(x, x, s).$$

Other kinds of preconditions are naturally encoded as qualification constraints. These restrict the behavior of a robot by putting constraints on what the desired states should be. For example, in robot motion planning, we don't want the robot running into obstacles:

$$obstacle(x) \supset \neg contact(x, s).$$

[4] We thank Vladimir Lifschitz for the example.

In the tower of Hanoi, it is against the rules of the game to have a bigger disc on top of a smaller one:

$$bigger(x, y) \supset \neg on(x, y, s).$$

In an employee-salary database, it may be the company's policy that an employee's salary cannot exceed her supervisor's:

$$sal(x, u, s) \wedge sal(y, v, s) \wedge supervisor(x, y, s) \supset u \geq v.$$

So far we have considered only state constraints represented by simple state formulas. These are what are called *static integrity constraints* in database theory. (See (Reiter [21]) for a situation calculus treatment of database updates.) It is particularly interesting that other kinds of formulas can also be used to constrain action preconditions. For example, the standard *dynamic integrity constraint* "an employee's salary must never decrease" can be represented as:

$$sal(x, u, s) \wedge sal(x, v, do(a, s)) \supset v \geq u.$$

Suppose we have the following successor state axiom for salary update, where $change(x, u)$ means change x's salary to u:

$$\begin{aligned} &Poss(a, s) \supset \{sal(x, u, do(a, s)) \equiv \\ &\quad [a = change(x, u) \vee (sal(x, u, s) \wedge \neg\exists v(a = change(x, v) \wedge u \neq v))]\}. \end{aligned}$$

Then by regressing the dynamic integrity constraint, we get a necessary condition for $Poss(change(x, u), s)$:

$$Poss(change(x, u), s) \supset [(\forall v).sal(x, v, s) \supset u \geq v].$$

Finally, some general remarks about the necessity of qualification constraints. If the purpose of a qualification constraint is to generate necessary conditions for $Poss$, one may ask, why not just supply such conditions directly? After all, one might argue, the intuition for deciding that a constraint is a qualification one is that the constraint should be enforced through action preconditions. There are several reasons why we don't want to list action preconditions explicitly:

1. This does not entirely eliminate the need for the constraints because we still have to check that the initial state satisfies the constraint (Corollary 9).
2. As Ginsberg and Smith argued in [5], there are both computational and epistemological reasons why listing action preconditions explicitly won't work.
3. Generating preconditions from state constraints results in a modular and generic theory.

To illustrate the last point, suppose we want to tell a robot that there is a crack on the floor of a room, and there is no way for him to cross it. We could examine the robot's knowledge base, find all actions that change his position, and add a precondition to all these actions that for them to be executable, the robot's destination must be on the same side of the crack as his starting position. But a much more uniform, modular and elegant way to do this is to tell the robot the following "dynamic constraint" which can be considered to be the "semantics" of the crack:

$$at(robot, x, s) \wedge at(robot, y, do(a, s)) \supset sameside(x, y),$$

and have the robot generate the correct preconditions by himself.

8 Other Kinds of State Constraints

We remarked earlier that there are state constraints which are different from both the ramification and the qualification ones. At the moment, we don't have a full story about them. However, there are some clear examples.

Certain control strategies seem to be best formalized as state constraints whose purpose is neither to compute the effects of actions, nor to generate action preconditions. For example, it is absolutely useless to repeat a state:

$$s \leq s' \wedge Equiv(s, do(a, s')) \supset Dont(a, s'),$$

where $Dont(a, s)$ means "don't do the action a in s", and $Equiv(s, s')$ means that s and s' are equivalent from the robot's point of view. For instance, if the only relevant fluent for the robot is $color(x, y, s)$, then we have

$$Equiv(s, s') \equiv (\forall x, y).[color(x, y, s) \equiv color(x, y, s')].$$

Notice that a difference between $Poss$ and $Dont$ is that we assume that the robot can't carry out an action unless $Poss$ is satisfied. However, even if $Dont$ is true, the robot can still carry out the action, although this may not be in his best interest. Therefore if a robot is unaware that painting a red block red again achieves nothing, there is nothing to prevent him from doing that.

As further examples of this kind of "control-oriented" state constraints, consider game playing. A player should avoid any move which will certainly result in a loss:

$$loss(do(a, s)) \supset Dont(a, s),$$

and as an example of two-step look ahead in a 2-player game:

$$[(\exists b).Poss(b, do(a, s)) \wedge loss(do(b, do(a, s)))] \supset Dont(a, s).$$

This kind of lookahead is analogous to alpha-beta pruning, minimax, etc.

9 Conclusions

9.1 What Have We Achieved?

We began by observing, as had Ginsberg and Smith before us, that there are at least two kinds of state constraints, one corresponding to ramifications, the other to qualifications. The former contribute indirect effects of actions; the latter are implicit axioms about $Poss$. We then proposed a two step procedure for determining an axiomatization which monotonically solves the ramification and qualification problems:

1. First determine, using the given effect axioms and ramification constraints, a set of successor state axioms. This is done by computing a set of new effect axioms using the old ones and the constraints, then completing the resulting set of effect axioms using the method of (Reiter [20]). There is a sufficient condition (Theorem 3) for determining when enough new effect axioms have been determined. At this point, the given ramification constraints are discarded in favour of the successor state axioms.

2. Secondly, by appealing to regression (using the successor state axioms computed in step 1) and the given qualification constraints, determine a set of action precondition axioms. At this point, the given qualification constraints are discarded in favour of the action precondition axioms.

We justified step 1 semantically by appealing to the minimization policy of (Lin and Shoham [11]). Step 2 we justified by simple Clark predicate completion.

The end result of this two step process is our proposed axiomatization for monotonically solving the ramification and qualification problems:

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{pre} \cup \mathcal{D}_{cls} \cup \mathcal{D}_{S_0}.$$

Here,

- Σ consists of foundational axioms for the situation calculus.
- $\mathcal{D}_{una}$ are unique names axioms for actions.
- $\mathcal{D}_{ss}$ are the successor state axioms computed in step 1.
- $\mathcal{D}_{pre}$ are the action precondition axioms computed in step 2.
- $\mathcal{D}_{cls}$ is a domain closure axiom for actions.
- $\mathcal{D}_{S_0}$ are initial state axioms which include the ramification and qualification constraints restricted to the initial state.

So far as we know, this is the first attempt in the literature to solve the ramification and qualification problems by providing an explicit, concise, and monotonic axiomatization.

9.2 What Haven't We Achieved?

In our two step approach to solving the ramification and qualification problems we computed successor state axioms first, ignoring the qualification constraints and the axioms giving explicit necessary conditions for *Poss*. Only in the second step were these axioms taken into account. This means that, semantically, we are applying a two step minimization policy. It is conceivable that there is a more appropriate uniform minimization policy, which would result in simultaneously computing successor state and action precondition axioms. Unfortunately, we have no idea at the moment how this can be correctly done. Our two step process is justified, informally, by our intuition that if performing an action leads to a state that violates a qualification constraint, then that action will be impossible to carry out in the real world. However, in order to know whether performing an action will lead to a violation of the constraint, we need relevant knowledge about the effects of the action, which is to say, we need the frame axioms. Technically, this means that we should compute *Poss* from state constraints by using regression, which relies on the prior availability of a set of successor state axioms.

Another problematic aspect of our solution is that we did not provide a syntactic way of distinguishing ramification from qualification constraints. This is a perplexing and unsatisfying situation made even worse by the fact that conjoining a ramification

constraint and a qualification constraint leads to a syntactically acceptable constraint of neither type.

Finally, we observed that there are other kinds of state constraints distinct from ramifications and qualifications; these appear to be control-oriented. Our intuition is that they are intimately related to various deontic notions ([14, 15]), but we have yet to explore this idea in depth.

Acknowledgements

We wish to thank Vladimir Lifschitz, Javier Pinto, and Richard Scherl for helpful discussions, and comments on earlier drafts of the paper. The research was supported in part by the Government of Canada Institute for Robotics and Intelligent Systems, and in part by the National Science and Engineering Research Council of Canada.

Appendix Proofs

Proposition 1

$$\begin{aligned}
\textit{Transitivity:} &\quad (s_1 < s_2 \wedge s_2 < s_3) \supset s_1 < s_3 \\
\textit{Anti-reflexivity:} &\quad \neg s < s \\
\textit{Unique names:} &\quad s_1 < s_2 \supset s_1 \neq s_2 \\
\textit{Induction on} <: &\quad (\forall P)[P(S_0) \wedge (\forall a, s)(P(s) \wedge Poss(a, s) \supset P(do(a, s))) \supset \\
&\qquad (\forall s)(S_0 \leq s \supset P(s))]
\end{aligned}$$

Proof *Transitivity.* We apply induction on s_3. If s_3 is S_0, then the sentence is entailed by axiom (12). Inductively, assume that $s_1 < s_2 \wedge s_2 < s_3 \supset s_1 < s_3$. We prove that $s_1 < s_2 \wedge s_2 < do(a, s_3) \supset s_1 < do(a, s_3)$. Suppose that $s_1 < s_2 \wedge s_2 < do(a, s_3)$. We prove that $s_1 < do(a, s_3)$. By axiom (13), $s_2 < do(a, s_3)$ is equivalent to $Poss(a, s_3) \wedge s_2 \leq s_3$. Thus we have $s_2 < s_3 \vee s_2 = s_3$. By the inductive assumption, the first disjunct and $s_1 < s_2$ (assumed) implies $s_1 < do(a, s_3)$. The second disjunct and $s_1 < s_2$ implies $s_1 < s_3$, and together with $Poss(a, s_3)$, implies $s_1 < do(a, s_3)$ as well. Therefore, by induction axiom (11), the transitivity axiom holds.

Anti-reflexivity. Use induction on s, applying transitivity and axiom (13) in the inductive step.

Unique names. Follows from anti-reflexivity by rewriting it as $s_1 = s_2 \supset \neg s_1 < s_2$.

Induction on $<$. Let $P'(s)$ stand for $S_0 \leq s \supset P(s)$; instantiate induction axiom (11) on P'.

Before proving Theorem 3, we prove Lemma 5.

Lemma 5 *For any ramification state constraint* $RC(s)$,

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \{RC(S_0)\} \models (\forall s).S_0 \leq s \supset RC(s)$$

iff

$$\mathcal{D}_{una} \models (\forall s, a).RC(s) \wedge Poss(a, s) \supset \mathcal{R}[RC(do(a, s))].$$

Proof The "if" part is straightforward by induction (Proposition 1). We prove the "only if" part. Suppose

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \{RC(S_0)\} \models (\forall s).S_0 \leq s \supset RC(s). \tag{23}$$

We prove that

$$\mathcal{D}_{una} \models (\forall s, a).RC(s) \wedge Poss(a, s) \supset \mathcal{R}[RC(do(a, s))].$$

Suppose M is a model of $\mathcal{D}_{una}$. Suppose σ_s and σ_a are variable assignments for state and action variables, respectively, such that $M, \sigma_s, \sigma_a \models RC(s) \wedge Poss(a, s)$. We show that $M, \sigma_s, \sigma_a \models \mathcal{R}[RC(do(a, s))]$. To that end, construct M' such that

1. M' and M share the same domains for sort *action* and for sort *object*.
2. M' interprets every state independent predicate and every state independent function the same as M.
3. $M' \models \Sigma$.
4. For every variable assignment for object variables σ_o, and for every fluent $F(\vec{x}, s)$, $M', \sigma_o \models F(\vec{x}, S_0)$ iff $M, \sigma_s, \sigma_o \models F(\vec{x}, s)$.
5. $M' \models (\forall a, s)Poss(a, s)$.
6. $M' \models \mathcal{D}_{ss}$.

Clearly,

$$M', \sigma_a \models \Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \{RC(S_0)\} \cup \{Poss(a, S_0)\}.$$

Therefore by our assumption (23), $M', \sigma_a \models RC(do(a, S_0))$. Now since $M' \models \mathcal{D}_{ss}$ and $M', \sigma_a \models Poss(a, S_0)$, we have that $M', \sigma_a \models RC(do(a, S_0))$ iff $M', \sigma_a \models \mathcal{R}[RC(do(a, S_0))]$. But $\mathcal{R}[RC(do(a, S_0))]$ is the same as the result of replacing s in $\mathcal{R}[RC(do(a, s))]$ by S_0, thus by our construction

$$M', \sigma_a \models \mathcal{R}[RC(do(a, S_0))] \text{ iff } M, \sigma_s, \sigma_a \models \mathcal{R}[RC(do(a, s))].$$

Therefore $M, \sigma_s, \sigma_a \models \mathcal{R}[RC(do(a, s))]$. This proves the "only if" part, thus the lemma.

Theorem 3 *Let $\mathcal{D}_{ss}$ be a set of the successor state axioms, one for each fluent, of the form:*

$$Poss(a, s) \supset F(x_1, ..., x_n, do(a, s)) \equiv [\Psi_F \vee (F(x_1, ..., x_n, s) \wedge \neg\Psi_{\neg F})]. \tag{24}$$

where $F(x_1, ..., x_n, s)$ is an (n+1)-ary fluent, and Ψ_F and $\Psi_{\neg F}$ satisfy (15) and (16), respectively. Suppose further that:

1. *The following* consistency condition *(Reiter [20]) holds for each F:*

$$\mathcal{D}_{una} \models (\forall s, a, x_1, ..., x_n)\neg(\Psi_F \wedge \Psi_{\neg F}). \tag{25}$$

2. *The ramification constraints relativised to states accessible from S_0 are derivable from the successor state axioms:*

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \{RC(S_0)\} \models (\forall s).S_0 \leq s \supset RC(s) \tag{26}$$

for every $(\forall s)RC(s) \in \mathcal{D}_{ram}$.

Then a first-order structure is a minimal model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ram} \cup \mathcal{D}_{ef}$$

iff it is a model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}^{-}_{ram} \cup \mathcal{D}^{S_0}_{ram},$$

where $\mathcal{D}^{-}_{ram}$ is the set of ramification constraints relativised to states which are not accessible from S_0:

$$\mathcal{D}^{-}_{ram} = \{(\forall s).S_0 \not\leq s \supset RC(s) \mid (\forall s)RC(s) \in \mathcal{D}_{ram}\},$$

and $\mathcal{D}^{S_0}_{ram}$ is the set of ramification constraints restricted to the initial state:

$$\mathcal{D}^{S_0}_{ram} = \{RC(S_0) \mid (\forall s)RC(s) \in \mathcal{D}_{ram}\}.$$

Proof Let M be a model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}^{-}_{ram} \cup \mathcal{D}^{S_0}_{ram}. \tag{27}$$

We show that M is a minimal model of

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ef} \cup \mathcal{D}_{ram}. \tag{28}$$

We first show that M is a model of the theory. Consider any fluent F. Suppose

$$(\forall a, s, \vec{x}).Poss(a, s) \supset \Psi^{+} \supset F(do(a, s))$$

is in $\mathcal{D}_{ef}$. Recall that by our construction, Ψ_F is equivalent to a formula of the form $\varphi_1 \vee \cdots \vee \varphi_n$ such that for some i, φ_i is Ψ^{+}. Thus

$$\Sigma \cup \mathcal{D}_{una} \models (\forall a, s, \vec{x}).\Psi^{+} \supset \Psi_F.$$

Therefore

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \models (\forall a, s, \vec{x}).Poss(a, s) \supset \Psi^{+} \supset F(do(a, s)).$$

Similarly, by using in addition the consistency condition (25), we can show that for any fluent F, if

$$(\forall a, s, \vec{x}).Poss(a, s) \supset \Psi^{-} \supset \neg F(do(a, s))$$

is in $\mathcal{D}_{ef}$, then

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \models (\forall a, s, \vec{x}).Poss(a, s) \supset \Psi^{-} \supset \neg F(do(a, s)).$$

Thus

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \models \mathcal{D}_{ef}.$$

Now by (26), we have

$$\Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ram}^{-} \cup \mathcal{D}_{ram}^{S_0} \models \mathcal{D}_{ram}.$$

Therefore, (27) implies (28). By the assumption, M is a model of (28), so it is also a model of (28). Suppose that M' is a model of (28), and σ_s a variable assignment for situations such that

1. M and M' have the same universe.
2. M and M' differ only in their interpretations of fluents.
3. For any assignment σ_o of object variables, and any fluent $F(x_1, ..., x_n, s)$,

$$M, \sigma_s, \sigma_o \models F(x_1, ..., x_n, s)$$

iff

$$M', \sigma_s, \sigma_o \models F(x_1, ..., x_n, s).$$

Now for any assignments σ_a and σ_o of action and object variables, respectively, and any fluent $F(x_1, ..., x_n, s)$, suppose

$$M, \sigma_s, \sigma_a, \sigma_o \models Poss(a, s) \wedge ab(a, F(x_1, ..., x_n, s)).$$

Then since M is a model of $\mathcal{D}_{ss}$, either

$$M, \sigma_s, \sigma_a, \sigma_o \models \neg F \wedge \Psi_F$$

or

$$M, \sigma_s, \sigma_a, \sigma_o \models F \wedge \neg \Psi_F \wedge \Psi_{\neg F}.$$

Either way, since Ψ_F and $\Psi_{\neg F}$ satisfy (15) and (16), respectively, and M' is a model of (28) and so a model of the left hand side of (15) and (16), we conclude that

$$M', \sigma_s, \sigma_a, \sigma_o \models ab(a, F(x_1, ..., x_n, s)).$$

From our definition of minimal models in Section 4, we conclude that M is a minimal model of (28).

Let M be a minimal model of (28). We show that it is a model of (27). To that end, construct a structure M' as follows:

1. Let the universe of M' be the same as that of M. Let M' and M interpret everything the same except for the truth values of fluents.

2. Let M' and M interpret the fluents the same in S_0, i.e. for every assignment σ_o and every fluent $F(x_1, ..., x_n, s)$,

$$M', \sigma_o \models F(x_1, ..., x_n, S_0)$$

iff

$$M, \sigma_o \models F(x_1, ..., x_n, S_0).$$

3. Inductively, for any variable assignment σ, if $M, \sigma \models \neg Poss(a, s)$, then for every fluent $F(x_1, ..., x_n)$,

$$M', \sigma \models F(x_1, ..., x_n, do(a, s))$$

iff

$$M, \sigma \models F(x_1, ..., x_n, do(a, s)).$$

If $M, \sigma \models Poss(a, s)$, then for every fluent $F(x_1, ..., x_n, s)$, and for every assignment σ', if $\sigma'(s) = \sigma(s)$, and $\sigma'(a) = \sigma(a)$, then

$$M', \sigma' \models F(x_1, ..., x_n, do(a, s))$$

iff

$$M', \sigma' \models \Psi_F \vee (F(x_1, ..., x_n, s) \wedge \neg \Psi_{\neg F}),$$

where Ψ_F and $\Psi_{\neg F}$ are as in (24).

Clearly, M' is well-defined. We claim that it is a model of (27). It is clear that we only need to show that it is a model of $\mathcal{D}^-_{ram}$. In fact, we claim that it is a model of $\mathcal{D}_{ram}$. This we prove by induction. Let $(\forall s)RC(s) \in \mathcal{D}_{ram}$. By construction, M' agrees with M on S_0, so it satisfies $RC(S_0)$. Let σ_s be an assignment of situation variables. We assume inductively that $M', \sigma_s \models RC(s)$; we show that for any assignment σ_a of action variables,

$$M', \sigma_s, \sigma_a \models RC(do(a, s)).$$

There are two cases. If $M', \sigma_s, \sigma_a \models \neg Poss(a, s)$, then by construction, M' agrees with M on $do(a, s)$, and so satisfies $RC(do(a, s))$. Now suppose $M', \sigma_s, \sigma_a \models Poss(a, s)$. By condition (26) and Lemma 5,

$$\mathcal{D}_{una} \models (\forall a, s).RC(s) \wedge Poss(a, s) \supset \mathcal{R}[RC(do(a, s))].$$

Therefore $M', \sigma_s, \sigma_a \models \mathcal{R}[RC(do(a, s))]$. But $M', \sigma_s, \sigma_a \models \{Poss(a, s)\} \cup \mathcal{D}_{ss}$, thus $M', \sigma_s, \sigma_a \models RC(do(a, s))$. So by induction, $M' \models (\forall s)RC(s)$. Therefore $M' \models \mathcal{D}_{ram}$.

We now show that $M = M'$. This we do by induction. By the construction of M', they agree on S_0. Let σ_s be an assignment of situation variables. Inductively, we assume that M and M' agree on s:

$$M', \sigma_s, \sigma_o \models F(x_1, ..., x_n, s)$$

iff

$$M, \sigma_s, \sigma_o \models F(x_1, ..., x_n, s),$$

for every assignment σ_o of object variables. We show that

$$M', \sigma_s, \sigma_a, \sigma_o \models F(x_1, ..., x_n, do(a, s))$$

iff

$$M, \sigma_s, \sigma_a, \sigma_o \models F(x_1, ..., x_n, do(a, s)),$$

for every assignments σ_a and σ_o of action and object variables, respectively. There are two cases. If $M, \sigma_s, \sigma_a, \sigma_o \models \neg Poss(a, s)$, then this follows from the construction of M'. Now assume that

$$M, \sigma_s, \sigma_a, \sigma_o \models Poss(a, s).$$

Since M' is a model of (27), as we have proved above in the first part of of our proof, if

$$M', \sigma_s, \sigma_a, \sigma_o \models ab(a, F(x_1, ..., x_n, s))$$

then

$$M, \sigma_s, \sigma_a, \sigma_o \models ab(a, F(x_1, ..., x_n, s))$$

as well. But M is minimal, so

$$M', \sigma_s, \sigma_a, \sigma_o \models ab(a, F(x_1, ..., x_n, s))$$

iff

$$M, \sigma_s, \sigma_a, \sigma_o \models ab(a, F(x_1, ..., x_n, s)).$$

But M and M' agree on s, so they also agree on $do(a, s)$. By induction, we have $M = M'$. Therefore M is a model of (27).

References

[1] Andrew B. Baker. Nonmonotonic reasoning in the framework of the situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[2] Keith L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logics and Databases*, pages 293–322. Plenum Press, New York, 1978.

[3] Tom Costello. Solutions to the ramification problem. In *Working Papers for 2nd Symposium on Logical Formalizations of Commonsense Reasoning*, pages 32–39, Austin, Texas, 1993.

[4] Jeff Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 1986.

[5] Matthew L. Ginsberg and David E. Smith. Reasoning about action II: the qualification problem. *Artificial Intelligence*, 35:311–342, 1988.

[6] Cordell C. Green. Application of theorem proving to problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI–69)*, pages 219–239, 1969.

[7] G. Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *Working Papers for 2nd Symposium on Logical Formalizations of Commonsense Reasoning*, pages 85–93, Austin, Texas, 1993.

[8] Vladimir Lifschitz. Formal theories of action. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence (IJCAI–87)*, pages 966–972, 1987.

[9] Vladimir Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[10] Vladimir Lifschitz. Restricted monotonicity. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI–93)*, 1993.

[11] Fangzhen Lin and Yoav Shoham. Provably correct theories of action: Preliminary report. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI–91)*, Anaheim, CA, 1991.

[12] John McCarthy. Epistemological problems of Artificial Intelligence. In *IJCAI-77*, pages 1038–1044. Cambridge, MA, 1977.

[13] John McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–118, 1986.

[14] John-Jules Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame Journal of Formal Logic*, 29:109–136, 1988.

[15] John-Jules Ch. Meyer, Hans Weigand, and Roel Wieringa. A specification language for static, dynamic and deontic integrity constraints. In J. Demetrovics and B. Thalheim, editors, *Proceedings of 2nd Symposium on Mathematical Fundamentals of Data-base systems*, pages 347–366, 1989. LNCS 364.

[16] Edwin P.D. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4:356–372, 1988.

[17] Javier Pinto. *Temporal Reasoning in the Situation Calculus*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, February 1994.

[18] Javier Pinto and Raymond Reiter. Extending the situation calculus with event occurrences. In *Second Symposium on Logical Formalizations of Commonsense Reasoning*, 1993.

[19] Raymond Reiter. A simple solution to the frame problem (sometimes). Technical report, Department of Computer Science, University of Toronto. In preparation.

[20] Raymond Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.

[21] Raymond Reiter. On specifying database updates. Technical report, Department of Computer Science, University of Toronto, 1992. KRR-TR-92-3.

[22] Raymond Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[23] Len K. Schubert. Monotonic solution to the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H.E. Kyberg, R.P. Loui, and G.N. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer Academic Press, Boston, MA, 1990.

[24] Steven Shapiro. *First-Order Solutions to the Frame Problem.* M.Sc. thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1993.

[25] Richard Waldinger. Achieving several goals simultaneously. In E. Elcock and D. Michie, editors, *Machine Intelligence*, pages 94–136. Ellis Horwood, Edinburgh, Scotland, 1977.