Proving Properties of States in the Situation Calculus

Raymond Reiter
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
and
The Canadian Institute for Advanced Research
email: reiter@ai.toronto.edu

Abstract

In the situation calculus, it is sometimes necessary to prove that certain properties are true in all world states accessible from the initial state. This is the case for some forms of reasoning about the physical world, for certain planning applications, and for verifying integrity constraints in databases. Not surprisingly, this requires a suitable form of mathematical induction. This paper motivates the need for proving properties of states in the situation calculus, proposes appropriate induction principles for this task, and gives examples of their use in databases and for reasoning about the physical world.

Abbreviated title: Proving Properties of States

1 Introduction

The situation calculus [8] is enjoying new popularity these days. One reason is that its expressiveness is considerably richer than has been commonly believed (Gelfond, Lifschitz and Rabinov [2], Pinto and Reiter [10], Schubert [16]). Another is the possibility of precisely characterizing the strengths and limitations of various general theories of actions expressed within its formalism (see Lin and Shoham [7], Lifschitz [5] and Reiter [12] for examples). Still another concerns its application in formalizing update transactions in database theory (Reiter [12, 15]).

Within the situation calculus, it is sometimes necessary to prove that certain properties are true in all world states accessible from the initial state. This is the case in reasoning about the physical world, where, for example, one might want to infer that if an object is broken, and if it never gets repaired, then it will always be broken. The same kinds of proofs are required in planning to show that no plan exists to achieve a certain world state. A closely related need for proving properties of world states arises in database theory, in connection with the verification of integrity constraints. An integrity constraint for a database specifies what counts as a legal database state; it is a property that every state must satisfy, for example, the property that no one's salary may decrease during the evolution of a personnel database. Verifying that a database satisfies a particular integrity constraint amounts to proving that the constraint is an entailment of the database. Not surprisingly, such proofs require a suitable form of mathematical induction.

This paper motivates the need for proving properties of states in the situation calculus, proposes appropriate induction principles for this task, and gives examples of their use in databases and in reasoning about the physical world.

2 Why Prove Properties of World States?

The kinds of properties we have in mind are of the form "No matter how the world evolves from its initial state, such-and-such will be true." There are at least three reasons for wanting to prove such properties, one having to do with reasoning about the physical world, another with certain planning applications, and a third with verifying integrity constraints in databases.

2.1 Reasoning about the Physical World

Many facts about the world concern properties which must hold no matter how the world unfolds, for example, if an object is broken and it never gets repaired, then it will always be broken.

$$\begin{aligned}(\forall x).broken(x, S_0) \wedge [(\forall s).S_0 \leq s \supset \neg occurs(repair(x), s)] \supset \\ (\forall s').S_0 \leq s' \supset broken(x, s').\end{aligned} \tag{1}$$

Here,

- S_0 denotes the initial state of the world.
- $s \leq s'$ means that s' is a possible future of s, i.e. that it is possible to reach state s' via some sequence of actions, beginning with state s.[1]
- $occurs(a, s)$ means that a is one of the actions in the sequence of actions leading from S_0 to s.

One concern of this paper is how an agent might derive such facts from some background situation calculus axiomatization.

2.2 Planning

The standard logical account of planning views this as a theorem proving task (Green [3]): To obtain a plan whose execution will lead to a world state s in which the goal $G(s)$ will be true, establish that

$$Axioms \models (\exists s).S_0 \leq s \wedge G(s).$$

Sometimes we would like to establish that no plan could possibly lead to a given world state. This is the problem of establishing that

$$Axioms \models (\forall s).S_0 \leq s \supset \neg G(s),$$

i.e. that in all possible future world states, G will be false.[2]

[1]See Section 3 below for an axiomatization of $\leq$.

[2]I am grateful to Fangzhen Lin for pointing out this use, in planning, for proofs that certain properties must be true in all world states.

2.3 Integrity Constraints in Database Theory

In the theory of databases, an *integrity constraint* specifies what counts as a legal database state; it is a property that every database state must satisfy (Reiter [11]). For example,

- Salaries are functional: No one may have two different salaries in the same database state.
- No one's salary may decrease during the evolution of the database.

The concept of an integrity constraint is intimately connected with that of database *evolution*; no matter how the database evolves, the constraint must be true in all database futures. It follows that in order to make formal sense of integrity constraints, we need a prior theory of database evolution. How do databases change? One way is via predefined update *transactions*, e.g.

- Change a person's salary to $.
- Register a student in a course.

We shall assume that such transactions provide the only mechanism for database state changes.

Following Reiter [14, 15], we propose that databases be represented in the situation calculus; updatable relations will be fluents, i.e. they will take a state argument. Moreover, update transactions will be treated exactly like actions in the AI planning domain, so that transactions will be functions. The next section illustrates this approach to the representation of databases and their transactions within the situation calculus.

2.3.1 The Basic Approach to Specifying Database Transactions: An Example

We consider a toy education database to illustrate our approach to specifying update transactions.

Relations

The database involves the following three relations:

1. $enrolled(st, course, s)$: Student st is enrolled in course $course$ when the database is in state s.

2. $grade(st, course, grade, s)$: The grade of student st in course $course$ is $grade$ when the database is in state s.

3. $prerequ(pre, course)$: pre is a prerequisite course for course $course$. Notice that this relation is state independent, so is not expected to change during the evolution of the database.

Initial Database State

We assume given some first order specification of what is true of the initial state S_0 of the database. These will be arbitrary first order sentences, the only restriction being that those predicates which mention a state, mention only the initial state S_0. Incomplete initial database states are permitted. Examples of information which might be true in the initial state are:

$$enrolled(Sue, C100, S_0) \vee enrolled(Sue, C200, S_0),$$

$$(\exists c) enrolled(Bill, c, S_0),$$
$$(\forall p).prerequ(p, P300) \equiv p = P100 \vee p = M100,$$
$$(\forall p) \neg prerequ(p, C100),$$
$$(\forall c).enrolled(Bill, c, S_0) \equiv c = M100 \vee c = C100 \vee c = P200,$$
$$enrolled(Mary, C100, S_0), \quad \neg enrolled(John, M200, S_0), \ldots$$
$$grade(Sue, P300, 75, S_0), \quad grade(Bill, M200, 70, S_0), \ldots$$
$$prerequ(M200, M100), \quad \neg prerequ(M100, C100), \ldots$$

Database Transactions

Update transactions will be denoted by function symbols, and will be treated exactly as are actions in the situation calculus. For the example, there will be three transactions:

1. $register(st, course)$: Register student st in course $course$.

2. $change(st, course, grade)$: Change the current grade of student st in course $course$ to $grade$.

3. $drop(st, course)$: Student st drops course $course$.

Transaction Preconditions

Normally, transactions have preconditions which must be satisfied by the current database state before the transaction can be "executed". In the example, it will be possible to register a student in a course iff she has obtained a grade of at least 50 in all prerequisites for the course:

$$Poss(register(st,c),s) \equiv \{(\forall p).prerequ(p,c) \supset (\exists g).grade(st,p,g,s) \land g \geq 50\}.$$

It is possible to change a student's grade iff he has a grade which is different than the new grade:

$$Poss(change(st,c,g),s) \equiv (\exists g').grade(st,c,g',s) \land g' \neq g.$$

A student may drop a course iff the student is currently enrolled in that course:

$$Poss(drop(st,c),s) \equiv enrolled(st,c,s).$$

Update Specifications

The following axioms specify the effects of the transactions on the database relations:

$$Poss(register(st,c),s) \supset enrolled(st,c,do(register(st,c),s)).$$

$$Poss(drop(st,c),s) \supset \neg enrolled(st,c,do(drop(st,c),s)).$$

$$Poss(change(st,c,g),s) \supset grade(st,c,g,do(change(st,c,g),s)).$$

As is well known in the planning literature (McCarthy and Hayes [9]), the above axioms are not sufficient to correctly specify all database futures; *frame axioms*, specifying those relations left invariant by each transaction, must also be provided. We defer proposing such frame axioms until Section 4.1; in the mean time, assume they have also been provided in conjunction with the above update specification axioms.

2.3.2 Integrity Constraints Revisited

Recall that an integrity constraint specifies what counts as a legal database state; it is a property that every database state must satisfy. Accordingly, it is natural to represent these as first order sentences, universally quantified over states. For example, no one may have two different grades for the same course in any database state:

$$(\forall s)(\forall st, c, g, g').S_0 \leq s \wedge grade(st, c, g, s) \wedge grade(st, c, g', s) \\ \supset g = g'. \tag{2}$$

In a personnel database, we might require that salaries must never decrease during the evolution of the database:

$$(\forall s, s')(\forall p, \$, \$').S_0 \leq s \wedge s \leq s' \wedge sal(p, \$, s) \wedge sal(p, \$', s') \\ \supset \$ \leq \$'. \tag{3}$$

These intuitions lead to:

Definition: Constraint Satisfaction

A database *satisfies* an integrity constraint IC iff the database entails the constraint:

$$Database \models IC.$$[3]

As it happens, *state constraints*, as they arise in the context of the *ramification problem* (Finger [1], Lin and Shoham [7]), have exactly the same character as database integrity constraints, and can be given an identical treatment. See Lin and Reiter [6] for details. Within the database community, the notion of integrity constraints as inductive entailments has been proposed, and implemented, by Sheard and Stemple [17] in the context of relational databases.

2.4 Summary

There are at least three reasons for wanting to prove properties of states in the situation calculus, one having to do with reasoning about the physical world, another with showing that no plan exists for certain planning problems, and the third with verifying integrity constraints in databases. In general, we shall assume given some situation calculus axiomatization, with a distinguished *initial state* S_0. Using this axiomatization, our objective is to prove properties of all states accessible (via some finite sequence of possible actions) from S_0,

[3]This definition should be contrasted with those in Reiter [11, 13]. It seems that there is not a unitary concept of integrity constraint in database theory, and that there are many subtleties involved.

for example, sentences like (1), (2) and (3). Notice that these sentences are universally quantified over states, in contrast to the standard theorem-proving account of plan synthesis (Green [3]), which requires proofs of sentences existentially quantified over states.

Not too surprisingly, proving such universally quantified sentences requires mathematical induction. Just what form these induction principles should take is the subject of the next section.

3 Formulating Suitable Induction Axioms

There is a close analogy between the situation calculus and the theory of the natural numbers; simply identify S_0 with the natural number 0, and $do(Add1, s)$ with the successor of the natural number s. In effect, an axiomatization in the situation calculus is a theory in which each "natural number" s has arbitrarily many successors.[4] Just as an induction axiom is necessary to prove anything interesting about the natural numbers, so also is induction required to prove general properties of states. This section is devoted to formulating some induction principles suitable for this task.

We shall use a many-sorted language for the situation calculus. The two domain independent sorts are *state* and *action*. There is a unique state constant symbol, S_0, denoting the initial state, and a binary state function symbol *do* with arguments of sort *state* and *action*, respectively. Throughout, variables s and a, with, or without subscripts and superscripts, will be of sort *state* and *action*, respectively.

We begin by postulating a second order induction axiom:

$$(\forall P).P(S_0) \wedge (\forall a, s)[P(s) \supset P(do(a, s))] \supset (\forall s)P(s). \tag{4}$$

Compare this with the induction axiom for the natural numbers:

$$(\forall P).P(0) \wedge (\forall x)[P(x) \supset P(succ(x))] \supset (\forall x)P(x).$$

Just as the induction axiom for the natural numbers restricts the domain of numbers to 0 and its successors, the effect of the induction axiom (4) is to restrict the state domain of any of its models to be isomorphic to the smallest set $\mathcal{S}$ satisfying:

[4]There could even be infinitely many successors whenever an action is parameterized by a real number, as for example *move*(*block*, *location*).

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.

Not every action is executable in every state. Accordingly, we introduce a binary predicate $Poss(a, s)$, meaning that it is possible to execute action a in state s. Next, we define an ordering relation $<$ on states. The intended interpretation of $s < s'$ is that state s' is reachable from state s by some sequence of one or more actions, each action of which is possible in that state resulting from executing the actions preceding it in the sequence.

$$(\forall s) \neg s < S_0. \tag{5}$$

$$(\forall a, s, s').s < do(a, s') \equiv Poss(a, s') \wedge s \leq s'. \tag{6}$$

Here, $s \leq s'$ is an abbreviation for $s < s' \vee s = s'$.

In the sequel, we shall refer to the above three axioms (4), (5) and (6) as the *foundational axioms.*

Proposition 1 *The following sentence is entailed by the foundational axioms:*

$$\begin{aligned}(\forall s, s').s < s' \equiv (\forall P).\{[(\forall a, s_1).Poss(a, s_1) \supset P(s_1, do(a, s_1))] \wedge \\ [(\forall a, s_1, s_2).Poss(a, s_2) \wedge P(s_1, s_2) \supset P(s_1, do(a, s_2))]\} \\ \supset P(s, s').\end{aligned} \tag{7}$$

Proof:
($\Leftarrow$) Take P to be $\leq$ in (7) and use (6).
($\Rightarrow$) Do induction (axiom (4)) on s' in the formula

$$\begin{aligned}(\forall s).s < s' \supset (\forall P).\{[(\forall a, s_1).Poss(a, s_1) \supset P(s_1, do(a, s_1))] \wedge \\ [(\forall a, s_1, s_2).Poss(a, s_2) \wedge P(s_1, s_2) \supset P(s_1, do(a, s_2))]\} \\ \supset P(s, s').\end{aligned}$$

Ξ

Proposition 1 informs us that $\leq$ is the smallest binary relation on states such that:

1. $\sigma \leq \sigma$, and
2. $\sigma \leq do(a, \sigma')$ whenever action a is possible in state σ' and $\sigma \leq \sigma'$.

Theorem 1 *The following sentence is entailed by the foundational axioms:*

$$(\forall W).W(S_0) \wedge [(\forall a,s).Poss(a,s) \wedge S_0 \leq s \wedge W(s) \supset W(do(a,s))] \\ \supset (\forall s).S_0 \leq s \supset W(s). \qquad (IP_{S_0 \leq s})$$

Proof:
Let W be a unary predicate variable. Using (6) and the instance $W(s) \supset s \leq s' \wedge W(s')$ for $P(s,s')$ in (7), we can derive the following second order sentence:

$$(\forall W)(\forall s).\{W(s) \wedge [(\forall a,s_1).W(s_1) \wedge s \leq s_1 \wedge Poss(a,s_1) \supset W(do(a,s_1))]\} \\ \supset (\forall s').s \leq s' \supset W(s').$$

The theorem is established by taking the instance S_0 of s in this.

Ξ

Sentence $(IP_{S_0 \leq s})$ provides an induction principle suitable for proving properties of states s when $S_0 \leq s$.

Frequently, we shall want to prove sentences of the form

$$(\forall s,s').S_0 \leq s \wedge s \leq s' \supset R(s,s').$$

The integrity constraint (3) is an example. Towards that end, we now derive a suitable induction principle.

Theorem 2 *The following sentence is entailed by the foundational axioms:*

$$(\forall R).R(S_0,S_0) \wedge \\ [(\forall a,s).Poss(a,s) \wedge S_0 \leq s \wedge R(s,s) \supset R(do(a,s),do(a,s))] \wedge \\ [(\forall a,s,s').Poss(a,s') \wedge S_0 \leq s \wedge s \leq s' \wedge R(s,s') \supset R(s,do(a,s'))] \\ \supset (\forall s,s').S_0 \leq s \wedge s \leq s' \supset R(s,s'). \qquad (IP_{S_0 \leq s \leq s'})$$

Proof: Using (7), we can derive:

$$(\forall s,s').s \leq s' \supset \\ (\forall P).\{P(s,s) \wedge [(\forall a,s_1).Poss(a,s_1) \supset P(s_1,do(a,s_1))] \wedge \\ [(\forall a,s_1,s_2).Poss(a,s_2) \wedge P(s_1,s_2) \supset P(s_1,do(a,s_2))]\} \\ \supset P(s,s').$$

Writing $S_0 \leq s \supset s \leq s' \wedge R(s,s')$ for $P(s,s')$ in this, we obtain, with the help of (6):

$$\begin{array}{l}
(\forall s,s').s \leq s' \supset \\
\quad (\forall R).\{S_0 \leq s \wedge R(s,s) \wedge \\
\qquad [(\forall a,s_1).Poss(a,s_1) \wedge S_0 \leq s_1 \supset R(s_1, do(a,s_1))] \wedge \\
\qquad [(\forall a,s_1,s_2).Poss(a,s_2) \wedge S_0 \leq s_1 \wedge s_1 \leq s_2 \wedge R(s_1,s_2) \supset R(s_1, do(a,s_2))] \\
\qquad\quad \supset R(s,s')\}.
\end{array} \tag{8}$$

Using $(IP_{S_0 \leq s})$ with $R(s,s)$ for $W(s)$, we obtain:

$$\begin{array}{l}
(\forall s).S_0 \leq s \supset \\
\quad (\forall R).R(S_0,S_0) \wedge \\
\quad [(\forall a,s_1).Poss(a,s_1) \wedge S_0 \leq s_1 \wedge R(s_1,s_1) \supset R(do(a,s_1), do(a,s_1))] \\
\qquad \supset R(s,s).
\end{array}$$

This, together with (8) entails the following:

$$\begin{array}{l}
(\forall R).R(S_0,S_0) \wedge \\
\quad [(\forall a,s).Poss(a,s) \wedge S_0 \leq s \supset R(s, do(a,s))] \wedge \\
\quad [(\forall a,s).Poss(a,s) \wedge S_0 \leq s \wedge R(s,s) \supset R(do(a,s), do(a,s))] \wedge \\
\quad [(\forall a,s,s').Poss(a,s') \wedge S_0 \leq s \wedge s \leq s' \wedge R(s,s') \supset R(s, do(a,s'))] \\
\qquad\quad \supset (\forall s,s').S_0 \leq s \wedge s \leq s' \supset R(s,s').
\end{array}$$

This can be simplified by observing that by the simple induction principle $(IP_{S_0 \leq s})$, the first and third conjuncts of the antecedent of this implication entail the second conjunct. This establishes the theorem.

Ξ

Sentence $(IP_{S_0 \leq s \leq s'})$ provides an induction principle suitable for proving properties of pairs of states s and s' when $S_0 \leq s \wedge s \leq s'$.

4 Using these Axioms in Practice

In this section, we give some examples of the use of these induction principles to prove properties of world states. Before doing so, we must revisit the frame problem, which we glossed over in Section 2.3.1.

4.1 Frame Axioms and the Explanation Closure Axioms of Haas and Schubert

As we all know, to correctly formalize dynamically changing worlds, some axiomatization is required which provides the same effects as frame axioms. At one extreme, the frame axioms themselves may be used. At the other, the frame axioms may be implicitly represented, as entailments of some uniform nonmonotonic policy, as for example in (Lin and Shoham [7]) or (Lifschitz [5]). For the purposes of invoking the induction principles of Section 3 to prove properties of world states, it is irrelevant how the frame axioms are represented, so long as they are available, whether explicitly or implicitly. Nevertheless, for purposes of actually constructing inductive proofs, it is desirable to have an explicit collection of axioms with the same force as frame axioms. One such approach, which we favour for the purposes of this paper, appeals to so-called *explanation closure axioms* as defined by Schubert [16], in elaborating on a proposal of Haas [4].

To illustrate the Haas-Schubert proposal, consider the transaction of registering a student in a course with multiple sections, without specifying in which section the student is placed. Denote the transaction by $register(st, c)$ and consider the relation $enrolled(st, c, sec, s)$ meaning that st is enrolled in section sec of course c when the database is in state s. We might then have the following axiom, specifying the effect, on the relation *enrolled*, of registering a student in a course:

$$\begin{aligned}&Poss(register(st,c),s) \supset \\ &\qquad enrolled(st, c, \textit{choose-section(st,c,s)}, do(register(st,c),s)).\end{aligned}$$

Here the function *choose-section(st,c,s)* determines a section of the course c for the student st when the database is in state s.

Now suppose that $\neg enrolled(st, c, sec, s)$ and $enrolled(st, c, sec, do(a, s))$ are both true. How can we explain the fact that the negation of *enrolled* ceases to be true? If we assume that the only way this can happen is by registering st in c, we can express this with the explanation closure axiom:

$$\begin{aligned}&Poss(a,s) \wedge \neg enrolled(st,c,sec,s) \wedge enrolled(st,c,sec,do(a,s)) \\ &\qquad \supset a = register(st,c).\end{aligned}$$

Notice that this axiom universally quantifies over transactions a. To see how this functions as a frame axiom, rewrite it in the logically equivalent form:

$$\begin{aligned}&Poss(a,s) \wedge \neg enrolled(st,c,sec,s) \wedge a \neq register(st,c)\\ &\qquad \supset \neg enrolled(st,c,sec,do(a,s)).\end{aligned}$$

This says that all transactions other than *register* leave the negation of *enrolled* invariant, which is the standard form of a frame axiom (actually, a set of frame axioms, one for each transaction distinct from *register*). Now suppose that $enrolled(st,c,sec,s)$ and $\neg enrolled(st,c,sec,do(a,s))$ are both true. How can we explain the fact that *enrolled* ceases to be true? If we assume that the only way this can happen is by the student dropping the course, or by changing sections of the course, we can express this with the explanation closure axiom:

$$\begin{aligned}&Poss(a,s) \wedge enrolled(st,c,sec,s) \wedge \neg enrolled(st,c,sec,do(a,s))\\ &\quad \supset a = drop(st,c) \vee (\exists sec')a = change\text{-}section(st,c,sec,sec').\end{aligned}$$

In general, an *explanation closure axiom* has one of the two forms

$$Poss(a,s) \wedge R(\mathbf{x},s) \wedge \neg R(\mathbf{x},do(a,s)) \supset \alpha_R(\mathbf{x},a,s),$$

and

$$Poss(a,s) \wedge \neg R(\mathbf{x},s) \wedge R(\mathbf{x},do(a,s)) \supset \beta_R(\mathbf{x},a,s).$$

In these, the action variable a is universally quantified. These say that if ever the fluent R changes truth value as a result of a state transition, then α_R or β_R provides an exhaustive explanation for that change.

As before, the best way to see how explanation closure axioms function like frame axioms, rewrite them in their logically equivalent forms:

$$Poss(a,s) \wedge R(\mathbf{x},s) \wedge \neg\alpha_R(\mathbf{x},a,s) \supset R(\mathbf{x},do(a,s))$$

and

$$Poss(a,s) \wedge \neg R(\mathbf{x},s) \wedge \neg\beta_R(\mathbf{x},a,s) \supset \neg R(\mathbf{x},do(a,s)).$$

These have the same syntactic form as frame axioms with the important difference that the action a is universally quantified. Whereas there would be $2 \times \mathcal{A} \times \mathcal{F}$ frame axioms, where $\mathcal{A}$ is the number of actions and $\mathcal{F}$ the number of fluents, there are just $2 \times \mathcal{F}$ explanation closure axioms. This parsimonious representation is achieved precisely by quantifying over actions in the explanation closure axioms, on the assumption that the "explanations" $\alpha_R(\mathbf{x},a,s)$ and $\beta_R(\mathbf{x},a,s)$ are reasonably compact.

With the notion of an explanation closure axiom in hand, we are now in a position to offer some examples of proofs by induction.

4.2 Examples of Inductive Proofs

Example 1

We here present some axioms sufficient to inductively prove that if an object is broken and it never gets repaired, then it will always be broken:

$$\begin{aligned}&(\forall x).broken(x, S_0) \wedge [(\forall s).S_0 \leq s \supset \neg occurs(repair(x), s)] \supset \\ &\qquad (\forall s').S_0 \leq s' \supset broken(x, s').\end{aligned} \tag{9}$$

Recall that $occurs(a, s)$ means that a is one of the actions in the sequence of actions leading from S_0 to s. We require a simple axiom for $occurs$:

$$(\forall a, s) occurs(a, do(a, s)).$$

We also assume the following explanation closure axiom for $broken$:

$$Poss(a, s) \wedge broken(x, s) \wedge \neg broken(x, do(a, s)) \supset a = repair(x).$$

Together with the simple induction principle ($IP_{S_0 \leq s}$), these axioms are sufficient to prove (9).

Example 2

We here show how to establish the integrity constraint that salaries must never decrease during the evolution of a database:

$$\begin{aligned}&(\forall s, s')(\forall p, \$, \$').S_0 \leq s \wedge s \leq s' \wedge sal(p, \$, s) \wedge sal(p, \$', s') \\ &\qquad \supset \$ \leq \$'.\end{aligned} \tag{10}$$

We assume that if it is possible to change a person's salary, the new salary must be greater than the old:

$$Poss(change\text{-}sal(p, \$), s) \supset (\exists \$').sal(p, \$', s) \wedge \$' < \$.$$

We need the following axiom relating salary-changing transactions to salaries:

$$Poss(change\text{-}sal(p, \$'), s) \wedge sal(p, \$, do(change\text{-}sal(p, \$'), s)) \supset \$ = \$'.$$

We assume the following explanation closure axiom for sal:

$$Poss(a, s) \wedge \neg sal(p, \$, s) \wedge sal(p, \$, do(a, s)) \supset a = change\text{-}sal(p, \$).$$

Assume further that, initially, the relation *sal* is functional in its second argument:

$$sal(p, \$, S_0) \wedge sal(p, \$', S_0) \supset \$ = \$'.$$

We also need a *unique names axiom* for *change-sal*:

$$change\text{-}sal(p, \$) = change\text{-}sal(p', \$') \supset p = p' \wedge \$ = \$'.$$

These axioms, together with the double induction principle ($IP_{S_0 \leq s \leq s'}$), are sufficient to prove (10).

Acknowledgments

My special thanks to Vladimir Lifschitz, Fangzhen Lin and Javier Pinto for many important suggestions about getting the induction axioms right. Other valuable advice and suggestions were provided by Michael Gelfond, Hector Levesque and Wiktor Marek as well as the referees. Support for this research was provided by grants from the National Science and Engineering Research Council of Canada, and the Government of Canada Institute for Robotics and Intelligent Systems.

References

[1] J. Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Stanford University, Stanford, CA, 1986.

[2] M. Gelfond, V. Lifschitz, and A. Rabinov. What are the limitations of the situation calculus? In *Working Notes, AAAI Spring Symposium Series on the Logical Formalization of Commonsense Reasoning*, pages 59–69, 1991.

[3] C. C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 183–205. American Elsevier, New York, 1969.

[4] A. R. Haas. The case for domain-specific frame axioms. In F. M. Brown, editor, *The frame problem in artificial intelligence. Proceedings of the 1987 workshop*, pages 343–348, Los Altos, California, 1987. Morgan Kaufmann Publishers, Inc.

[5] V. Lifschitz. Toward a metatheory of action. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 376–386, Los Altos, CA, 1991. Morgan Kaufmann Publishers, Inc.

[6] F. Lin and R. Reiter. State constraints revisited. In *Second Symposium on Logical Formalizations of Commonsense Reasoning*, pages 114–121, Austin, Texas. Jan. 11-13, 1993.

[7] F. Lin and Y. Shoham. Provably correct theories of action. In *Proceedings of the National Conference on Artificial Intelligence*, 1991.

[8] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

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

[10] J. Pinto and R. Reiter. Adding a time line to the situation calculus. In *Second Symposium on Logical Formalizations of Commonsense Reasoning*, pages 172–177, Austin, Texas, Jan. 11-13, 1993.

[11] R. Reiter. Towards a logical reconstruction of relational database theory. In M.L. Brodie, J. Mylopoulos, and J.W. Schmidt, editors, *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages*, pages 191–233. Springer, New York, 1984.

[12] R. 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 359–380. Academic Press, San Diego, CA, 1991.

[13] R. Reiter. What should a database know? *Journal of Logic Programming*, 14(1-2):127–153, 1992.

[14] R. Reiter. On specifying database updates. Technical report, Department of Computer Science, University of Toronto, July, 1992.

[15] R. Reiter. On formalizing database updates: preliminary report. In *Proc. 3rd International Conference on Extending Database Technology*, pages 10–20, Vienna, March 23 - 27, 1992.

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

[17] T. Sheard and D. Stemple. Automatic verification of database transaction safety. *ACM Transactions on Database Systems*, 14(3):322–368, 1989.