How to Progress a Database (and Why)
I. Logical Foundations

Fangzhen Lin and Ray Reiter*
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
email: fl@ai.toronto.edu reiter@ai.toronto.edu

Abstract
One way to think about STRIPS is as a mapping from databases to databases, in the following sense: Suppose we want to know what the world would be like if an action, represented by the STRIPS operator $\alpha$, were done in some world, represented by the STRIPS database $D_0$. To find out, simply perform the operator $\alpha$ on $D_0$ (by applying $\alpha$’s elementary add and delete revision operators to $D_0$). We describe this process as progressing the database $D_0$ in response to the action $\alpha$. In this paper, we consider the general problem of progressing an initial database in response to a given sequence of actions. We appeal to the situation calculus and an axiomatization of actions which addresses the frame problem (Reiter [13], Lin and Reiter [8]). This setting is considerably more general than STRIPS. Our results concerning progression are mixed. The (surprising) bad news is that, in general, to characterize a progressed database we must appeal to second order logic. The good news is that there are many useful special cases for which we can compute the progressed database in first order logic; not only that, we can do so efficiently.

1 INTRODUCTION
One way to think about STRIPS is as a mapping from databases to databases, in the following sense: Suppose we want to know what the world would be like if an action, represented by the STRIPS operator $\alpha$, were done in some world, represented by the STRIPS database $D_0$. To find out, simply perform the operator $\alpha$ on $D_0$ (by applying $\alpha$’s elementary add and delete revision operators to $D_0$). We describe this process as progressing the database $D_0$ in response to the action $\alpha$ (cf. Rosenschein [15] and Pednault [9]). The resulting database describes the effects of the action on the world represented by the initial database. However, it may not always be convenient or even possible to describe the effects of actions as a simple process of progressing an initial world description. As we shall see in this paper, once we go beyond STRIPS-like systems, progression becomes surprisingly complicated.

In this paper, we consider the general problem of progressing an initial database in response to a given sequence of actions. We appeal to the situation calculus and an axiomatization of actions which addresses the frame problem (Reiter [13], Lin and Reiter [8]). This setting is considerably more general than STRIPS. Our results concerning progression are mixed. The (surprising) bad news is that, in general, to characterize a progressed database we must appeal to second order logic. The good news is that there are many useful special cases for which we can compute the progressed database in first order logic; not only that, we can do so efficiently.

The need to progress a database arises for us in a robotics setting. In our approach to controlling a robot, we must address the so-called projection problem: Answer the query $Q(do(A, S_0))$, where $do(A, S_0)$ denotes the situation resulting from performing the sequence of actions $A$ beginning with the initial situation $S_0$. This can be done using regression (cf. Waldinger [17], Pednault [10], and Reiter [12]) to reduce the projection problem to one of entailment from the initial database, consisting of sentences about the initial situation $S_0$. Unfortunately, regression suffers from a number of drawbacks in this application:

*Fellow of the Canadian Institute for Advanced Research

1This is also the way that database practitioners think about database updates (Abiteboul [1]). In fact, the STRIPS action and database update paradigms are essentially the same. Accordingly, this paper is as much about database updates as it is about STRIPS actions and their generalizations.

2Joint work with Yves Lespérance, Hector Levesque, Bill Millar and Richard Scherl.
1. After the robot has been functioning for a long time, the sequence $A$, consisting of all the actions it has performed since the initial situation, is extremely long, and regressing over such a long sequence becomes a computational burden.

2. Similarly, after a long while, the world state often becomes so rearranged that significantly many final steps of the regression become entirely unnecessary.

3. Most significantly, for robotics, perceptual actions (Scherl and Levesque [16]) lead to new facts being added to the database. But such facts are true in the current situation — the one immediately following the perceptual action — whereas the other (old) database facts are true in $S_0$. Reasoning about databases containing mixed facts — facts about the current and initial situations — is very complicated, and we know of no satisfactory way to do this.

Our way of addressing these problems with regression is to periodically progress the robot’s database. In particular, every perceptual action is accompanied by a progression of the database, coupled with the addition of the perceived fact to the resulting database. We envisage that these database progression computations can be done off-line, during the time when the robot is busy performing physical actions, like moving about.

2 LOGICAL PRELIMINARIES

The language $L$ of the situation calculus is a many-sorted first-order one with the sorts state for situations, action for actions, and object for anything else. We have the following domain independent predicates and functions: a unique constant $S_0$ of sort state; a binary function $do(a, s)$ that denotes the state resulting from performing the action $a$ in the state $s$; a binary predicate $Poss(a, s)$ that expresses the conditions for the action $a$ to be executable in the state $s$; and a binary predicate $<: state \times state$. We shall follow convention, and write $<$ in infix form. By $s < s'$ we mean that $s'$ can be obtained from $s$ by a sequence of executable actions. As usual, $s \leq s'$ will be a shorthand for $s < s' \lor s = s'$. We assume a finite number of state independent predicates which are the ones with arity $object^n$, $n \geq 0$, a finite number of state independent functions which are the ones with arity $object^n \rightarrow object$, $n \geq 0$, and a finite number of fluents which are predicate symbols of arity $object^n \times state$, $n \geq 0$.

We shall denote by $L^2$ the second-order extension of $L$. Our foundational axioms for the situation calculus will be in $L^2$ (Lin and Reiter [8]), because we need induction on situations (Reiter [14]).

We shall frequently need to restrict the situation calculus to a particular situation. For instance, the initial database is defined to be a finite set of sentences in $L$ that do not mention any state terms except $S_0$, and do not mention $Poss$ and $<$. For this purpose, for any state term $st$, we shall define $L_{st}$ to be the subset of $L$ that does not mention any other state terms except $st$, does not quantify over state variables, and does not mention $Poss$ and $<$. Formally, it is the smallest set satisfying

1. $\varphi \in L_{st}$ provided $\varphi \in L$ does not mention any state term.

2. $F(t_1, \ldots, t_n, st) \in L_{st}$ provided $F$ is a fluent of the right arity, and $t_1, \ldots, t_n$ are terms of the right sort.

3. If $\varphi$ and $\varphi'$ are in $L_{st}$, so are $\neg \varphi$, $\varphi \lor \varphi'$, $\varphi \land \varphi'$, $\varphi \rightarrow \varphi'$, $(\forall x)\varphi$, $(\exists x)\varphi$, $(\forall a)\varphi$, $(\exists a)\varphi$, and $(\exists p)\varphi$, where $x$ and $a$ are variables of sort object and action, respectively.

We remark here that according to this definition, $(\forall a)F(do(a, S_0))$ will be in $L_{do(a, S_0)}$. This may seem odd when we want sentences in $L_{st}$ to be propositions about situation $st$. Fortunately, we shall use $L_{st}$ only when $st$ is either a ground term or a simple variable of sort state.

We shall use $L^2_{st}$ to denote the second-order extension of $L_{st}$ by predicate variables of arity $object^n$, $n \geq 0$. So the second-order sentence $(\exists p)(\forall x) p(x) \equiv F(x, S_0)$ is in $L^2_{st}$, but $(\exists p)(\forall x)(\exists s) p(x, s) \equiv F(x, S_0)$ is not, since the latter quantifies over a predicate variable of arity $object \times state$. Formally, $L^2_{st}$ is the smallest set satisfying

1. Every formula in $L_{st}$ is also in $L^2_{st}$.

2. $p(t_1, \ldots, t_n) \in L^2_{st}$ provided $p$ is a predicate variable of arity $object^n$, $n \geq 0$, and $t_1, \ldots, t_n$ are terms of sort object.

3. If $\varphi$ and $\varphi'$ are in $L^2_{st}$, so are $\neg \varphi$, $\varphi \lor \varphi'$, $\varphi \land \varphi'$, $\varphi \rightarrow \varphi'$, $(\forall x)\varphi$, $(\exists x)\varphi$, $(\forall a)\varphi$, $(\exists a)\varphi$, and $(\exists p)\varphi$, where $x$ and $a$ are variables of sort object and action, respectively, and $p$ is a predicate variable of arity $object^n$, $n \geq 0$.

3 BASIC ACTION THEORIES

We assume that our action theory $D$ has the following form (cf. Reiter [13] and Lin and Reiter [8]):

$$D = \Sigma \cup D_{st} \cup D_{ap} \cup D_{ana} \cup D_{sa},$$

where

- $\Sigma$, given below, is the set of the foundational axioms for situations.
\(D_{ss}\) is a set of successor state axioms of the form:\(^3\)
\[
\text{Poss}(a, s) \supset F(\bar{x}, \text{do}(a, s)) \equiv \Phi_F(\bar{x}, a, s),
\]
where \(F\) is a fluent, and \(\Phi_F(\bar{x}, a, s)\) is in \(L_s\).

- \(D_{ap}\) is a set of action precondition axioms of the form:
\[
\text{Poss}(A(\bar{x}), s) \equiv \Psi_A(\bar{x}, s),
\]
where \(A\) is an action, and \(\Psi_A(\bar{x}, s)\) is in \(L_s\).

- \(D_{una}\) is the set of unique names axioms for actions: For any two different actions \(A(\bar{x})\) and \(A'(\bar{y})\), we have
\[
A(\bar{x}) \neq A'(\bar{y}),
\]
and for any action \(A(x_1, \ldots, x_n)\), we have
\[
A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n) \supset
x_1 = y_1 \land \cdots \land x_n = y_n.
\]

- \(D_{s}\), the initial database, is a finite set of first-order sentences in \(L_{s}\).

We shall give an example of our action theory in a moment. First, we briefly explain our foundational axioms \(\Sigma\) since they are independent of particular applications. \(\Sigma\) contains axioms about the structure of situations. Formally, \(\Sigma\) is the following set of axioms:
\[
S_0 \neq \text{do}(a, s),
\]
\[
\text{do}(a_1, s_1) = \text{do}(a_2, s_2) \supset (a_1 = a_2 \land s_1 = s_2),
\]
\[
(\forall P)[P(S_0) \land (\forall a, s)(P(s) \supset P(\text{do}(a, s))) \supset (\forall s)P(s)],
\]
\[
-s < S_0,
\]
\[
s < \text{do}(a, s') \equiv (\text{Poss}(a, s') \land s \leq s').
\]

Notice the similarity between \(\Sigma\) and Peano Arithmetic. The first two axioms are unique names assumptions. They eliminate finite cycles, and merging. The third axiom is second-order induction. It amounts to the domain closure axiom which says that every situation has to be obtained from repeatedly applying do to \(S_0\).\(^4\) The last two axioms define \(<\) inductively.

\(\Sigma\) is the only place where axioms about the structure of situations can appear. It is needed only if we want to show, usually by induction, that a state constraint of the form \((\forall s)C(s)\) is entailed by an action theory. For the purpose of temporal projection, in particular progression as we shall see, \(D\) has exactly the same effect as \(D - \Sigma\): For any formula \(\varphi(s)\) in \(L_s\), and any sequence \(A\) of actions,
\[
D \models \varphi(\text{do}(A, S_0))
\]

\(^3\) In the following, unless otherwise stated, all free variables in a formula are assumed to be universally quantified from the outside.

\(^4\) For a discussion of the use of induction in the situation calculus, see (Reiter [14]).

This follows directly from the following proposition which will be used throughout this paper.

**Proposition 3.1** Given any model \(M\) of \(D_{ss} \cup D_{ap} \cup D_{una} \cup D_{s}\), there is a model \(M'\) of \(D\) such that:

1. \(M'\) and \(M\) have the same domains for sorts action and object, and interpret all state independent predicates and functions the same;

2. For any sequence \(A\) of actions, any fluent \(F\), and any variable assignment \(\sigma:\)
\[
M', \sigma \models F(\bar{x}, \text{do}(A, S_0))
\]
iff
\[
M, \sigma \models F(\bar{x}, \text{do}(A, S_0)).
\]

**Example 3.1** An educational database (Reiter [13]). There are two fluents:

- \(\text{enrolled}(st, \text{course}, s)\): student \(st\) is enrolled in course \(\text{course}\) in state \(s\).

- \(\text{grade}(st, \text{course}, \text{grade}, s)\): the grade of \(st\) in course \(\text{course}\) is grade \(\text{grade}\) in state \(s\).

There are two state independent predicates:

- \(\text{prereq}(\text{pre}, \text{course})\): \(\text{pre}\) is a prerequisite course for course \(\text{course}\).

- \(\text{better}(\text{grade}1, \text{grade}2)\): grade \(\text{grade}1\) is better than grade \(\text{grade}2\).

There are three database transactions:

- \(\text{register}(st, \text{course})\): register the student \(st\) into course \(\text{course}\).

- \(\text{change}(st, \text{course}, \text{grade})\): change the grade of the student \(st\) in course \(\text{course}\) to grade \(\text{grade}\).

- \(\text{drop}(st, \text{course})\): drop the student \(st\) from course \(\text{course}\).

This setting can be axiomatized as follows.

\(D_{ss}\) is the set of following successor state axioms:
\[
\text{Poss}(a, s) \supset
\]
\[
\epsilon_{\text{enrolled}}(st, c, \text{do}(a, s)) \equiv
a = \text{register}(st, c) \lor
\text{enrolled}(st, c, s) \land a \neq \text{drop}(st, c),
\]
\[
\text{Poss}(a, s) \supset
\]
\[
\text{grade}(st, c, g, \text{do}(a, s)) \equiv
a = \text{change}(st, c) \lor
\text{grade}(st, c, g, s) \land (\forall g')a \neq \text{change}(st, c, g').
\]
4 FORMAL FOUNDATIONS

Let $\alpha$ be a ground simple action, e.g. $\text{enroll}(\text{Sue}, 100)$, and let $S_0$ denote the state term $do(\alpha, S_0)$. A progression $D_{S_0}$ of $D_{S_0}$ in response to $\alpha$ should have the following properties:

1. $D_{S_0}$ is a set of sentences about state $S_0$ only, i.e., in $L_{S_0}$ or in $L_{S_0}^2$.
2. For all queries about the future of $S_0$, $D$ is equivalent (in a suitable formal sense) to $\Sigma \cup D_{S_0} \cup D_{ap} \cup D_{una} \cup D_{S_0}$

In other words, $D_{S_0}$ acts like the new initial database wrt all possible future evolutions of the theory following $\alpha$.

To define progression, we first introduce an equivalence relation over structures. Let $M$ and $M'$ be structures (for our language) with the same domains for sorts action and object. Define $M' \sim_{S_0} M$ iff the following two conditions hold:

1. $M'$ and $M$ interpret all predicate and function symbols which do not take any arguments of sort state identically.
2. $M$ and $M'$ agree on all fluents at $S_0$: For every predicate fluent $F$, and every variable assignment $\sigma$, $M', \sigma \models F(\bar{x}, do(\alpha, S_0))$ iff $M, \sigma \models F(\bar{x}, do(\alpha, S_0))$.

It is clear that $\sim_{S_0}$ is an equivalence relation. If $M' \sim_{S_0} M$, then $M'$ agrees with $M$ on $S_0$ on fluents and state independent predicates and functions, but is free to vary its interpretation of everything else on all other states. In particular, they can interpret $Poss$ and $do$ differently. We have the following simple lemma.

**Lemma 4.1** If $M \sim_{S_0} M'$, then for any formula $\phi$ in $L_{S_0}^2$, and any variable assignment $\sigma$, $M, \sigma \models \phi$ iff $M', \sigma \models \phi$. So we define

**Definition 4.1** A set of sentences $D_{S_0}$ in $L_{S_0}^2$ is a progression of the initial database $D_{S_0}$ to $S_0$ (wrt $D$) iff for any structure $M$, $M$ is a model of $D_{S_0}$ iff there is a model $M'$ of $D$ such that $M \sim_{S_0} M'$.

Notice that we define the new database only up to logical equivalence. We allow the new database to contain second-order sentences because, as we shall see later, first-order logic is not expressive enough for our purposes.

**Proposition 4.1** Let $D_{S_0}$ be a progression of the initial database to $S_0$. Then $\models D_{S_0} \subseteq \models (\Sigma \cup D_{S_0} \cup D_{ap} \cup D_{una} \cup D_{S_0})$.

**Proposition 4.2** Let $D_{S_0}$ be a progression of the initial database to $S_0$. Then for every model $M$ of $\Sigma \cup D_{S_0} \cup D_{ap} \cup D_{una} \cup D_{S_0}$, there exists a model $M'$ of $D$ such that:

1. $M'$ and $M$ interpret all state independent predicate and function symbols identically.
2. For every variable assignment $\sigma$, and every predicate fluent $F$, $M', \sigma \models s \land F(\bar{x}, s)$ iff $M, \sigma \models s \land F(\bar{x}, s)$.

**Proof:** Let $M$ be a model of $\Sigma \cup D_{S_0} \cup D_{ap} \cup D_{una} \cup D_{S_0}$. Since $M$ is a model of $D_{S_0}$, there is a model $M'$ of $\Sigma \cup D_{S_0} \cup D_{ap} \cup D_{una} \cup D_{S_0}$ such that $M' \sim_{S_0} M$. It can be easily seen that $M'$ satisfies the desired properties.

From these two propositions, we conclude that $D$ and $\Sigma \cup D_{S_0} \cup D_{ap} \cup D_{una} \cup D_{S_0}$ agree on all states $\geq S_0$. So $D_{S_0}$ really does characterize the result of progressing the initial database in response to the action $\alpha$. Furthermore, the following theorem says that the new database, when it exists, entails the same set of sentences in $L_{S_0}^2$ as $D$.

**Theorem 1** Let $D_{S_0}$ be a progression of the initial database to $S_0$. For any sentence $\phi \in L_{S_0}^2$, $D_{S_0} \models \phi$ iff $D \models \phi$.

**Proof:** If $D \models \phi$, then by Lemma 1, we have $D_{S_0} \models \phi$. If $D_{S_0} \models \phi$, then $D \models \phi$ by Proposition 4.1.
From this theorem, we see that if \( \mathcal{D}_b \) is a progression, then it is a strongest post-condition (cf. Pelednaud [9], Dijkstra and Scholten [3], and others) of the precondition \( \mathcal{D}_a \) w.r.t. the action \( \alpha \). A result by Pelednaud [9] shows that \( \mathcal{D}_a \) cannot in general be a finite set of first-order sentences in \( \mathcal{L}_{a_b} \). In the following, we shall extend this result, and show that \( \mathcal{D}_a \) cannot in general be a set of first-order sentences in \( \mathcal{L}_{a_b} \).

4.1 Progression Is Not Always First-Order Definable

At first glance, the fact that progression cannot always be done in first-order logic may seem obvious in light of Theorem 1, and the fact that \( \mathcal{D} \) includes a second-order induction axiom. However, as we mentioned in section 3, for the purpose of progression, \( \mathcal{D} \) is equivalent to \( \mathcal{D} = \Sigma \), which is a finite set of first-order sentences.

We shall construct a basic action theory \( \mathcal{D} \) and two structures \( M_1 \) and \( M_2 \) with the following properties:

1. \( M_1 \models \mathcal{D} \).
2. \( M_1 \) and \( M_2 \) satisfy the exactly same set of sentences in \( \mathcal{L}_{a_b} \).
3. There is no model \( M' \) of \( \mathcal{D} \) such that \( M' \sim_{a_b} M_2 \).

It will then follow from our definition that for \( \mathcal{D} \), the progression of the initial database to \( a_b \) cannot be in \( \mathcal{L}_{a_b} \). This is possible because for \( M \sim_{a_b} M' \) to hold, \( M \) and \( M' \) must be isomorphic with respect to sort \( \text{object} \); but in number theory, there are nonstandard models that satisfy exactly the same set of first-order sentences as the standard model, and it is this property which we now use to show that progression is not always first-order definable.

We now proceed to construct a such basic action theory. Consider the following theory \( \mathcal{D} \) with a unary fluent \( F_1 \), and a binary fluent \( F_2 \), one action constant symbol \( A \), one constant symbol 0, and one unary function symbol \( \text{succ} \):

\[
\begin{align*}
\mathcal{D}_a &= \emptyset, \mathcal{D}_{una} = \emptyset, \\
\mathcal{D}_p &= \{(\forall s).\text{Poss}(A, s) \equiv \text{True}\}, \\
\mathcal{D}_{ss} &= \text{the following pair of axioms:} \\
\text{Poss}(a, s) &\supset [F_1(\text{do}(a, s)) \equiv (\exists x).\neg F_2(x, s)], \\
\text{Poss}(a, s) &\supset (\forall x).F_2(x, \text{do}(a, s)) \equiv \neg F_2(x, s) \land (\exists y).F_2(y, s) \lor \neg F_2(x, s) \land x \neq 0 \land (\forall y).F_2(y, s) \lor \neg F_2(x, s).
\end{align*}
\]

To understand the successor state axioms, think of the constant symbol 0 as the number 0, and the unary function \( \text{succ} \) as the successor function. \( F_1 \) simply keeps track of the truth value of \( F_2 \) in the previous state, and for \( F_2(x, \text{do}(a, s)) \) to be true, either \( x = 0 \) and \( F_2(x, s) \), or both \( F_2(x, s) \) and \( F_2(\text{predecessor}(x), s) \) have the same truth values.

Consider a structure \( M \) such that:

1. \( M \) is a standard model of arithmetic with respect to sort \( \text{object} \). Thus the domain for \( \text{object} \) in \( M \) is the set of nonnegative numbers, 0 is mapped to the number 0, and \( \text{succ} \) is mapped to the successor function.
2. \( M \models F_1(\text{do}(A, S_0)) \land (\forall x).F_2(x, \text{do}(A, S_0)). \)

Our first observation is that there cannot be a model \( M' \) of \( \mathcal{D} \) such that \( M \sim_{a_b} M' \). Suppose otherwise. Then \( M' \) also satisfies the above two properties 1 and 2. From \( M' \models \mathcal{D}_a \), and \( M' \models F_1(\text{do}(A, S_0)) \), we have \( M' \models (\exists x).\neg F_2(x, S_0) \). Similarly, from \( M' \models (\forall x).F_2(x, \text{do}(A, S_0)) \), by the successor state axiom for \( F_2 \), we have \( M' \models F_2(0, S_0) \land F_2(\text{succ}(0), S_0) \land \ldots \).

Thus \( M' \models (\forall y).F_2(x, S_0) \), a contradiction. Therefore there is not a model \( M' \) of \( \mathcal{D} \) such that \( M \sim_{a_b} M' \).

We now show that there is a model \( M' \) of \( \mathcal{D} \) such that for any sentence \( \varphi \) in \( \mathcal{L}_{a_b} \), \( M \models \varphi \iff M' \models \varphi \). By Skolem's theorem (cf. Kleene [5], page 326), there is a first-order structure \( M' \) such that for any sentence \( \varphi \) in \( \mathcal{L}_{a_b} \), \( M \models \varphi \iff M' \models \varphi \), and \( (M, 0, \text{succ}) \) are not isomorphic, i.e., \( M \) and \( M' \) are not isomorphic on sort \( \text{object} \). In particular, \( M' \models F_1(\text{do}(A, S_0)) \land (\forall x).F_2(x, \text{do}(A, S_0)). \) Now revise \( M' \) into a structure \( M^{*} \) such that:

1. \( M' \) and \( M^{*} \) have the same domains for sorts \( \text{action} \) and \( \text{object} \), and interpret state independent predicates and functions the same.
2. \( M' \models (\forall s, a).\text{Poss}(A, s) \).
3. \( M' \models \Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{a} \).
4. For the truth values of the fluentes on \( S_0 \): \( M' \models F_1(S_0) \), and for the truth values of \( F_2(x, S_0) \), we have that for any variable assignment \( \sigma \):

(a) If \( \sigma(x) \) is a standard number, i.e., there is a \( n \geq 0 \) such that \( M', \sigma \models x = \text{succ}^n(0) \), then \( M', \sigma \models F_2(x, S_0) \).

(b) If \( \sigma(x) \) is a nonstandard number, i.e., there is no \( n \geq 0 \) such that \( M', \sigma \models x = \text{succ}^n(0) \), then \( M', \sigma \models \neg F_2(x, S_0) \). Notice that since \( M^{*} \) and \( M' \) are not isomorphic on sort \( \text{object} \) with respect to Peano arithmetic, there must be a nonstandard number in the domain of \( M^{*} \), and thus in the domain of \( M' \).

5. For the truth values of the fluentes on \( \text{do}(A, S_0) \):

For any fluent \( F_i \) and any variable assignment \( \sigma \), \( M', \sigma \models F(x, \text{do}(A, S_0)) \) iff \( M^{*}, \sigma \models F(x, \text{do}(A, S_0)). \)
6. Inductively, for any variable assignment \( \sigma \), if

\[ M', \sigma \models do(A, S_0) < s, \]

then the truth values of the fluents on \( s \) will be determined according to the successor state axioms and the truth values of the fluents on \( do(A, S_0) \); if

\[ M', \sigma \models S_0 < s \land \neg do(A, S_0) < s, \]

then the truth values of the fluents on \( s \) will be determined according to the successor state axioms and the truth values of the fluents on \( S_0 \). This will define the truth values of the fluents on every state because \( M' \models (\forall s).S_0 \leq s \), which follows from the fact that \( M' \models (\forall a, s)Poss(a, s) \).

It is clear that \( M' \sim_{S_A} M^* \). It follows that \( M' \) and \( M \) satisfy the same set of sentences in \( L_{S_A} \). We now show that \( M' \) satisfies the successor state axioms. By the construction of \( M' \), we only need to prove that it satisfies the successor state axioms instantiated to \( S_0 \) and action \( A \), i.e.,

\[ M' \models Poss(A, S_0) \supset [F_1(do(A, S_0)) \equiv (3x)\neg F_1(x, S_0)], \]

and

\[ M' \models Poss(A, S_0) \supset (\forall y).F_2(x, do(A, S_0)) \equiv x = 0 \land F_2(0, S_0) \lor\]

\[ F_2(x, S_0) \land (3y).x = suc(y) \land F_2(y, S_0) \lor\]

\[ \neg F_2(x, S_0) \land x \neq 0 \land\]

\[ (\forall y)(x = suc(y) \supset \neg F_2(y, S_0)).\]

To show the first one, we need to prove that \( M' \models (\exists x).\neg F_2(s, S_0) \). This follows from our construction of \( M' \) and the existence of nonstandard numbers in the domain of \( M' \). To show the second one, we need to prove that

\[ M' \models (\forall x).x = 0 \lor F_2(0, S_0) \lor\]

\[ F_2(x, S_0) \land (3y).x = suc(y) \land F_2(y, S_0) \lor\]

\[ \neg F_2(x, S_0) \land x \neq 0 \land\]

\[ (\forall y)(x = suc(y) \supset \neg F_2(y, S_0)).\]

There are three cases:

1. If \( x = 0 \), then \( F_2(0, S_0) \) follows from our construction.

2. If \( x = suc^n(0) \) for some \( n > 0 \), then both \( F_2(suc^n(0), S_0) \) and \( F_2(suc^{n-1}(0), S_0) \) hold. Thus \( F_2(x, S_0) \land (3y).x = suc(y) \land F_2(y, S_0) \lor\]

\[ \neg F_2(x, S_0) \land x \neq 0 \land\]

\[ (\forall y)(x = suc(y) \supset \neg F_2(y, S_0)).\]

3. If \( x \) is a nonstandard number, then \( F_2(x, S_0) \) does not hold. Furthermore, for any \( y \) such that \( x = suc(y) \), \( y \) is also a nonstandard number, so \( F_2(y, S_0) \) does not hold either. Thus \( \neg F_2(x, S_0) \land x \neq 0 \land (\forall y)(x = suc(y) \supset \neg F_2(y, S_0)) \).

Therefore, \( M' \) satisfies the successor state axioms instantiated to \( S_0 \) and \( A \). So \( M' \models D_{ss} \). This means that \( M' \models D \), and \( M' \) and \( M \) satisfy the same sentences in \( L_{S_A} \). Following the discussion at the beginning of the example, we see that the new database at \( S_A \) for \( D \) cannot be captured by a set of first-order sentences.

4.2 Progression Is Always Second-Order Definable

We now show that, by appealing to second-order logic, progression always exists. We shall first introduce some notation.

Given a finite set \( D_{ss} \) of successor state axioms, we define the instantiation of \( D_{ss} \) on an action term \( at \) and a state term \( st \), written \( D_{ss}[at, st] \), to be the sentence:

\[ \bigwedge \text{Poss(at, st) } \supset \]

\[ F \text{ is a fluent } (\forall x).F(x, do(at, st)) ] (3y).F(x, y, st) \]

where

\[ (\forall a, s).Poss(a, s) ] (3y).F(x, y, s) \]

is the successor state axiom for \( F \) in \( D_{ss} \).

Given a formula \( \varphi \) in \( L^2 \), the lifting of \( \varphi \) on the state \( st \), written \( \varphi \upharpoonright st \), is the result of replacing every fluent atom of the form \( F(t_1, ..., t_n, s) \) by a new predicate variable \( p(t_1, ..., t_n) \) of arity object\( ^* \). For instance, \( \text{enrolled}(John, C200, S0) \wedge \text{enrolled}(John, C100, S0) \upharpoonright S0 \)

is \( p(John, C200) \wedge p(John, C100) \) \( ^5 \).

**Lemma 4.2** The following are some simple properties of lifting:

1. If \( \varphi \) is a sentence that does not mention \( st \), then \( \varphi \upharpoonright st \) is \( \varphi \).

2. If \( \varphi \) is a sentence in \( L^2_{ss} \), then \( \varphi \upharpoonright st \) is a state independent sentence.

3. If \( \varphi \) does not contain quantifiers over states, then \( \varphi \models \varphi \upharpoonright st \).

Now we can state the main theorem of this section:

**Theorem 2** Let \( D_{ss} \) be the union of \( D_{una} \) together with the sentence:

\[ (3p_1, ..., p_k) \{ (\forall \varphi \in D_{ss} \upharpoonright \varphi_D)[Poss(\varphi_D)] \upharpoonright S0, \]

where

\[ 1. p_1, ..., p_k \text{ are the new predicate variables introduced during the lifting.} \]

\( ^5 \)Lifting as we have defined it does not generally preserve logical equivalence. For instance, \( \text{[\( \forall s \).F(s)] \upharpoonright S0} \) is \( \text{[\( \forall s \).F(s)] \upharpoonright S0} \), but the logically equivalent \( [F(S_0) \wedge (\forall s).F(s)] \upharpoonright S0 \) is \( \text{[\( \forall s \).F(s)] \upharpoonright S0} \). Fortunately, we shall only be lifting those sentences that do preserve logical equivalence.
2. $\Psi_a$ is a sentence in $\mathcal{L}_{S_a}$ such that 
$\text{Poss}(\alpha, S_0) \equiv \Psi_a$

is an instance of the the axiom in $\mathcal{D}_{ap}$ corresponding to the action of $\alpha$.

3. $D_x[\alpha, S_0](\text{Poss}/\Psi_a)$ is the result of replacing $\text{Poss}(\alpha, S_0)$ by $\Psi_a$ in $D_x[\alpha, S_0]$.

Then $D_{S_a}$ is a progression of $D_{S_a}$ to $S_a$ wrt $D$.

**Proof:** First, it is clear that the sentences in $D_{S_a}$ are in $\mathcal{L}_{S_a}^2$.

Let $M$ be a structure. We need to show that $M \models D_{S_a}$ iff there is a model $M'$ of $D$ such that $M \sim_{S_a} M'$.

Suppose that there is a model $M'$ of $D$ such that $M \sim_{S_a} M'$. By Lemma 4.2, $D \models D_{S_a}$; thus $M' \models D_{S_a}$. Therefore by Lemma 4.1, $M \models D_{S_a}$.

Now suppose that $M \models D_{S_a}$. Then there is a variable assignment $\sigma$ such that 

$$M, \sigma \models \bigwedge_{\varphi \in D_{S_a}} \varphi \land D_x[\alpha, S_0](\text{Poss}/\Psi_a), S_0.$$ 

Now construct a structure $M'$ such that

1. $M$ and $M'$ have the same universe, and interpret all state independent function and predicate symbols identically.
2. For every fluent $F$, if $F(\bar{x}, S_0)$ is lifted in $D_{S_a}$ as $p$, then 

$$M', \sigma' \models F(\bar{x}, S_0) \text{ iff } M, \sigma \models p(\bar{x}).$$

3. $M' \models D_{S_a} \cup D_{ap}$.
4. If $M' \models \neg \Psi_a$, then for any fluent $F$, and any variable assignment $\sigma'$, 

$$M', \sigma' \models F(\bar{x}, S_0) \text{ iff } M, \sigma \models F(\bar{x}, S_0).$$

It is clear that such a $M'$ exists. We claim that $M \sim_{S_a} M'$. There are two cases:

1. If $M' \models \neg \Psi_a$, then it follows from our construction that for any fluent $F$, and any variable assignment $\sigma'$,

$$M', \sigma' \models F(\bar{x}, S_0),$$

and $M, \sigma \models F(\bar{x}, S_0)$.

2. If $M' \models \Psi_a$, then since $M' \models D_{ap}$, and $D_{ap} \models \text{Poss}(\alpha, S_0) \equiv \Psi_a$, therefore $M' \models \text{Poss}(\alpha, S_0)$. But $M' \models D_{S_a}$. Thus for any fluent $F$, and any variable assignment $\sigma'$,

$$M', \sigma' \models F(\bar{x}, S_0) \text{ iff } M, \sigma \models \Phi_F(\bar{x}, \alpha, S_0),$$

where $\Phi_F$ is as in the successor state axiom for $F$ in $D_{S_a}$. Now since $M' \models \Psi_a$, by our construction of $M'$, we have that $M, \sigma \models \Psi_a \models S_0$. But

$$M, \sigma \models D_{S_a}[\alpha, S_0](\text{Poss}/\Psi_a), S_0.$$ 

Therefore for any fluent $F$, and any variable assignment $\sigma'$ such that $\sigma'(p) = \sigma(p)$ for any predicate variable $p$,

$$M, \sigma' \models F(\bar{x}, S_0) \text{ iff } M', \sigma' \models \Phi_F(\bar{x}, \alpha, S_0) \models S_0. \quad (2)$$

But for any variable assignment $\sigma'$ such that $\sigma'(p) = \sigma(p)$ for any predicate variable $p$, since $\Phi_F(\bar{x}, \alpha, S_0)$ is in $\mathcal{L}_{S_a}$, by our construction of $M'$,

$$M, \sigma' \models \Phi_F(\bar{x}, \alpha, S_0) \models S_0 \text{ iff } M', \sigma' \models \Phi_F(\bar{x}, \alpha, S_0),$$

Therefore from (1) and (2), we see that for any fluent $F$, and any variable assignment $\sigma'$,

$$M', \sigma' \models F(\bar{x}, S_0) \text{ iff } M, \sigma' \models F(\bar{x}, S_0).$$

It follows then that $M \sim_{S_a} M'$. By the construction of $M'$ and the fact that $M \models D_{una}$, we have that $M' \models D_{S_a} \cup D_{ap} \cup D_{una}$. Thus from Proposition 3.1, there is a model $M''$ of $D$ such that $M \sim_{S_a} M''$. Then by the transitivity of $\sim_{S_a}$, we have that $M \sim_{S_a} M''$. This concludes the proof that $D_{S_a}$ as defined is progressed database.

It is clear that the theorem still holds when the initial database $D_{S_a}$ is a finite set of second-order sentences in $\mathcal{L}_{S_a}^2$. Therefore, at least in principle, the theorem can be repeatedly applied to progress the initial database in response to a sequence of actions.

The new database $D_{S_a}$ as defined in the theorem can be unwieldy. However, it can often be simplified by using the unique names axioms in $D_{una}$, as we shall see in the following example.

**Example 4.1** Consider our educational database.

The instantiation of the successor state axioms on drop$(\text{Suc}, C100)$ and $S_0$, $D_{S_a}[\text{drop}(\text{Suc}, C100), S_0]$, is the conjunction of the following two sentences, where $\alpha = \text{drop}(\text{Suc}, C100)$ and $S_a = \text{do}(\alpha, S_0)$:

$$\text{Poss}(\alpha, S_0) \supset \text{enrolled}(\text{st}, c, S_a) \equiv \alpha = \text{register}(\text{st}, c) \lor \text{enrolled}(\text{st}, c, S_0) \land \alpha \neq \text{drop}(\text{st}, c),$$

$$\text{Poss}(\alpha, S_0) \supset \text{grade}(\text{st}, c, g, S_a) \equiv \alpha = \text{change}(\text{st}, c) \lor \text{grade}(\text{st}, c, g, S) \land (\forall g')\alpha \neq \text{change}(\text{st}, c, g').$$

By unique names axioms, these two sentences can be simplified to

$$\text{Poss}(\alpha, S_0) \supset \text{enrolled}(\text{st}, c, S_a) \equiv \text{enrolled}(\text{st}, c, S_0) \land (\text{Suc} \neq \text{st} \lor C100 \neq c),$$

$$\text{Poss}(\alpha, S_0) \supset \text{grade}(\text{st}, c, g, S_a) \equiv \text{grade}(\text{st}, c, g, S),$$

By $D_{ap}$,

$$\text{Poss}(\alpha, S_0) \equiv \text{enrolled}(\text{Suc}, C100, S_0).$$
Thus $D_{ss}[\alpha, S_0](\text{Poss}/\Psi_\alpha)$ is the conjunction of the following two sentences:

- $\text{enrolled}(\text{Sue}, C100, S_0) \supset \text{enrolled}(st, c, S_0) \equiv \text{enrolled}(st, c, S_0) \land (Sue \neq st \lor C100 \neq c)$,
- $\text{enrolled}(\text{Sue}, C100, S_0) \supset \text{grade}(st, c, g, S_0) \equiv \text{grade}(st, c, g, S_0)$.

Thus $(\exists p_1, p_2)(\forall \varphi \in D_{ss}[\alpha, S_0](\text{Poss}/\Psi_\alpha)) \land S_0$ is

$(\exists p_1, p_2). {\text{John}} \neq {\text{Sue}} \neq C100 \neq C200 \land \text{prerequ}(C100, C200) \land \text{prerequ}(C100, C200) \land (\exists p_1, p_2). {\text{John}} \neq {\text{Sue}} \neq C100 \neq C200 \land \text{enrolled}(st, c, S_0) \equiv p_1(st, c) \land (\exists p_1, p_2). {\text{Sue}} \neq st \land C100 \neq c \land \text{grade}(st, c, g, S_0) \equiv p_2(st, c, g)$.

This is equivalent to

$(\exists p_1, p_2). {\text{John}} \neq {\text{Sue}} \neq C100 \neq C200 \land \text{prerequ}(C100, C200) \land (\exists p_1, p_2). {\text{John}} \neq {\text{Sue}} \neq C100 \neq C200 \land \text{enrolled}(st, c, S_0) \equiv p_1(st, c) \land (\exists p_1, p_2). {\text{Sue}} \neq st \land C100 \neq c)$.

which is equivalent to

$(\exists p_1, p_2). {\text{John}} \neq {\text{Sue}} \neq C100 \neq C200 \land \text{prerequ}(C100, C200) \land (\exists p_1, p_2). {\text{John}} \neq {\text{Sue}} \neq C100 \neq C200 \land \text{enrolled}(st, c, S_0) \equiv p_1(st, c)$.

Finally, we have a first-order representation for $D_{ss}$, which is $D_{una}$ together with the following sentences:

- $\text{John} \neq \text{Sue} \neq C100 \neq C200$,
- $\text{prerequ}(C100, C200)$,
- $\text{enrolled}(\text{John}, C100, S_0) \lor \text{enrolled}(\text{John}, C200, S_0)$,
- $\neg \text{enrolled}(\text{Sue}, C100, S_0)$.

To summarize, we have shown that in general, progression is definable only in second-order logic. However, there are some interesting special cases for which progression can be done in first-order logic. We shall give two such special cases.

5 PROGRESSION WITH RELATIVELY COMPLETE INITIAL DATABASES

We say $D_{ss}$ is relatively complete (wrt state independent propositions) if it is a set of state independent sentences together with a set of sentences, one for each fluent $F$, of the form:

$(\forall \bar{x}). F(\bar{x}, S_0) \equiv \Pi_F(\bar{x})$,

where $\Pi_F(\bar{x})$ is a state independent formula whose free variables are among $\bar{x}$. Clearly, for relatively complete $D_{ss}$, if it is complete about the state independent sentences: For any state independent sentence $\Pi$,

- either $D_{ss} \models \Pi$ or $D_{ss} \models \neg \Pi$,

then it is also complete about $S_0$: For any sentence $\varphi$ in $L_{S_0}$,

- either $D_{ss} \models \varphi$ or $D_{ss} \models \neg \varphi$.

Theorem 3 Let $D$ be an action theory with a relatively complete initial database $D_{ss}$, and let $\alpha$ be a ground action term such that $D \models \text{Poss}(\alpha, S_0)$. Then the following set:

$D_{una} \cup \{ \varphi \mid \varphi \in D_{ss} \text{ is state independent} \} \cup \{(\forall \bar{x}). F(\bar{x}, do(\alpha, S_0)) \equiv \Phi_F(\bar{x}, \alpha, S_0)[S_0] \mid F \text{ is a fluent} \}$

is a progression of $D_{ss}$ to $S_0$, where

1. $\Phi_F(\bar{x}, \alpha, S_0)$ is as in the successor state axiom for $F$ in $D_{ss}$;
2. $\Phi_F(\bar{x}, \alpha, S_0)[S_0]$ is the result of replacing, in $\Phi_F(\bar{x}, \alpha, S_0)$, every occurrence of $F'(\bar{t})$ by $\Pi_{F'}(\bar{t})$, where $\Pi_{F'}$ is as in the corresponding axiom for $F'$ in $D_{ss}$, and this replacement is performed for every fluent $F'$ mentioned in $\Phi_F(\bar{x}, \alpha, S_0)$.

Proof: Denote the set of the sentences of the theorem by $S$. Clearly, $S$ is a set of first-order sentences in $L_{S_0}$. It is easy to see that $S \models D_{ss}$. Conversely, it is clear that $D \models S$. Thus by Theorem 1, $D_{ss} \models S$. ■

Clearly, the progressed database at $S_0$ as given by the theorem is also relatively complete. Thus the theorem can be repeatedly applied to progress a relatively complete initial database in response to a sequence of executable actions. Notice that the new database
will include $D_{una}$ and the state independent axioms in $D_{S_0}$; therefore we can use these axioms to simplify $\Phi_F(\vec{x}, \alpha, S_0)[S_0]$.

**Example 5.1** Consider again our educational database example. Suppose now that the initial database $D_{S_0}$ consists of the following axioms:

\begin{align*}
J & \neq S & \neq C & \neq 100 & \neq 200, \\
better(70, 50), \\
prequs(C100, C200), \\
enrolled(st, c, S_0) & \equiv \\
(st = John \land c = C100) \lor (st = Sue \land c = C200), \\
grade(st, c, S_0) & \equiv \\
st = Sue \land c = C100 \land g = 70.
\end{align*}

Clearly $D_{S_0}$ is relatively complete, and $D \models Poss(a, S_0)$, where $\alpha = \text{drop}(John, C100)$. From the axiom for $enrolled$ in $D_{S_0}$, we see that $\Pi_{enrolled}(st, c)$ is the formula:

$$(st = John \land c = C100) \lor (st = Sue \land c = C200).$$

Now from the successor state axiom for $enrolled$ in Example 3.1, we see that $\Phi_{enrolled}(st, c, a, S_0)$, the condition under which $enrolled(st, c, do(a, s))$ will be true, is the formula:

$$a = \text{register}(st, c) \lor (enrolled(st, c, s) \land a \neq \text{drop}(st, c)).$$

Therefore $\Phi_{enrolled}(st, c, \alpha, S_0)[S_0]$ is the formula:

$$\text{drop}(John, C100) = \text{register}(st, c) \lor (st = John \land c = C100) \lor (st = Sue \land c = C200) \land \text{drop}(John, C100) \neq \text{drop}(st, c)).$$

By the unique names axioms in $D_{una}$, this can be simplified to:

$$(st = John \land c = C100) \lor (st = Sue \land c = C200) \land (John \neq st \lor C100 \neq c).$$

By the unique names axioms in $D_{S_0}$, this can be further simplified to:

$$st = Sue \land c = C200.$$

Therefore we obtain the following axiom about $do(\alpha, S_0)$:

$$enrolled(st, c, do(\alpha, S_0)) \equiv st = Sue \land c = C200.$$ 

Similarly, we have:

$$\text{grade}(st, c, g, do(\alpha, S_0)) \equiv st = Sue \land c = C100 \land g = 70.$$

Therefore a progression to $do(\text{drop}(John, C100), S_0)$ is $D_{una}$ together with the following sentences:

$$John \neq S \neq C \neq 100 \neq 200,$$

$$\text{better}(70, 50),$$

$$\text{prequs}(C100, C200),$$

$$enrolled(st, c, do(\alpha, S_0)) \equiv st = Sue \land c = C200,$$

$$\text{grade}(st, c, g, do(\alpha, S_0)) \equiv st = Sue \land c = C100 \land g = 70.$$

### 6 PROGRESSION IN THE CONTEXT FREE CASE

In this section we consider progression wrt context-free action theories. A successor state axiom for $F$ is context-free if it has the form:

$$Poss(a, s) \supset F(\vec{x}, do(a, s)) \equiv$$

$$(\exists \vec{u})(a = A_1(\vec{\xi}_1, \vec{u}) \land E_1) \lor \cdots \lor$$

$$(\exists \vec{v})(a = A_m(\vec{\xi}_m, \vec{v}) \land E_m) \lor$$

$$F(\vec{x}, s) \land \neg(\exists \vec{u})(a = B_1(\vec{\chi}_1, \vec{u}) \land E_{m+1}) \lor \cdots \lor$$

$$\neg(\exists \vec{v})(a = B_n(\vec{\chi}_n, \vec{v}) \land E_{m+n}),$$

where $\vec{\xi}_1$ and $\vec{\chi}_1$ denote sequences of all, or just some (including none) of the $\vec{x}$, the $A$'s and $B$'s are actions, and $E_1, \ldots, E_{m+n}$ are propositional formulas constructed from equality literals over the domain objects, i.e., they are quantifier free, and do not mention terms of sort state and action. The successor state axioms in our educational database are all context-free. So are the following successor state axioms:

$$Poss(a, s) \supset \text{holding}(x, do(a, s)) \equiv a = \text{pick up}(x) \lor$$

$$\text{holding}(x, s) \land a \neq \text{drop}(x) \land (\exists u)a = \text{put}(x, u).$$

$$Poss(a, s) \supset \text{on}(x, y, do(a, s)) \equiv a = \text{move}(x, y) \lor$$

$$\text{on}(x, y, s) \land (\exists z)(a = \text{move}(x, z) \land z \neq y).$$

The following successor state axiom is not context-free:

$$Poss(a, s) \supset \text{dead}(x, do(a, s)) \equiv$$

$$\exists y)a = \text{explode}, \text{bomb,} \text{close}(y) \land \text{close}(x, y) \lor$$

$$\text{dead}(x, s).$$

Given any action terms $A_1(\vec{t}_1)$ and $A_2(\vec{t}_2)$, by the unique names axioms, the equality $A_1(\vec{t}_1) = A_2(\vec{t}_2)$ is either equivalent to $false$ or, when $A_1$ and $A_2$ are the same, equivalent to $t_1 = t_2$.

\footnote{\vec{x} = \vec{y}$ is an abbreviation for $x_1 = y_1 \land \cdots \land x_n = y_n$. Notice that when both $\vec{x}$ and $\vec{y}$ are the empty sequence, $\vec{x} = \vec{y}$ is logically equivalent to $true$. It is equivalent to $false$ when $\vec{x}$ and $\vec{y}$ have different lengths.}
\[ \text{Poss}(\bar{A}(ar{t}), s) \supset \]
\[ \neg E\bar{r} \land (\neg F(\bar{z}, s) \lor E_{-F}) \supset \]
\[ \neg F(\bar{z}, do(\bar{A}(\bar{t}), s)). \] (4)

For instance, by the above successor state axiom for
holding, we have
\[ \text{Poss}(\text{drop}(x), s) \supset \text{holding}(y, do(\text{drop}(x), s)) \equiv \]
\[ \text{holding}(y, s) \land y \neq x. \]

Here \( E_{\text{holding}} \) is \( \text{false} \), and \( E_{-\text{holding}} \) is \( x = y \).

Now assume that:

1. \( \mathcal{D}_{S_0} \) is a set of state independent sentences, and
   sentences of the form
   \[ E \supset \pm F(x_1, \ldots, x_n, S_0), \] (5)
   where \( E \) is a propositional formula constructed from
   equality literals over the domain objects. For example,
   \[ \text{ontable}(x, S_0), \]
   \[ x \neq A \lor \neg \text{ontable}(x, S_0), \]
   \[ x = A \land y = B \supset \text{on}(x, y, S_0), \]
   are all of this form.

2. \( \mathcal{D}_{S_0} \) is coherent in the sense that for every fluent
   \( F \), whenever \((\forall \bar{z}).E_1 \supset F(\bar{z}, S_0)\)
   and \((\forall \bar{z}).E_2 \supset \neg F(\bar{z}, S_0)\)
   are in \( \mathcal{D}_{S_0} \), then
   \[ \{ \varphi \mid \varphi \in \mathcal{D}_{S_0} \text{ is state independent} \} \models (\forall \bar{z}).(E_1 \land E_2). \]

This means that \( \mathcal{D}_{S_0} \) cannot use axioms of the
form (5) to encode state independent sentences:
For any state independent sentence \( \varphi \), \( \mathcal{D}_{S_0} \models \varphi \)
iff
\[ \{ \varphi \mid \varphi \in \mathcal{D}_{S_0} \text{ is state independent} \} \models \varphi. \]

3. \( \mathcal{D}_{S_0} \) is a set of context-free successor state axioms.

4. \( \alpha \) is a ground action term, say \( A(\bar{t}) \).

5. \( \alpha \) is possible initially: \( \mathcal{D} \models \text{Poss}(\alpha, S_0) \).

For example, our educational database in Example 3.1
with the following initial database:

\[ \text{Sue} \neq \text{John} \neq C100 \neq C200 \neq 50, \]
\[ \text{st} = \text{Sue} \land c = C100 \supset \text{enrolled}(\text{st}, c, S_0), \]
\[ \text{st} = \text{Sue} \land c = C200 \supset \neg \text{enrolled}(\text{st}, c, S_0), \]
\[ \text{st} = \text{Sue} \land c = C100 \land g = 50 \supset \text{grade}(\text{st}, c, g, S_0), \]

satisfies the above conditions for \( \alpha = \text{drop}(\text{Sue}, C100) \).

To compute \( \mathcal{D}_{S_0} \), use Theorem 1 to construct a set \( S \),
initially empty, of sentences as follows:

1. If \( \varphi \in \mathcal{D}_{S_0} \) is state independent, then \( \varphi \in S \).

2. For any fluent \( F \), by (3) and (4), the coherence
   assumption, and the assumption that \( \mathcal{D} \models \text{Poss}(\alpha, S_0) \),
   add to \( S \) the sentences
   \[ E\bar{r} \supset F(\bar{z}, do(\alpha, S_0)), \]
   \[ E_{-F} \supset \neg F(\bar{z}, do(\alpha, S_0)). \]

3. For any fluent \( F \), if \((\forall \bar{z}).E \supset F(\bar{z}, S_0)\) is in
   \( \mathcal{D}_{S_0} \), then, by (3) and the assumption that \( \mathcal{D} \models \text{Poss}(\alpha, S_0) \),
   add to \( S \) the sentence
   \[ E \land \neg E_{-F} \supset \neg F(\bar{z}, do(\alpha, S_0)). \]

4. For any fluent \( F \), if \((\forall \bar{z}).E \supset \neg F(\bar{z}, S_0)\) is in
   \( \mathcal{D}_{S_0} \), then, by (4) and the assumption that \( \mathcal{D} \models \text{Poss}(\alpha, S_0) \),
   add to \( S \) the sentence
   \[ \neg E\bar{r} \land E_{-F} \supset \neg F(\bar{z}, do(\alpha, S_0)). \]

For example, consider again our educational database
with the above initial database, and
\( \alpha = \text{drop}(\text{Sue}, C100) \),
we have
\[ \text{Poss}(\alpha, s) \supset \text{enrolled}(\text{st}, c, do(\alpha, s)) \equiv \]
\[ \text{enrolled}(\text{st}, c, s) \land \neg (\text{st} = \text{Sue} \land c = 51), \]
\[ \text{Poss}(\alpha, s) \supset \text{grade}(\text{st}, c, g, do(\alpha, s)) \equiv \]
\[ \text{grade}(\text{st}, c, g). \]

Thus \( E_{\text{enrolled}} \) is \( \text{false} \), \( E_{\text{enrolled}} \) is
\[ \text{st} = \text{Sue} \land c = 100, \]
and \( E_{\text{grade}} \) and \( E_{\text{grade}} \) are both \( \text{false} \). Then the
above procedure will give us the following set \( S \):

\[ \text{John} \neq \text{Sue} \neq \text{C100} \neq \text{C200} \neq 50, \]
\[ (\text{st} = \text{Sue} \land c = \text{C100}) \supset \neg \text{enrolled}(\text{st}, c, \text{S0}), \]
\[ (\text{st} = \text{Sue} \land c = \text{C200}) \supset \neg \text{enrolled}(\text{st}, c, \text{S0}), \]
\[ (\text{st} = \text{Sue} \land c = \text{C100} \land g = 50 \supset \text{grade}(\text{st}, c, g, \text{S0})). \]

As we show in the following theorem, together with \( \mathcal{D}_{\text{una}} \),
this is a progression of \( \mathcal{D}_{S_0} \) to \( S_\alpha \).

**Theorem 4** Under the aforesaid assumptions,
\( S \cup \mathcal{D}_{\text{una}} \) is a progression of \( \mathcal{D}_{S_0} \) to \( S_\alpha \).

**Proof:** It is clear that \( \mathcal{D} \models S \cup \mathcal{D}_{\text{una}} \), and \( S \) is a set
of sentences in \( S_{S_0} \). Therefore by Theorem 1, \( \mathcal{D}_{S_0} \models \]
\( S \cup \mathcal{D}_{\text{una}} \). To prove the converse, we show that for any
model \( M \) of \( S \cup \mathcal{D}_{\text{una}} \), there is a model \( M' \) of \( \mathcal{D} \)
such that \( M \models S_\alpha \). Suppose now that \( M \) is a model of
\( S \cup \mathcal{D}_{\text{una}} \). We construct \( M' \) as follows:

1. \( M' \) and \( M \) have the same domains for sorts \( \text{action} \)
   and \( \text{object} \), and interpret all state independent
   predicates and functions the same.

2. For each fluent \( F \), \( M' \) interprets it on \( S_0 \) as follows:

   a. \[ \text{Poss}(\alpha, s) \supset \text{enrolled}(\text{st}, c, do(\alpha, s)) \equiv \]
   \[ \text{enrolled}(\text{st}, c, s) \land \neg (\text{st} = \text{Sue} \land c = 51), \]
   \[ \text{Poss}(\alpha, s) \supset \text{grade}(\text{st}, c, g, do(\alpha, s)) \equiv \]
   \[ \text{grade}(\text{st}, c, g). \]
(a) For every variable assignment \( \sigma \), if \((\forall \bar{x})E \supset F(\bar{x}, S_0)\) is in \( D_{S\alpha} \), and \( M, \sigma \models E \) (thus \( M', \sigma \models E \) as well), then \( M', \sigma \models F(\bar{x}, S_0) \).

(b) Similarly, for every variable assignment, if \((\forall \bar{x})E \supset \neg F(\bar{x}, S_0)\) is in \( D_{S\alpha} \), and \( M, \sigma \models E \) (thus \( M', \sigma \models E \) as well), then \( M', \sigma \models \neg F(\bar{x}, S_0) \).

(c) For every variable assignment \( \sigma \), if \( F(\bar{x}, S_0) \) has not been assigned a truth value by one of the above two steps, then \( M', \sigma \models F(\bar{x}, S_0) \)

Notice that by the coherence assumption for \( D_{S\alpha} \), our construction is well-defined.

3. \( M' \models \Sigma \). This can be done according to Proposition 3.1.

Clearly \( M' \models D \). We show now that \( M \sim_{S\alpha} M' \). For any fluent \( F \), suppose the successor state axiom for it is

\[
\text{Poss}(\alpha, s) \supset F(\bar{x}, do(\alpha, s)) \equiv E_F \lor (F(\bar{x}, s) \land \neg E_F).
\]

Given a variable assignment \( \sigma \), suppose \( M', \sigma \models F(\bar{x}, do(\alpha, S_0)) \). Since \( \mathcal{D} \models \text{Poss}(\alpha, S_0) \), by the above successor state axiom, there are two cases:

1. \( M', \sigma \models E_F \). This implies \( M, \sigma \models E_F \). Now since \( E_F \supset F(\bar{x}, do(\alpha, S_0)) \) is in \( \mathcal{S} \), and \( M \) is a model of \( S \), thus \( M, \sigma \models F(\bar{x}, do(\alpha, S_0)) \) as well.

2. \( M', \sigma \models \neg E_F \land F(\bar{x}, S_0) \land \neg E_{-F} \). From \( M', \sigma \models F(\bar{x}, S_0) \), by our construction, either \( M, \sigma \models F(\bar{x}, do(\alpha, S_0)) \), or there is a sentence \( E \supset F(\bar{x}, S_0) \) in \( D_{S\alpha} \) such that \( M, \sigma \models E \). Suppose it is the latter. Then by our construction of \( \mathcal{S} \), it contains \( E \land \neg E_{-F} \supset F(\bar{x}, do(\alpha, S_0)) \). Thus \( M, \sigma \models F(\bar{x}, do(\alpha, S_0)) \) as well.

Similarly, if \( M', \sigma \models \neg F(\bar{x}, do(\alpha, S_0)) \), then \( M, \sigma \models \neg F(\bar{x}, do(\alpha, S_0)) \) as well. Therefore \( M \sim_{S\alpha} M' \).

We have some remarks:

1. The new database \( \mathcal{S} \) has the same form as \( D_{S\alpha} \), so this process can be iterated.

2. The generation of \( \mathcal{S} \) is very fast, and the size of \( \mathcal{S} \) is bounded by the sum of the size of \( D_{S\alpha} \) and the twice the number of fluents.

3. The \( E \)'s in context free successor state axioms can be any state independent formulas. Thus a limited context dependency can be handled.

We emphasize that the results of this section depend on the fact that the initial database has a certain specific form. In fact, a result by Pednault [9] shows that for context-free actions and arbitrary \( D_{S\alpha} \), progression is not always guaranteed to yield finite first-order theories.

7 SUMMARY

1. We have argued the need for progressing a database.

2. We have defined a formal notion of progression, and showed that in general, to capture it we need second-order logic.

3. We have studied two special cases for which progression is first order definable, and which can be done efficiently.

4. Although we don’t discuss them here, there are other cases for which progression can be done in first order logic. One such case concerns actions with finitary effects, i.e., for any fluent, the action changes the truth values of the fluent at only a finite number of instances.

5. The complexity of progression depends on both the form of the initial database, and the form of the action theory. A relatively complete initial database can be progressed efficiently wrt any successor state axioms. On the other hand, even for context free successor state axioms, progression is not guaranteed to yield finite first-order theories.

6. In a companion paper (Lin and Reiter [7]) we explore the consequences of our results on progression for the semantics of STRIPS-like systems. Ever since STRIPS was first introduced (Fikes and Nilsson [4]), its logical semantics has been problematic. There have been many proposals in the literature (e.g., Lifschitz [6], Pednault [11], Bacchus and Yang [2]). These all have in common a reliance on meta-theoretic operations on logical theories in order to capture the add and delete lists of STRIPS operators, but it has never been clear exactly what these operations correspond to declaratively, especially when they are applied to logically incomplete theories. In the companion to this paper, we provide a semantics for STRIPS-like systems in terms of basic theories of actions in the situation calculus. On our view, STRIPS is a mechanism for computing the progression of an initial situation calculus database under the effects of an action. We illustrate this idea by specifying two different versions of STRIPS in the situation calculus as well as a generalization of STRIPS that appeals to relational database theory.
Acknowledgements

For their generous advice and feedback, we wish to thank the other members of the University of Toronto Cognitive Robotics Group: Yves Lespérance, Hector Levesque, Bill Millar, Daniel Marcu, and Richard Scheir. This research was funded by the Government of Canada National Sciences and Engineering Research Council, and the Institute for Robotics and Intelligent Systems.

References


