ELSEVIER

Artificial Intelligence 92 (1997) 131–167

Artificial Intelligence

How to progress a database ★

Fangzhen Lin [1], Ray Reiter *

Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S 3H5

Received March 1996; revised October 1996

Abstract

One way to think about a STRIPS operator 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 α, were done in some world, represented by the STRIPS database $\mathcal{D}_0$. To find out, simply perform the operator α on $\mathcal{D}_0$ (by applying α's elementary *add* and *delete* revision operators to $\mathcal{D}_0$). We describe this process as *progressing the database* $\mathcal{D}_0$ in response to the action α.

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 (1991)). 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.

Finally, we relate these results about progression to STRIPS-like systems by providing a semantics for such systems in terms of a purely declarative situation calculus axiomatization for actions and their effects. On our view, STRIPS operators provide a *mechanism* for computing the progression of an initial situation calculus database under the effects of an action. We illustrate this idea by describing two different STRIPS mechanisms, and proving their correctness with respect to their situation calculus specifications. © 1997 Elsevier Science B.V.

Keywords: Situation calculus; Theories of actions; Regression; Progression; STRIPS; Strongest postconditions

★ This paper revises, and combines, results that first appeared in F. Lin and R. Reiter's "How to progress a database (and why) I. Logical foundations" [12] and "How to progress a database II. The STRIPS connection" [15].

* Corresponding author. E-mail: reiter@ai.toronto.edu. Fellow of the Canadian Institute for Advanced Research.

[1] E-mail: fl@ai.toronto.edu.

PII S0004-3702(96)00044-6

1. Introduction

One way to think about STRIPS operators 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 α, were done in some world, represented by the STRIPS database $\mathcal{D}_0$. To find out, simply perform the operator α on $\mathcal{D}_0$ (by applying α's elementary *add* and *delete* revision operators to $\mathcal{D}_0$). We describe this process as *progressing the database* $\mathcal{D}_0$ in response to the action α (cf. Rosenschein [25] and Pednault [16]). The resulting database describes the effects of the action on the world represented by the initial database.[2] 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 [21], Lin and Reiter [13]). 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.

Finally, we relate these results about progression to STRIPS-like systems by providing a semantics for such systems in terms of a purely declarative situation calculus axiomatization for actions and their effects. On our view, a STRIPS operator is a *mechanism* for computing the progression of an initial situation calculus database under the effects of an action. We illustrate this idea by describing two different STRIPS mechanisms, and proving their correctness with respect to their situation calculus specifications.

The need to progress a database arises for us in a robotics setting. In our approach to controlling a robot [8, 10], 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 [28], Pednault [17], and Reiter [21]) 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:

1. After the robot has been functioning for a long period, the sequence A, consisting of all the actions it has performed since the initial situation, has become extremely long, and regressing over such a sequence becomes computationally expensive.

[2] This is also the way that database practitioners think about database updates (Abiteboul [1]). In fact, the STRIPS action and the 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. For more on the database perspective, see Reiter [23].

2. Similarly, after a long while, the initial 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 [26]) 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 $\mathcal{L}$ of the situation calculus is first order, many sorted, with sorts *situation* for situations, *action* for actions, and *object* for everything else. It has the following domain independent predicates and functions: a constant S_0 of sort *situation* denoting the initial situation; a binary function $do(a, s)$ denoting the situation resulting from performing the action a in the situation s; a binary predicate $Poss(a, s)$ meaning that the action a is possible (executable) in situation s; and a binary predicate $<$: *situation*$\times$ *situation*. $s < s'$ means that s' can be reached from s by a sequence of executable actions. We assume a finite number of *situation independent* predicates with arity *object*^{n}, $n \geqslant 0$, a finite number of *situation independent* functions with arity *object*$^n \rightarrow$ *object*, $n \geqslant 0$, and a finite number of *fluents* which are predicate symbols of arity *object*$^n \times$ *situation*, $n \geqslant 0$. We denote by $\mathcal{L}^2$ the second-order extension of $\mathcal{L}$. Our foundational axioms for the situation calculus will be in $\mathcal{L}^2$ (Lin and Reiter [13]), because we need induction on situations (Reiter [22]).

Often, we must restrict the situation calculus to a particular situation. For example, the initial database is a finite set of sentences in $\mathcal{L}$ that do not mention any situation terms except S_0, and do not mention *Poss* and $<$. For this purpose, for any situation term st, we define $\mathcal{L}_{st}$ to be the subset of $\mathcal{L}$ that does not mention any other situation terms except st, does not quantify over situation variables, and does not mention *Poss* or $<$. Formally, it is the smallest set satisfying:

1. $\varphi \in \mathcal{L}_{st}$ provided $\varphi \in \mathcal{L}$ does not mention any situation term.
2. $F(t_1, \ldots, t_n, st) \in \mathcal{L}_{st}$ provided F is a fluent of the right arity, and $t_1, \ldots, t_n$ are terms of the right sort.
3. If φ and φ' are in $\mathcal{L}_{st}$, so are $\neg\varphi$, $\varphi \vee \varphi'$, $\varphi \wedge \varphi'$, $\varphi \supset \varphi'$, $\varphi \equiv \varphi'$, $(\forall x)\varphi$, $(\exists x)\varphi$, $(\forall a)\varphi$, and $(\exists a)\varphi$, 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 $\mathcal{L}_{do(a,S_0)}$. This may seem odd when we want sentences in $\mathcal{L}_{st}$ to be propositions about situation *st*. Fortunately, we shall use $\mathcal{L}_{st}$ only when *st* is either a ground term or a simple variable of sort *situation*.

We shall use $\mathcal{L}^2_{st}$ to denote the second-order extension of $\mathcal{L}_{st}$ by predicate variables of arity *object*^{n}, $n \geqslant 0$. So the second-order sentence $(\exists p)(\forall x).p(x) \equiv F(x, S_0)$ is in $\mathcal{L}^2_{S_0}$, 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* × *situation*. Formally, $\mathcal{L}^2_{st}$ is the smallest set satisfying:

1. Every formula in $\mathcal{L}_{st}$ is also in $\mathcal{L}^2_{st}$.
2. $p(t_1, \ldots, t_n) \in \mathcal{L}^2_{st}$ provided p is a predicate variable of arity *object*^{n}, $n \geqslant 0$, and $t_1, \ldots, t_n$ are terms of sort *object*.
3. If φ and φ' are in $\mathcal{L}^2_{st}$, so are $\neg\varphi$, $\varphi \vee \varphi'$, $\varphi \wedge \varphi'$, $\varphi \supset \varphi'$, $\varphi \equiv \varphi'$, $(\forall x)\varphi$, $(\exists x)\varphi$, $(\forall a)\varphi$, $(\exists a)\varphi$, $(\forall p)\varphi$, 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}, $n \geqslant 0$.

3. Basic action theories

We assume given a *basic action theory* $\mathcal{D}$, having the following form (cf. Reiter [23] and Lin and Reiter [13]):[3]

$$\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0},$$

where:

- Σ, given below, is the set of the foundational axioms for situations.
- $\mathcal{D}_{ss}$ is a set of successor state axioms of the form:[4]

$$Poss(a, s) \supset [F(\boldsymbol{x}, do(a, s)) \equiv \Phi_F(\boldsymbol{x}, a, s)], \tag{1}$$

  where F is a fluent, and $\Phi_F(\boldsymbol{x}, a, s)$ is in $\mathcal{L}_s$. Informally, a successor state axiom about F specifies the truth values of F in the successor situation $do(a, s)$ in terms of properties of the current situation s.
- $\mathcal{D}_{ap}$ is a set of action precondition axioms of the form:

$$Poss(A(\boldsymbol{x}), s) \equiv \Psi_A(\boldsymbol{x}, s),$$

[3] We emphasize that a basic action theory is monotonic; we are not presenting here any nonmonotonic approaches to solving the frame, ramification or qualification problems. An approach to such problems, using a nonmonotonic logic, is described in [13]. This sometimes allows one to derive a (monotonic) basic action theory from state constraints, but it is important to note that the resulting theory does not contain the original state constraints; it does, however, include the same "information content" as was present in the original state constraints. Accordingly, the basic action theories of this paper do *not* include state constraints.

[4] In the following, unless otherwise stated, all free variables in a formula are assumed to be prenex universally quantified. Variables will always begin with a lower case Roman character; constants will always begin with upper case.

where A is an action, and $\Psi_A(\boldsymbol{x}, s)$ is in $\mathcal{L}_s$. An action precondition axiom specifies necessary and sufficient conditions under which an action can be (physically) performed.

- $\mathcal{D}_{una}$ is the set of unique names axioms for actions: for any two different actions $A(\boldsymbol{x})$ and $A'(\boldsymbol{y})$, we have

$$A(\boldsymbol{x}) \neq A'(\boldsymbol{y}),$$

and for any action $A(x_1, \ldots, x_n)$, we have

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

- $\mathcal{D}_{S_0}$, the initial database, is a finite set of first-order sentences in $\mathcal{L}_{S_0}$.

The following is an example of a basic action theory. Notice that Σ, the foundational axioms for the situation calculus given below, will be independent of any domain, and $\mathcal{D}_{una}$ can be automatically generated once the language has been specified, so to define a basic action theory, one need only specify the successor state, action precondition, and initial situation axioms.

Example 3.1. An educational database (Reiter [23]). There are two fluents:

- *enrolled*(*stu*, *course*, *s*): student *stu* is enrolled in course *course* in situation *s*.
- *grade*(*stu*, *course*, *grade*, *s*): the grade of *stu* in *course* is *grade* in situation *s*.

There are two situation independent predicates:

- *prereq*(*pre*, *course*): *pre* is a prerequisite course for course *course*.
- *better*(*grade*1, *grade*2): grade *grade*1 is better than grade *grade*2.

There are three database transactions:

- *register*(*stu*, *course*): register the student *stu* in course *course*, with precondition that the student has satisfied all of the prerequisites for *course* by obtaining a grade better than 50 in each prerequisite.
- *change*(*stu*, *course*, *grade*): change the grade of the student *stu* in course *course* to *grade*. This action can always be performed.
- *drop*(*stu*, *course*): drop the student *stu* from course *course*, with precondition that the student is currently enrolled in *course*.

This setting can be axiomatized as follows.

$\mathcal{D}_{ss}$ consists of the following successor state axioms:

$$\begin{aligned} &Poss(a, s) \supset [enrolled(stu, c, do(a, s)) \equiv \\ &\qquad a = register(stu, c) \vee enrolled(stu, c, s) \wedge a \neq drop(stu, c)], \end{aligned}$$

$$\begin{aligned} &Poss(a, s) \supset [grade(stu, c, g, do(a, s)) \equiv \\ &\qquad a = change(stu, c, g) \vee \\ &\qquad grade(stu, c, g, s) \wedge \neg(\exists g')(g \neq g' \wedge a = change(stu, c, g'))]. \end{aligned}$$

$\mathcal{D}_{ap}$ consists of the following action precondition axioms:

$$\begin{aligned} &Poss(register(stu, c), s) \equiv \\ &(\forall pr).prereq(pr, c) \supset (\exists g)(grade(stu, pr, g, s) \wedge better(g, 50)), \end{aligned}$$

$$Poss(change(stu, c, g), s) \equiv True,$$

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

$\mathcal{D}_{S_0}$, the initial database, can be any finite set of axioms about the initial situation, or axioms which mention no situation, for example, the following:[5]

$$John \neq Sue \neq C100 \neq C200,$$

$$prereq(C100, C200),$$

$$enrolled(Sue, C100, S_0),$$

$$enrolled(John, C100, S_0) \vee enrolled(John, C200, S_0).$$

We shall now present our domain independent foundational axioms Σ which specify the structure of situations. Informally, Σ stipulates that the space of situations is a tree with S_0 at the root and with actions the only way of generating new nodes (situations). Formally, Σ consists of the following axioms:

$$S_0 \neq do(a, s), \tag{2}$$

$$do(a_1, s_1) = do(a_2, s_2) \supset (a_1 = a_2 \wedge s_1 = s_2), \tag{3}$$

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

$$\neg s < S_0, \tag{5}$$

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

Notice the similarity between Σ 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 a domain closure axiom which says that every situation must be obtained by repeatedly applying *do* to S_0.[6] The last two axioms define $<$ inductively.

Σ are the only axioms in a basic action theory about the structure of situations. It is often needed 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, $\mathcal{D}$ has exactly the same effect as $\mathcal{D} - \Sigma$: for any formula $\varphi(s)$ in $\mathcal{L}_s$, and any sequence A of ground action terms,

$$\mathcal{D} \models \varphi(do(A, S_0))$$

iff

$$\mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models \varphi(do(A, S_0)).$$

[5] In general, $t_1 \neq t_2 \neq \cdots \neq t_n$ stands for the $n(n-1)$ inequalities: $t_1 \neq t_2 \wedge \cdots \wedge t_1 \neq t_n \wedge \cdots \wedge t_{n-1} \neq t_n$.

[6] For a discussion of the use of induction in the situation calculus, see Reiter [22].

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

Proposition 3.2. *Given any model M^- of $\mathcal{D} - \Sigma$, there is a model M of $\mathcal{D}$ such that:*

1. *M^- and M have the same domains for sorts action and object, and interpret all situation independent predicates and functions the same;*
2. *for any sequence $\mathbf{A}$ of ground action terms, any fluent F, and any variable assignment ν:*[7]

$$M, \nu \models F(\boldsymbol{x}, do(\mathbf{A}, S_0)) \quad \textit{iff} \quad M^-, \nu \models F(\boldsymbol{x}, do(\mathbf{A}, S_0)).$$

Proof. We begin with the observation that no sentence of $\mathcal{D} - \Sigma = \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ mentions an equality atom whose arguments are of sort *situation*, and (2) and (3) are unique names axioms about situations. It follows from this that if M^- is a model of $\mathcal{D} - \Sigma$, then there is a model M of $\mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \{(2), (3)\}$ such that the conditions of the proposition are satisfied. So without loss of generality, we can assume that M^- is a model of $\mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \{(2), (3)\}$.

In the following, we use ξ^M for the denotation of the symbol ξ in an interpretation M. Given M^-, construct a structure M as follows. First, let M's domains be the same as that of M^- for sorts *action* and *object*. Next, let the domain dom^M_{sit} for the sort *situation* be the smallest subset of the *situation* domain of M^- such that:

1. $S_0^{M^-} \in dom^M_{sit}$.
2. If $\sigma \in dom^M_{sit}$ and if α is an element of the *action* domain of M^-, then $do^{M^-}(\alpha, \sigma) \in dom^M_{sit}$.

So, M has exactly the same domain of sorts *action* and *object* as does M^-, and its situation domain is a subset of that of M^-.

To complete the specification of M, we describe how it interprets function and predicate symbols.

1. M interprets all situation independent function and predicate symbols (including the equality predicate) exactly as does M^-.
2. M interprets the equality predicate over situation terms of dom^M_{sit} exactly as does M^-.
3. M interprets *do*, *Poss*, and fluents over M's domain exactly as does M^- over this domain.
4. Finally, we specify how M interprets the $<$ relation on situations. $<^M$ is the smallest set with the properties:
   (a) If $\sigma \in dom^M_{sit}$ and $(\alpha, \sigma) \in Poss^M$, then $(\sigma, do(\alpha, \sigma)) \in <^M$.
   (b) If $\sigma, \sigma' \sigma'' \in dom^M_{sit}$ and $(\sigma, \sigma') \in <^M$ and $(\sigma', \sigma'') \in <^M$, then $(\sigma, \sigma'') \in <^M$.

We prove that M is a model of $\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$, from which the proposition follows.

1. To begin, consider any sentence of $\mathcal{L}$ of the form $(\forall s)\phi$, where ϕ does not mention $<$, where ϕ does not mention an equality atom with situation arguments, and where ϕ does not quantify over situations. Then whenever M^- is a model

[7] $M, \nu \models \varphi$ means that M satisfies φ under the variable assignment ν.

of $(\forall s)\phi$, so is M. This is so because M and M^- interpret *do*, *Poss*, fluents and situation independent function and predicate symbols identically over the elements of M's domain, and M's domain for sort *situation* is a subset of that for M^-. Since every sentence of $\mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is of the form $(\forall s)\phi$, or is situation independent, it follows that M is a model for $\mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$, since M^- is.

2. It remains to prove that M is a model of Σ.
   (a) M satisfies the unique names axioms (2) and (3) for situations because M^- does.
   (b) M satisfies the induction axiom (4), because this says that M's situation domain is the smallest set containing S_0^M which is closed under the function do^M, and this is true of M's situation domain.
   (c) Finally, it is not hard to see that $<^M$, as defined in 4 above, satisfies the axioms (5) and (6) of Σ.

The conditions of the proposition follow from the properties of M. □

4. Formal foundations

Let α be a ground simple action, e.g. $enrolled(Sue, C100)$, and let S_α denote the situation term $do(\alpha, S_0)$. A progression $\mathcal{D}_{S_\alpha}$ of $\mathcal{D}_{S_0}$ in response to α should have the following properties:

1. $\mathcal{D}_{S_\alpha}$ is a set of sentences about situation S_α only, i.e., in $\mathcal{L}_{S_\alpha}$ or in $\mathcal{L}^2_{S_\alpha}$.
2. For all queries about the future of S_α, $\mathcal{D}$ is equivalent (in a suitable formal sense) to

$$\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha}.$$

In other words, $\mathcal{D}_{S_\alpha}$ acts like the new initial database with respect to all possible future evolutions of the theory following the "performance" of the action α.

Semantically, the models of $\mathcal{D}_{S_\alpha}$ should include those of $\mathcal{D}$. But since $\mathcal{D}_{S_\alpha}$ is a set of sentences about S_α only, any structure that is "isomorphic at S_α" to a model of $\mathcal{D}$ should also be a model of $\mathcal{D}_{S_\alpha}$, and these should be all the models of $\mathcal{D}_{S_\alpha}$. Another way of thinking about progressing $\mathcal{D}_{S_0}$ to $\mathcal{D}_{S_\alpha}$ is that we want $\mathcal{D}$ to "forget about" what is true of the initial situation and all those situations that are reachable from S_0 but not from S_α. This means we are interested in those models of $\mathcal{D}_{S_\alpha}$ and of $\mathcal{D}$ which "don't care" about what is true in $\mathcal{D}_{S_0}$. To make these intuitions precise, we first need to define what we mean by "isomorphic at S_α". To that end, we 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 $M' \sim_{S_\alpha} M$, ("M and M' are isomorphic at S_α") iff the following two conditions hold:

1. M' and M interpret all predicate and function symbols which do not take any arguments of sort *situation* identically.
2. M and M' agree on all fluents at S_α: for every predicate fluent F, and every variable assignment σ,

$$M', \sigma \models F(\boldsymbol{x}, do(\alpha, S_0)) \quad \text{iff} \quad M, \sigma \models F(\boldsymbol{x}, do(\alpha, S_0)).$$

Clearly, $\sim_{S_\alpha}$ is an equivalence relation. If $M' \sim_{S_\alpha} M$, then M' agrees with M on S_α on fluents and situation independent predicates and functions, but is free to vary its interpretation of everything else on all other situations. In particular, they can interpret *Poss* and *do* differently. We have the following simple lemma.

Lemma 4.1. *If $M \sim_{S_\alpha} M'$, then for any formula φ in $\mathcal{L}^2_{S_\alpha}$, and any variable assignment σ, $M, \sigma \models \varphi$ iff $M', \sigma \models \varphi$.*

We can now make the following definition:

Definition 4.2. A set of sentences $\mathcal{D}_{S_\alpha}$ in $\mathcal{L}^2_{S_\alpha}$ is a *progression* of the initial database $\mathcal{D}_{S_0}$ to S_α (with respect to $\mathcal{D}$) iff for any structure M, M is a model of $\mathcal{D}_{S_\alpha}$ iff there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$.

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 4.3. *Let $\mathcal{D}_{S_\alpha}$ be a progression of the initial database to S_α. Then every model of $\mathcal{D}$ is a model of $\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha}$.*

Proposition 4.4. *Let $\mathcal{D}_{S_\alpha}$ be a progression of the initial database to S_α. Then for every model M of*

$$\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha},$$

there exists a model M' of $\mathcal{D}$ such that:

1. *M' and M interpret all situation independent predicate and function symbols identically.*
2. *For every variable assignment σ, and every predicate fluent F,*

$$M', \sigma \models S_\alpha \leqslant s \wedge F(\boldsymbol{x}, s) \quad \textit{iff} \quad M, \sigma \models S_\alpha \leqslant s \wedge F(\boldsymbol{x}, s).$$

Proof. Let M be a model of

$$\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha}.$$

Since M is a model of $\mathcal{D}_{S_\alpha}$, there is a model M' of

$$\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$$

such that $M' \sim_{S_\alpha} M$. It can be easily seen that M' has the desired properties. □

From these two propositions, we conclude that $\mathcal{D}$ and $\Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_\alpha}$ agree on all situations $\geqslant S_\alpha$. So $\mathcal{D}_{S_\alpha}$ really does characterize the result of progressing the initial database in response to the action α. Furthermore, the following theorem says that the new database, when it exists, entails the same set of sentences in $\mathcal{L}^2_{S_\alpha}$ as $\mathcal{D}$:

Theorem 4.5. *Let $\mathcal{D}_{S_\alpha}$ be a progression of the initial database to S_α. For any sentence $\varphi \in \mathcal{L}^2_{S_\alpha}$, $\mathcal{D}_{S_\alpha} \models \varphi$ iff $\mathcal{D} \models \varphi$.*

Proof. If $\mathcal{D} \models \varphi$, then by Lemma 4.1, we have $\mathcal{D}_{S_\alpha} \models \varphi$. If $\mathcal{D}_{S_\alpha} \models \varphi$, then $\mathcal{D} \models \varphi$ by Proposition 4.3. □

This theorem informs us that $\mathcal{D}_{S_\alpha}$ is a *strongest postcondition* (cf. Pednault [16], Dijkstra and Scholten [3], and others) of the precondition $\mathcal{D}_{S_0}$ with respect to the action α.

Pednault [16], by defining progression as the set of first-order sentences in $\mathcal{L}_{S_\alpha}$ that are entailed by $\mathcal{D}$, shows that his definition of progression cannot in general be a finite set of first-order sentences in $\mathcal{L}_{S_\alpha}$. By Theorem 4.5, this result applies to our definition as well. In the next section, we shall extend this result, and show that $\mathcal{D}_{S_\alpha}$ need not even be a set of first-order sentences in $\mathcal{L}_{S_\alpha}$.

4.1. Progression is not always first-order definable

At first glance, the fact that progression cannot always be expressed in first-order logic may seem obvious in light of Theorem 4.5, 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 exactly the same set of sentences in $\mathcal{L}_{S_\alpha}$.
3. There is no model M' of $\mathcal{D}$ such that $M' \sim_{S_\alpha} M_2$.

It will then follow from our definition that for $\mathcal{D}$, the progression of the initial database to S_α cannot be in $\mathcal{L}_{S_\alpha}$. Suppose otherwise, then by property 1, $M_1 \models \mathcal{D}_{S_\alpha}$; by property 2 and the assumption that $\mathcal{D}_{S_\alpha}$ is a set of sentences in $\mathcal{L}_{S_\alpha}$, we have $M_2 \models \mathcal{D}_{S_\alpha}$ as well, but this contradicts property 3 and our definition of progression.

We now proceed to construct such a basic action theory, and the two associated structures. 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 *succ*:

$$\mathcal{D}_{una} = \emptyset.$$

$$\mathcal{D}_{S_0} = \{(\forall x).x \neq 0 \supset (\exists y)x = succ(y)\}.$$

$$\mathcal{D}_{ap} = \{(\forall s).Poss(A, s) \equiv True\}.$$

$\mathcal{D}_{ss}$ consists of the following pair of axioms:

$$Poss(a, s) \supset [F_1(do(a, s)) \equiv (\exists x)\neg F_2(x, s)],$$

$$Poss(a,s) \supset \{F_2(x,do(a,s)) \equiv$$
$$x = 0 \wedge F_2(0,s) \vee$$
$$x \neq 0 \wedge F_2(x,s) \equiv (\exists y)[x = succ(y) \wedge F_2(y,s)]\}.$$

For an intuitive reading of the successor state axioms, think of the constant symbol 0 as the number 0, and the unary function *succ* as the successor function. Then for any x, $F_2(x,do(a,s))$ holds iff either $x = 0$ and $F_2(0,s)$ holds, or $F_2(x,s)$ and $F_2(predecessor(x),s)$ have the same truth value. The purpose of F_1 is to keep track of the truth values of F_2 in the previous situation.

We now proceed to construct the two models M_1 and M_2 that satisfy the above-mentioned three properties. We first construct M_2 which is a structure such that:[8]

1. M_2 is a standard model of arithmetic with respect to sort *object*. Thus the domain for *object* in M_2 is the set of nonnegative numbers, 0 is mapped to the number 0, and *succ* is mapped to the successor function.
2. $M_2 \models F_1(do(A,S_0)) \wedge (\forall x)F_2(x,do(A,S_0))$.

We claim that there cannot be a model M' of $\mathcal{D}$ such that $M_2 \sim_{S_A} M'$. Suppose otherwise. Then M' also satisfies properties 1 and 2 above. Since $M' \models \mathcal{D}_{ss}$, and $M' \models F_1(do(A,S_0))$, we have

$$M' \models (\exists x)\neg F_2(x,S_0).$$

Similarly, since $M' \models (\forall x)F_2(x,do(A,S_0))$, by the successor state axiom for F_2, we have $M' \models F_2(0,S_0) \wedge F_2(succ(0),S_0) \wedge \cdots$. Thus $M' \models (\forall x)F_2(x,S_0)$, a contradiction. Therefore there is no model M' of $\mathcal{D}$ such that $M_2 \sim_{S_n} M'$.

We now construct a model M_1 of $\mathcal{D}$ such that for any sentence φ in $\mathcal{L}_{S_A}$, $M_1 \models \varphi$ iff $M_2 \models \varphi$. The construction of M_1 is in two steps. First, by using Skolem's theorem for number theory, we construct a structure M^* which satisfies exactly the same set of sentences in $\mathcal{L}_{S_A}$ as M_2. We then revise M^* into a model of $\mathcal{D}$ in such a way that the above property continues to hold, thus obtaining the desired model M_1.

By Skolem's theorem (cf. Kleene [7, p. 326]), there is a first-order structure M^* such that for any sentence φ in $\mathcal{L}_{S_A}$, $M_2 \models \varphi$ iff $M^* \models \varphi$, and $(M_2,0,succ)$ and $(M^*,0,succ)$ are not isomorphic, i.e., M_2 and M^* are not isomorphic on sort *object*. In particular, since

$$M \models F_1(do(A,S_0)) \wedge (\forall x)F_2(x,do(A,S_0)),$$

and $F_1(do(A,S_0)) \wedge (\forall x)F_2(x,do(A,S_0))$ is a sentence in $\mathcal{L}_{S_A}$, we have

$$M^* \models F_1(do(A,S_0)) \wedge (\forall x)F_2(x,do(A,S_0)).$$

[8] We thank one of the referees for suggesting the following picture that may help the reader better understand the successor state axioms and the construction of M_2. Imagine an infinite row of lights labelled $0,1,2,\ldots$ (like floor indicators in an elevator). The lights come on or go off according to the following rules: if the first light is on (off), it stays on (off) forever; any other light comes (stays) on if it and its predecessor were both on or both off together, and goes (stays) off otherwise. Then the only way that all the lights are on in the next step ($(\forall x)F_2(x,do(A,S_0))$ holds) yet there was a light off initially ($(\exists x)\neg F_2(x,S_0)$, i.e., $F_1(do(A,S_0))$ holds) would be that the initially off light be "somewhere else"—a nonstandard number!

Now revise M^* into a structure M_1 such that:

1. M_1 and M^* have the same domains for sorts *action* and *object*, and interpret situation independent predicates and functions the same.
2. $M_1 \models (\forall a, s)Poss(a, s)$.
3. $M_1 \models \Sigma \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$.
4. For the truth values of the fluents on S_0: $M_1 \models F_1(S_0)$, and for the truth values of $F_2(x, S_0)$, we have that for any variable assignment σ:
   (a) If $\sigma(x)$ is a standard number, i.e., there is an $n \geqslant 0$ such that $M_1, \sigma \models x = succ^n(0)$, then $M_1, \sigma \models F_2(x, S_0)$.
   (b) If $\sigma(x)$ is a nonstandard number, i.e., there is no $n \geqslant 0$ such that $M_1, \sigma(x) \models x = succ^n(0)$, then $M_1, \sigma \models \neg F_2(x, S_0)$. Notice that since M^* and M_2 are not isomorphic on sort *object* with respect to Peano arithmetic, there must be a nonstandard number in the domain of M^*, and thus in the domain of M_1.
5. For the truth values of the fluents on $do(A, S_0)$: for any fluent F, and any variable assignment σ, $M_1, \sigma \models F(\boldsymbol{x}, do(A, S_0))$ iff $M^*, \sigma \models F(\boldsymbol{x}, do(A, S_0))$.
6. Inductively, for any variable assignment σ, if

   $$M_1, \sigma \models do(A, S_0) < s,$$

   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

   $$M_1, \sigma \models S_0 < s \wedge \neg do(A, S_0) < s,$$

   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 situation because $M_1 \models (\forall s).S_0 \leqslant s$, which follows from the fact that $M_1 \models (\forall a, s)Poss(a, s)$.

Clearly, $M_1 \sim_{S_A} M^*$. It follows that M_1 and M_2 satisfy the same set of sentences in $\mathcal{L}_{S_A}$. We now show that M_1 satisfies the successor state axioms. By the construction of M_1, we only need to prove that it satisfies the successor state axioms instantiated to S_0 and action A, i.e.,

$$M_1 \models Poss(A, S_0) \supset [F_1(do(A, S_0)) \equiv (\exists x)\neg F_2(x, S_0)],$$

and

$$\begin{aligned} M_1 \models\ & Poss(A, S_0) \supset \\ & (\forall x)\{F_2(x, do(A, S_0)) \equiv \\ & \quad x = 0 \wedge F_2(0, S_0) \vee \\ & \quad x \neq 0 \wedge F_2(x, S_0) \equiv (\exists y)[x = succ(y) \wedge F_2(y, S_0)]\}. \end{aligned}$$

To show the first, we need to prove that $M_1 \models (\exists x)\neg F_2(s, S_0)$. This follows from our construction of M_1 and the existence of nonstandard numbers in the domain of M_1. To show the second, we need to prove that

$$\begin{aligned} M_1 \models\ & (\forall x)\{x = 0 \wedge F_2(0, S_0) \vee \\ & \quad x \neq 0 \wedge F_2(x, S_0) \equiv (\exists y)[x = succ(y) \wedge F_2(y, S_0)]\}. \end{aligned}$$

There are three cases:

1. If $x = 0$, then $F_2(0, S_0)$ follows from our construction.
2. If $x = succ^n(0)$ for some $n > 0$, then both $F_2(succ^n(0), S_0)$ and $F_2(succ^{n-1}(0), S_0)$ hold.
3. If x is a nonstandard number, then $F_2(x, S_0)$ does not hold. Furthermore, for any y such that $x = succ(y)$, y is also a nonstandard number, so $F_2(y, S_0)$ does not hold either. Moreover, by the axiom in $\mathcal{D}_{S_0}$, such a y exists.

Therefore, M_1 satisfies the successor state axioms instantiated to S_0 and A. So $M_1 \models \mathcal{D}_{ss}$. This means that $M_1 \models \mathcal{D}$, and M_1 and M_2 satisfy the same set of sentences in $\mathcal{L}_{S_A}$. Therefore we have constructed two models M_1 and M_2 that satisfy the three conditions in the beginning of this subsection, so the progression to S_A for $\mathcal{D}$ 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 first introduce some notation.

Given a finite set $\mathcal{D}_{ss}$ of successor state axioms, define the *instantiation* of $\mathcal{D}_{ss}$ on an action term at and a situation term st, written $\mathcal{D}_{ss}[at, st]$, to be the sentence:

$$\bigwedge_{F \text{ is a fluent}} Poss(at, st) \supset (\forall \boldsymbol{x}).F(\boldsymbol{x}, do(at, st)) \equiv \Phi_F(\boldsymbol{x}, at, st),$$

where

$$(\forall a, s).Poss(a, s) \supset (\forall \boldsymbol{x})[F(\boldsymbol{x}, do(a, s)) \equiv \Phi_F(\boldsymbol{x}, a, s)]$$

is the successor state axiom for F in $\mathcal{D}_{ss}$.

Given a formula φ in $\mathcal{L}^2$, the *lifting* of φ on the situation st, written $\varphi \uparrow st$, is the result of replacing every fluent atom of the form $F(t_1, \ldots, t_n, st)$ by a new predicate variable $p(t_1, \ldots, t_n)$ of arity $object^n$. For example,

$$enrolled(John, C200, S_0) \wedge enrolled(John, C100, S_0) \uparrow S_0$$

is $p(John, C200) \wedge p(John, C100)$.[9]

Lemma 4.6. *The following are some simple properties of lifting:*

1. *If φ is a sentence that does not mention st, then $\varphi \uparrow st$ is φ.*
2. *If φ is a sentence in $\mathcal{L}^2_{st}$, then $\varphi \uparrow st$ is a situation independent sentence.*
3. *If φ does not mention quantifiers over situations, then $\varphi \models (\exists p_1, \ldots, p_k)\varphi \uparrow st$, where $p_1, \ldots, p_k$ are the new predicate variables introduced during the lifting.*

With the above notation in hand, we can describe a procedure for computing the progression of the initial database $\mathcal{D}_{S_0}$ in response to the action α:

[9] Lifting as we have defined it does not generally preserve logical equivalence. For example, $[(\forall s).F(s)] \uparrow S_0$ is $(\forall s).F(s)$, but the logically equivalent $[F(S_0) \wedge (\forall s).F(s)] \uparrow S_0$ is $p \wedge (\forall s).F(s)$. Fortunately, we shall only be lifting those sentences that do preserve logical equivalence.

1. Instantiate the successor state axioms with α and S_0 to get $\mathcal{D}_{ss}[\alpha, S_0]$. This will be the only use made of the successor state axioms.
2. Replace $Poss(\alpha, S_0)$ in the above instantiation by the corresponding conditions on the right hand side of the action precondition axiom for α. This will be the only use made of the action precondition axioms.
3. The resulting formula and those in the initial database will generally mention S_0, but the progression needs to be about S_α only, so we need to somehow "forget" S_0 without losing any information. This is done by lifting S_0 from the formulas.

This procedure is justified, and described more precisely, by the following, which is the main theorem of this section:

Theorem 4.7. *Let $\mathcal{D}_{S_\alpha}$ be the union of $\mathcal{D}_{una}$ together with the sentence:*

$$(\exists p_1, \ldots, p_k)\left\{\left(\bigwedge_{\varphi \in \mathcal{D}_{S_0}} \varphi\right) \wedge \mathcal{D}_{ss}[\alpha, S_0](Poss/\Psi_\alpha)\right\} \uparrow S_0,$$

where

1. *$p_1, \ldots, p_k$ are the new predicate variables introduced during the lifting.*
2. *Ψ_α is a sentence in $\mathcal{L}_{S_0}$ such that*

$$Poss(\alpha, S_0) \equiv \Psi_\alpha$$

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

2. *$\mathcal{D}_{ss}[\alpha, S_0](Poss/\Psi_\alpha)$ is the result of replacing $Poss(\alpha, S_0)$ by Ψ_α in $\mathcal{D}_{ss}[\alpha, S_0]$.*

Then $\mathcal{D}_{S_\alpha}$ is a progression of $\mathcal{D}_{S_0}$ to S_α with respect to $\mathcal{D}$.

Proof. First, it is clear that the sentences in $\mathcal{D}_{S_\alpha}$ are in $\mathcal{L}^2_{S_\alpha}$.

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

Suppose that there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$. By Lemma 4.6, $\mathcal{D} \models \mathcal{D}_{S_\alpha}$, thus $M' \models \mathcal{D}_{S_\alpha}$. Therefore by Lemma 4.1, $M \models \mathcal{D}_{S_\alpha}$.

Now suppose that $M \models \mathcal{D}_{S_\alpha}$. Then there is a variable assignment σ such that

$$M, \sigma \models \left(\bigwedge_{\varphi \in \mathcal{D}_{S_0}} \varphi\right) \wedge \mathcal{D}_{ss}[\alpha, S_0](Poss/\Psi_\alpha) \uparrow S_0.$$

Now construct a structure M' such that:

1. M and M' have the same universe, and interpret all situation independent function and predicate symbols identically.
2. For every fluent F, if $F(\boldsymbol{x}, S_0)$ is lifted in $\mathcal{D}_{S_\alpha}$ as p, then

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

3. $M' \models \mathcal{D}_{ss} \cup \mathcal{D}_{ap}$.
4. If $M' \models \neg\Psi_\alpha$, then for any fluent F, and any variable assignment σ',

$$M', \sigma' \models F(\boldsymbol{x}, S_\alpha) \quad \text{iff} \quad M, \sigma' \models F(\boldsymbol{x}, S_\alpha).$$

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

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

$$M', \sigma' \models F(\boldsymbol{x}, S_\alpha) \quad \text{iff} \quad M, \sigma' \models F(\boldsymbol{x}, S_\alpha).$$

2. If $M' \models \Psi_\alpha$, then since $M' \models \mathcal{D}_{ap}$, and $\mathcal{D}_{ap} \models Poss(\alpha, S_0) \equiv \Psi_\alpha$, therefore $M' \models Poss(\alpha, S_0)$. But $M' \models \mathcal{D}_{ss}$. Thus for any fluent F, and any variable assignment σ',

$$M', \sigma' \models F(\boldsymbol{x}, S_\alpha) \quad \text{iff} \quad M', \sigma' \models \Phi_F(\boldsymbol{x}, \alpha, S_0), \tag{7}$$

where Φ_F is as in the successor state axiom (1) for F in $\mathcal{D}_{ss}$. Now since $M' \models \Psi_\alpha$, by our construction of M', we have that $M, \sigma \models \Psi_\alpha \upharpoonright S_0$. But

$$M, \sigma \models \mathcal{D}_{ss}[\alpha, S_0](Poss/\Psi_\alpha) \upharpoonright S_0.$$

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

$$M, \sigma' \models F(\boldsymbol{x}, S_\alpha) \quad \text{iff} \quad M', \sigma' \models \Phi_F(\boldsymbol{x}, \alpha, S_0) \upharpoonright S_0. \tag{8}$$

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

$$M, \sigma' \models \Phi_F(\boldsymbol{x}, \alpha, S_0) \upharpoonright S_0 \quad \text{iff} \quad M', \sigma' \models \Phi_F(\boldsymbol{x}, \alpha, S_0).$$

Therefore from (7) and (8), we see that for any fluent F, and any variable assignment σ',

$$M', \sigma' \models F(\boldsymbol{x}, S_\alpha) \quad \text{iff} \quad M, \sigma' \models F(\boldsymbol{x}, S_\alpha).$$

It follows then that $M \sim_{S_\alpha} M'$. By the construction of M' and the fact that $M \models \mathcal{D}_{una}$, we have that $M' \models \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una}$. Thus from Proposition 3.2, there is a model M'' of $\mathcal{D}$ such that $M' \sim_{S_\alpha} M''$. Then by the transitivity of $\sim_{S_\alpha}$, we have that $M \sim_{S_\alpha} M''$. This concludes the proof that $\mathcal{D}_{S_\alpha}$ as defined is a progressed database. □

It is clear that the theorem still holds when the initial database $\mathcal{D}_{S_0}$ is a finite set of second-order sentences in $\mathcal{L}^2_{S_0}$. 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 $\mathcal{D}_{S_\alpha}$ as defined in the theorem can be unwieldy. However, it can often be simplified by using the unique names axioms in $\mathcal{D}_{una}$, as we shall see in the following example.

Example 4.8. Consider our educational database. The instantiation of the successor state axioms on $drop(Sue, C100)$ and S_0, $\mathcal{D}_{ss}[drop(Sue, C100), S_0]$ is the conjunction of the following two sentences, where $\alpha = drop(Sue, C100)$ and $S_\alpha = do(\alpha, S_0)$:

$$\begin{aligned} Poss(\alpha, S_0) \supset [&enrolled(stu, c, S_\alpha) \equiv \\ &\alpha = register(stu, c) \vee \\ &enrolled(stu, c, S_0) \wedge \alpha \neq drop(stu, c)], \end{aligned}$$

$$Poss(\alpha, S_0) \supset [grade(stu, c, g, S_\alpha) \equiv \alpha = change(stu, c, g) \vee grade(stu, c, g, S_0) \wedge \neg(\exists g')(g \neq g' \wedge \alpha = change(stu, c, g'))].$$

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

$$Poss(\alpha, S_0) \supset [enrolled(stu, c, S_\alpha) \equiv enrolled(stu, c, S_0) \wedge (Sue \neq stu \vee C100 \neq c)],$$

$$Poss(\alpha, S_0) \supset [grade(stu, c, g, S_\alpha) \equiv grade(stu, c, g, S_0)].$$

By $\mathcal{D}_{ap}$,

$$Poss(\alpha, S_0) \equiv enrolled(Sue, C100, S_0).$$

Thus $\mathcal{D}_{ss}[\alpha, S_0](Poss/\Psi_\alpha)$ is the conjunction of the following two sentences:

$$enrolled(Sue, C100, S_0) \supset [enrolled(stu, c, S_\alpha) \equiv enrolled(stu, c, S_0) \wedge (Sue \neq stu \vee C100 \neq c)],$$

$$enrolled(Sue, C100, S_0) \supset [grade(stu, c, g, S_\alpha) \equiv grade(stu, c, g, S_0)].$$

Thus $(\exists p_1, p_2)[(\bigwedge_{\varphi \in \mathcal{D}_{S_0}} \varphi) \wedge \mathcal{D}_{ss}[\alpha, S_0](Poss/\Psi_\alpha)] \upharpoonright S_0$ is

$$\begin{aligned}(\exists p_1, p_2).\ & John \neq Sue \neq C100 \neq C200 \wedge \\ & [p_1(John, C100) \vee p_1(John, C200)] \wedge \\ & p_1(Sue, C100) \wedge prereq(C100, C200) \wedge \\ & p_1(Sue, C100) \supset enrolled(stu, c, S_\alpha) \equiv \\ & \qquad [p_1(stu, c) \wedge (Sue \neq stu \vee C100 \neq c)] \wedge \\ & p_1(Sue, C100) \supset grade(stu, c, g, S_\alpha) \equiv p_2(stu, c, g).\end{aligned}$$

This is equivalent to

$$\begin{aligned}& John \neq Sue \neq C100 \neq C200 \wedge prereq(C100, C200) \wedge \\ & (\exists p_1, p_2).\ [p_1(John, C100) \vee p_1(John, C200)] \wedge p_1(Sue, C100) \wedge \\ & \qquad enrolled(stu, c, S_\alpha) \equiv [p_1(stu, c) \wedge (Sue \neq stu \vee C100 \neq c)] \wedge \\ & \qquad grade(stu, c, g, S_\alpha) \equiv p_2(stu, c, g),\end{aligned}$$

which is equivalent to

$$\begin{aligned}& John \neq Sue \neq C100 \neq C200 \wedge prereq(C100, C200) \wedge \\ & (\exists p_1).\ [p_1(John, C100) \vee p_1(John, C200)] \wedge \\ & \qquad p_1(Sue, C100) \wedge \\ & \qquad enrolled(stu, c, S_\alpha) \equiv [p_1(stu, c) \wedge (Sue \neq stu \vee C100 \neq c)].\end{aligned}$$

Now, $enrolled(stu, c, S_\alpha) \equiv [p_1(stu, c) \wedge (Sue \neq stu \vee C100 \neq c)]$ can be broken into two cases:

$$Sue = stu \wedge C100 = c \supset$$
$$enrolled(stu, c, S_\alpha) \equiv [p_1(stu, c) \wedge (Sue \neq stu \vee C100 \neq c)] \wedge$$
$$Sue \neq stu \vee C100 \neq c \supset$$
$$enrolled(stu, c, S_\alpha) \equiv [p_1(stu, c) \wedge (Sue \neq stu \vee C100 \neq c)],$$

that is,

$$Sue = stu \wedge C100 = c \supset \neg enrolled(stu, c, S_\alpha) \wedge$$
$$Sue \neq stu \vee C100 \neq c \supset [enrolled(stu, c, S_\alpha) \equiv p_1(stu, c)],$$

so we can continue simplifying $(\exists p_1, p_2)[(\bigwedge_{\varphi \in \mathcal{D}_{S_0}} \varphi) \wedge \mathcal{D}_{ss}[\alpha, S_0](Poss/\Psi_\alpha)] \upharpoonright S_0$ into:

$$John \neq Sue \neq C100 \neq C200 \wedge prereq(C100, C200) \wedge$$
$$[enrolled(John, C100, S_\alpha) \vee enrolled(John, C200, S_\alpha)] \wedge$$
$$\neg enrolled(Sue, C100, S_\alpha) \wedge$$
$$(\exists p_1).\ p_1(Sue, C100) \wedge$$
$$Sue \neq stu \vee C100 \neq c \supset [enrolled(stu, c, S_\alpha) \equiv p_1(stu, c)].$$

Therefore we have a first-order representation for $\mathcal{D}_{S_\alpha}$, which is $\mathcal{D}_{una}$ together with the following sentences:

$$John \neq Sue \neq C100 \neq C200,$$

$$prereq(C100, C200),$$

$$enrolled(John, C100, S_\alpha) \vee enrolled(John, C200, S_\alpha),$$

$$\neg enrolled(Sue, C100, S_\alpha).$$

4.3. More on first-order progression

Theorem 4.5 informs us that, in particular, the progression of $\mathcal{D}_{S_0}$ entails the same set of *first-order* sentences about S_α as does $\mathcal{D}$. In view of this, one may wonder why we did not define progression to be the set of *first-order* sentences in $\mathcal{L}_{S_\alpha}$ entailed by $\mathcal{D}$. Indeed, this is basically what Pednault did [16], and will, by definition, side step our negative result that, in general, progression cannot be captured in first-order logic. There are several reasons why we did not do this. First, such a definition is purely syntactic, and hence has an arbitrary quality to it. What justifies the prior assumption that progression is first order definable, especially in view of the fact that many other notions, for example transitive closure, are not? Ideally, one should begin, as we did, with a purely semantic characterization of one's intuitions about database progression, and see where that leads. Secondly, Peppas et al. [19] show that, for

quite general action theories, progression defined in terms of first-order entailments, may lose information, in the sense that a first-order sentence about a future situation of S_α may be a consequence of $\mathcal{D}$ but not of $(\mathcal{D} - \mathcal{D}_{S_0}) \cup \mathcal{D}_{S_\alpha}$. While this result by Peppas et al. is for more general action theories than ours, it does show that it is not a priori obvious that a first-order definition of progression is warranted. Unfortunately, we have not been able to find a result for basic theories of actions comparable to that of Peppas et al. Nevertheless, we are convinced of the following:

Conjecture 4.9. *For an arbitrary basic action theory $\mathcal{D}$, and an arbitrary ground action α, let $\mathcal{F}_{S_\alpha}$ be the set of first-order sentences in $\mathcal{L}_{S_\alpha}$ entailed by $\mathcal{D}$. Then there is a basic action theory $\mathcal{D}$, a ground action term α and a first-order sentence σ such that σ is entailed by $(\mathcal{D} - \mathcal{D}_{S_0}) \cup \mathcal{D}_{S_\alpha}$ but not by $(\mathcal{D} - \mathcal{D}_{S_0}) \cup \mathcal{F}_{S_\alpha}$.*

If true, this conjecture would establish that a definition of progression in terms of first-order entailments would be too weak.

However, for an important class of first-order sentences, this "weaker" definition of progression is entirely adequate. Specifically, for addressing the projection problem, first-order progression is sufficient.

Proposition 4.10. *Suppose that $\phi(s) \in \mathcal{L}_s$, and that $\boldsymbol{A}$ is a sequence of ground action terms. Then*

$$(\mathcal{D} - \mathcal{D}_{S_0}) \cup \mathcal{D}_{S_\alpha} \models S_\alpha \leqslant do(\boldsymbol{A}, S_\alpha) \wedge \phi(do(\boldsymbol{A}, S_\alpha)) \tag{9}$$

iff

$$(\mathcal{D} - \mathcal{D}_{S_0}) \cup \mathcal{F}_{S_\alpha} \models S_\alpha \leqslant do(\boldsymbol{A}, S_\alpha) \wedge \phi(do(\boldsymbol{A}, S_\alpha)). \tag{10}$$

Proof. We make use of the soundness and completeness of regression, as described in Reiter [24]. Specifically, for sentences of the form $S_\alpha \leqslant do(\boldsymbol{A}, S_\alpha) \wedge \phi(do(\boldsymbol{A}, S_\alpha))$, Reiter shows how, using the regression operator, to determine a first-order sentence $\sigma \in \mathcal{L}_{S_\alpha}$ such that

$$\mathcal{D} - \mathcal{D}_{S_0} \models \sigma \equiv [S_\alpha \leqslant do(\boldsymbol{A}, S_\alpha) \wedge \phi(do(\boldsymbol{A}, S_\alpha))]. \tag{11}$$

Moreover, Reiter shows that (10) iff $\mathcal{D}_{una} \cup \mathcal{F}_{S_\alpha} \models \sigma$ iff (since $\mathcal{D}_{una} \subset \mathcal{F}_{S_\alpha}$) $\mathcal{F}_{S_\alpha} \models \sigma$ iff (by the definition of $\mathcal{F}_{S_\alpha}$) $\mathcal{D} \models \sigma$ iff (by the remarks following the proof of Proposition 4.4) $(\mathcal{D} - \mathcal{D}_{S_0}) \cup \mathcal{D}_{S_\alpha} \models \sigma$ iff (by (11)) (9). □

5. Progression with relatively complete initial databases

In the previous section we showed that, in general, progression is definable only in second-order logic. However, there are some interesting and important special cases for which progression is first-order definable. In this section and the next, we consider two such cases.

We say $\mathcal{D}_{S_0}$ is *relatively complete* if it is a set of situation independent sentences combined with a set of sentences, one for each fluent F, of the form:

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

where $\Pi_F(\boldsymbol{x})$ is a situation independent formula whose free variables are among $\boldsymbol{x}$.

When $\mathcal{D}_{S_0}$ is relatively complete, the truth value of each fluent F in the initial situation is completely determined by the truth value of the situation independent formula $\Pi_F(\boldsymbol{x})$. It does not follow that the initial database must be logically complete. It will be only when the initial situation uniquely determines the truth values of the situation independent predicates. Hence the terminology "relative completeness". For example, in the blocks world, one may want to specify that initially all and only green blocks are on the table, without saying which blocks are green:

$$ontable(x, S_0) \equiv green(x).$$

Theorem 5.1. *Let $\mathcal{D}$ be an action theory with a relatively complete initial database $\mathcal{D}_{S_0}$, and let α be a ground action term such that $\mathcal{D} \models Poss(\alpha, S_0)$. Then the following set:*

$$\mathcal{D}_{una} \cup \{\varphi \mid \varphi \in \mathcal{D}_{S_0} \textit{ is situation independent}\} \cup$$
$$\{(\forall \boldsymbol{x}).F(\boldsymbol{x}, do(\alpha, S_0)) \equiv \Phi_F(\boldsymbol{x}, \alpha, S_0)[S_0] \mid F \textit{ is a fluent}\}$$

is a progression of $\mathcal{D}_{S_0}$ to S_α, where

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

Proof. Denote the set of the sentences of the theorem by $\mathcal{S}$. Clearly, $\mathcal{S}$ is a set of first-order sentences in $\mathcal{L}_{S_\alpha}$. It is easy to see that $\mathcal{S} \models \mathcal{D}_{S_\alpha}$. Conversely, it is clear that $\mathcal{D} \models \mathcal{S}$. Thus by Theorem 4.5, $\mathcal{D}_{S_\alpha} \models \mathcal{S}$. □

Clearly, the progressed database at S_α 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 $\mathcal{D}_{una}$ and the situation independent axioms in $\mathcal{D}_{S_0}$; therefore we can use these axioms to simplify $\Phi_F(\boldsymbol{x}, \alpha, S_0)[S_0]$.

Example 5.2. Consider again our educational database example. Suppose now that the initial database $\mathcal{D}_{S_0}$ consists of the following axioms:

$$John \neq Sue \neq C100 \neq C200,$$

$$better(70, 50),$$

$$prereq(C100, C200),$$

$$enrolled(stu, c, S_0) \equiv (stu = John \wedge c = C100) \vee (stu = Sue \wedge c = C200),$$

$$grade(stu, c, g, S_0) \equiv stu = Sue \wedge c = C100 \wedge g = 70.$$

$\mathcal{D}_{S_0}$ is relatively complete, and $\mathcal{D} \models Poss(\alpha, S_0)$, where $\alpha = drop(John, C100)$. From the axiom for *enrolled* in $\mathcal{D}_{S_0}$, we see that $\Pi_{enrolled}(stu, c)$ is the formula:

$$(stu = John \wedge c = C100) \vee (stu = Sue \wedge c = C200).$$

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

$$a = register(stu, c) \vee (enrolled(stu, c, s) \wedge a \neq drop(stu, c)).$$

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

$$drop(John, C100) = register(stu, c) \vee$$
$$\{[(stu = John \wedge c = C100) \vee (stu = Sue \wedge c = C200)] \wedge$$
$$drop(John, C100) \neq drop(stu, c)\}.$$

By the unique names axioms in $\mathcal{D}_{una}$, this can be simplified to

$$[(stu = John \wedge c = C100) \vee (stu = Sue \wedge c = C200)] \wedge$$
$$\neg(stu = John \wedge c = C100).$$

By the unique names axioms in $\mathcal{D}_{S_0}$, this can be further simplified to

$$stu = Sue \wedge c = C200.$$

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

$$enrolled(stu, c, do(\alpha, S_0)) \equiv stu = Sue \wedge c = C200.$$

Similarly, we have:

$$grade(stu, c, g, do(\alpha, S_0)) \equiv stu = Sue \wedge c = C100 \wedge g = 70.$$

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

$$John \neq Sue \neq C100 \neq C200,$$

$$better(70, 50),$$

$$prereq(C100, C200),$$

$$enrolled(stu, c, do(\alpha, S_0)) \equiv stu = Sue \wedge c = C200,$$

$$grade(stu, c, g, do(\alpha, S_0)) \equiv stu = Sue \wedge c = C100 \wedge g = 70.$$

6. Progression in the context free case

In this section we consider progression with respect to context free action theories. A successor state axiom for F is *context free* iff it has the form:

$$Poss(a,s) \supset [F(\boldsymbol{x}, do(a,s)) \equiv \gamma_F^+(\boldsymbol{x},a) \vee (F(\boldsymbol{x},s) \wedge \neg\gamma_F^-(\boldsymbol{x},a))], \tag{12}$$

where $\gamma_F^+(\boldsymbol{x},a)$ and $\gamma_F^-(\boldsymbol{x},a)$ are situation independent formulas whose free variables are among those in $\boldsymbol{x}, a$. The successor state axioms in our educational database are all context free. So is the following successor state axiom:

$$Poss(a,s) \supset [broken(x, do(a,s)) \equiv a = drop(x) \wedge fragile(x) \vee broken(x,s) \wedge a \neq repair(x)].$$

The following successor state axiom is not context free:

$$Poss(a,s) \supset [dead(x, do(a,s)) \equiv (\exists y)(a = explode\_bomb\_at(y) \wedge close(x,y,s)) \vee dead(x,s)].$$

Intuitively, a successor state axiom for fluent F is context free iff F's truth value in the next situation $do(a,s)$ depends on F's truth value in the current situation s, but is independent of the truth values of any other fluents in s.

Now assume that:

1. $\mathcal{D}_{S_0}$ is a set of situation independent sentences, and sentences of the form

$$E \supset \pm F(x_1, \ldots, x_n, S_0), \tag{13}$$

where F is a fluent and E is a situation independent formula. For example,

$$ontable(x, S_0),$$

$$x \neq A \supset \neg ontable(x, S_0),$$

$$fragile(x) \supset broken(x, S_0)$$

are all of this form. The following are not of this form:

$$ontable(x, S_0) \vee onfloor(x, S_0),$$

$$(\exists x) ontable(x, S_0).$$

2. $\mathcal{D}_{S_0}$ is *coherent* in the sense that for every fluent F, whenever $(\forall \boldsymbol{x}).E_1 \supset F(\boldsymbol{x}, S_0)$ and $(\forall \boldsymbol{x}).E_2 \supset \neg F(\boldsymbol{x}, S_0)$ are in $\mathcal{D}_{S_0}$, then

$$\{\varphi \mid \varphi \in \mathcal{D}_{S_0} \text{ is situation independent}\} \models (\forall \boldsymbol{x}).\neg(E_1 \wedge E_2).$$

This means that $\mathcal{D}_{S_0}$ cannot use axioms of the form (13) to encode situation independent sentences: for any situation independent sentence ϕ, $\mathcal{D}_{S_0} \models \phi$ iff $\{\varphi \mid \varphi \in \mathcal{D}_{S_0} \text{ is situation independent}\} \models \phi$.

3. $\mathcal{D}_{ss}$ is a set of context free successor state axioms.

4. α is a ground action term, and is possible initially: $\mathcal{D} \models Poss(\alpha, S_0)$.
5. For each fluent F, the following consistency condition (Reiter [21]) is satisfied:

$$\mathcal{D}_{ap} \cup \mathcal{D}_{una} \models \neg(\exists \boldsymbol{x}, a, s).Poss(a, s) \wedge \gamma_F^+(\boldsymbol{x}, a) \wedge \gamma_F^-(\boldsymbol{x}, a), \tag{14}$$

where F's successor state axiom has the form (12).

The consistency condition (14) deserves a brief explanation. Following Pednault [18] and Schubert [27], Reiter [21] provides a solution to the frame problem in the absence of state constraints which syntactically transforms a pair of effect axioms for a given fluent F into a successor state axiom for F. The effect axioms are assumed to have the syntactic forms: [10]

$$Poss(a, s) \wedge \gamma_F^+(\boldsymbol{x}, a, s) \supset F(\boldsymbol{x}, do(a, s)),$$

and

$$Poss(a, s) \wedge \gamma_F^-(\boldsymbol{x}, a, s) \supset \neg F(\boldsymbol{x}, do(a, s)).$$

Reiter applies the *explanation closure* idea of Schubert [27] to obtain the following frame axioms for F:

$$Poss(a, s) \wedge \neg F(\boldsymbol{x}, s) \wedge F(\boldsymbol{x}, do(a, s)) \supset \gamma_F^+(\boldsymbol{x}, a, s),$$

$$Poss(a, s) \wedge F(\boldsymbol{x}, s) \wedge \neg F(\boldsymbol{x}, do(a, s)) \supset \gamma_F^-(\boldsymbol{x}, a, s).$$

The successor state axiom

$$Poss(a, s) \supset F(\boldsymbol{x}, do(a, s)) \equiv \gamma_F^+(\boldsymbol{x}, a) \vee (F(\boldsymbol{x}, s) \wedge \neg\gamma_F^-(\boldsymbol{x}, a))$$

is logically equivalent to the conjunction of the above four sentences, whenever the consistency condition holds. Notice that the consistency condition makes good sense: if it were violated, so that for some $\boldsymbol{X}$, A, S we have $Poss(A, S)$, $\gamma_F^+(\boldsymbol{X}, A, S)$, and $\gamma_F^-(\boldsymbol{X}, A, S)$, then we could derive an immediate inconsistency from the above two effect axioms.

It is easy to verify that each fluent in our educational database satisfies the consistency condition.

Under these assumptions, to compute $\mathcal{D}_{S_\alpha}$, use Theorem 4.5 to construct a set $\mathcal{S}$, initially empty, of sentences as follows:

1. If $\varphi \in \mathcal{D}_{S_0}$ is situation independent, then $\varphi \in \mathcal{S}$.
2. For any fluent F, add to $\mathcal{S}$ the sentences

$$\gamma_F^+(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)), \tag{15}$$

$$\gamma_F^-(\boldsymbol{x}, \alpha) \supset \neg F(\boldsymbol{x}, do(\alpha, S_0)). \tag{16}$$

3. For any fluent F, if $(\forall \boldsymbol{x}).E \supset F(\boldsymbol{x}, S_0)$ is in $\mathcal{D}_{S_0}$, then add to $\mathcal{S}$ the sentence

$$E \wedge \neg\gamma_F^-(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)). \tag{17}$$

[10] In general, γ_F^+ and γ_F^- may be situation dependent.

4. For any fluent F, if $(\forall \boldsymbol{x}).E \supset \neg F(\boldsymbol{x}, S_0)$ is in $\mathcal{D}_{S_0}$, then add to $\mathcal{S}$ the sentence

$$E \wedge \neg \gamma_F^+(\boldsymbol{x}, \alpha) \supset \neg F(\boldsymbol{x}, do(\alpha, S_0)). \tag{18}$$

Theorem 6.1. *Under the afore-mentioned assumptions, $\mathcal{S} \cup \mathcal{D}_{una}$ is a progression of $\mathcal{D}_{S_0}$ to S_α.*

Proof. We use Theorem 4.5. First we show that $\mathcal{D} \models \mathcal{S} \cup \mathcal{D}_{una}$. $\mathcal{D} \models \mathcal{D}_{una}$ trivially. Suppose $\varphi \in \mathcal{S}$, we show by cases that $\mathcal{D} \models \varphi$:

1. $\varphi \in \mathcal{D}_{S_0}$ is situation independent. Trivial.
2. φ is (15). By the successor state axiom (12) of F in $\mathcal{D}$, we have

$$Poss(\alpha, S_0) \wedge \gamma_F^+(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)).$$

   From this and our assumption that $\mathcal{D} \models Poss(\alpha, S_0)$, we have

$$\gamma_F^+(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)),$$

   that is, φ.
3. φ is (16). Again by the successor state axiom (12) of F in $\mathcal{D}$, we have

$$Poss(\alpha, S_0) \supset [\gamma_F^-(\boldsymbol{x}, \alpha) \wedge \neg \gamma_F^+(\boldsymbol{x}, \alpha) \supset \neg F(\boldsymbol{x}, do(\alpha, S_0))].$$

   Now by the consistency condition (14), we have

$$Poss(\alpha, S_0) \supset [\gamma_F^-(\boldsymbol{x}, \alpha) \supset \neg F(\boldsymbol{x}, do(\alpha, S_0))].$$

   So we have

$$\gamma_F^-(\boldsymbol{x}, \alpha) \supset \neg F(\boldsymbol{x}, do(\alpha, S_0)),$$

   that is, φ.
4. φ is (17). By the successor state axiom (12) of F in $\mathcal{D}$, we have

$$Poss(\alpha, S_0) \supset [F(\boldsymbol{x}, S_0) \wedge \neg \gamma_F^-(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0))].$$

   So

$$F(\boldsymbol{x}, S_0) \wedge \neg \gamma_F^-(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)).$$

   But $(\forall \boldsymbol{x})(E \supset F(\boldsymbol{x}, S_0))$ is in $\mathcal{D}_{S_0}$, so we have

$$E \wedge \neg \gamma_F^-(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)),$$

   that is, φ.
5. φ is (18). By the successor state axiom (12) of F in $\mathcal{D}$, we have

$$Poss(\alpha, S_0) \supset [\neg F(\boldsymbol{x}, S_0) \wedge \neg \gamma_F^+(\boldsymbol{x}, \alpha) \supset \neg F(\boldsymbol{x}, do(\alpha, S_0))].$$

   So

$$\neg F(\boldsymbol{x}, S_0) \wedge \neg \gamma_F^+(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)).$$

But $(\forall \boldsymbol{x})(E \supset \neg F(\boldsymbol{x}, S_0))$ is in $\mathcal{D}_{S_0}$, so we have

$$E \wedge \neg \gamma_F^+(\boldsymbol{x}, \alpha) \supset \neg F(\boldsymbol{x}, do(\alpha, S_0)),$$

that is, φ.

By our construction of $\mathcal{S}$, this proves that $\mathcal{D} \models \mathcal{S}$. But $\mathcal{D}_{una} \cup \mathcal{S}$ is a set of sentences in $\mathcal{L}_{S_\alpha}$. Therefore by Theorem 4.5, $\mathcal{D}_{S_\alpha} \models \mathcal{S} \cup \mathcal{D}_{una}$.

To prove the converse, we show that for any model M of $\mathcal{S} \cup \mathcal{D}_{una}$, there is a model M' of $\mathcal{D}$ such that $M \sim_{S_\alpha} M'$. Suppose now that M is a model of $\mathcal{S} \cup \mathcal{D}_{una}$. We construct M' as follows:

1. M' and M have the same domains for sorts *action* and *object*, and interpret all situation independent predicates and functions the same.
2. For each fluent F, M' interprets F on S_0 as follows:
   (a) For every variable assignment σ, if $(\forall \boldsymbol{x}).E \supset F(\boldsymbol{x}, S_0)$ is in $\mathcal{D}_{S_0}$, and $M, \sigma \models E$ (thus $M', \sigma \models E$ as well), then $M', \sigma \models F(\boldsymbol{x}, S_0)$.
   (b) Similarly, for every variable assignment, if $(\forall \boldsymbol{x}).E \supset \neg F(\boldsymbol{x}, S_0)$ is in $\mathcal{D}_{S_0}$, and $M, \sigma \models E$ (thus $M', \sigma \models E$ as well), then $M', \sigma \models \neg F(\boldsymbol{x}, S_0)$.
   (c) For every variable assignment σ, if $F(\boldsymbol{x}, S_0)$ has not been assigned a truth value by one of the above two steps, then $M', \sigma \models F(\boldsymbol{x}, S_0)$ iff $M, \sigma \models F(\boldsymbol{x}, do(\alpha, S_0))$.

   Notice that by our coherence assumption for $\mathcal{D}_{S_0}$, our construction is well defined.
3. M' interprets *Poss* according to $\mathcal{D}_{ap}$, and interprets the truth values of the fluents on reachable situations according to $\mathcal{D}_{ss}$.
4. $M' \models \Sigma$. This can be done according to Proposition 3.2.

Clearly $M' \models \mathcal{D}$. We show now that $M \sim_{S_\alpha} M'$. For any fluent F, suppose the successor state axiom for F is

$$Poss(\alpha, s) \supset F(\boldsymbol{x}, do(\alpha, s)) \equiv \gamma_F^+(\boldsymbol{x}, \alpha) \vee (F(\boldsymbol{x}, s) \wedge \neg \gamma_F^-(\boldsymbol{x}, \alpha)).$$

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

1. $M', \sigma \models \gamma_F^+(\boldsymbol{x}, \alpha)$. This implies $M, \sigma \models \gamma_F^+(\boldsymbol{x}, \alpha)$. Now since $\gamma_F^+(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0)) \in \mathcal{S}$, and M is a model of $\mathcal{S}$, thus $M, \sigma \models F(\boldsymbol{x}, do(\alpha, S_0))$ as well.
2. $M', \sigma \models F(\boldsymbol{x}, S_0) \wedge \neg \gamma_F^-(\boldsymbol{x}, \alpha)$. Since $M', \sigma \models F(\boldsymbol{x}, S_0)$, by our construction, either $M, \sigma \models F(\boldsymbol{x}, do(\alpha, S_0))$, or there is a sentence $E \supset F(\boldsymbol{x}, S_0)$ in $\mathcal{D}_{S_0}$ such that $M, \sigma \models E$. Suppose the latter. Then by our construction of $\mathcal{S}$, it contains $E \wedge \neg \gamma_F^-(\boldsymbol{x}, \alpha) \supset F(\boldsymbol{x}, do(\alpha, S_0))$. Thus $M, \sigma \models F(\boldsymbol{x}, do(\alpha, S_0))$ as well.

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

Note the following:

1. The new database $\mathcal{S}$ has the same syntactic form as $\mathcal{D}_{S_0}$, so this process can be iterated.
2. The computation of $\mathcal{S}$ is very efficient, and the size of $\mathcal{S}$ is bounded by the sum of the size of $\mathcal{D}_{S_0}$ and twice the number of fluents.

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 [16] shows that for context free actions and arbitrary $\mathcal{D}_{S_0}$, progression is not always guaranteed to yield finite first-order theories.

7. STRIPS

Ever since STRIPS was first introduced (Fikes and Nilsson [6]), its logical semantics has been problematic. There have been many proposals in the literature (e.g. Lifschitz [11], Erol, Nau and Subrahmanian [4], Bacchus and Yang [2]). These all have in common a reliance on meta-theoretic operations on logical theories 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 sequel, we shall provide a semantics for STRIPS-like systems in terms of a purely declarative situation calculus axiomatization for actions and their effects. On our view, a STRIPS operator is a *mechanism* for computing the progression of an initial situation calculus database under the effects of an action. We shall illustrate this idea by describing two different STRIPS mechanisms, and proving their correctness with respect to their situation calculus specifications.

Following Lifschitz [11], define an *operator description* to be a triple (P, D, A), where P is a sentence of a first-order language $\mathcal{L}_{STRIPS}$ and D (the *delete list*) and A (the *add list*) are sets of sentences of $\mathcal{L}_{STRIPS}$. A *world description* W is any set of sentences of $\mathcal{L}_{STRIPS}$. A *STRIPS system* consists of:

1. a world description W_0, called the *initial* world description,
2. a binary relation $\rhd \subseteq 2^{\mathcal{L}_{STRIPS}} \times \mathcal{L}_{STRIPS}$,[11]
3. a set Op of symbols called *operators*, and
4. a family of operator descriptions $\{(P_\alpha, D_\alpha, A_\alpha)\}_{\alpha \in Op}$.

With each operator α is associated a world description W_α, the *successor world description of* W_0, defined by $W_\alpha = (W_0 - D_\alpha) \cup A_\alpha$. A successor world description W_α is *admissible* iff $W_0 \rhd P_\alpha$.

Sometimes, but not always, $\rhd$ will be the standard entailment relation for the first-order language $\mathcal{L}_{STRIPS}$. In this case, admissibility simply corresponds to the fact that the precondition P_α is entailed by the initial world description W_0, in which case, on the standard view of STRIPS, the operator α is applicable. However, our intuitions about STRIPS are not standard, and we prefer to leave open the interpretation of the "entailment" relation $\rhd$.

Our semantics for STRIPS systems is indirect; we define certain classes of theories in the situation calculus and show how to associate suitable STRIPS systems with those theories. Only STRIPS systems associated with such situation calculus theories will, on our account of STRIPS, be assigned a semantics. This leaves many STRIPS systems

[11] In his treatment of STRIPS, Lifschitz does not provide for the relation $\rhd$.

(namely those without an associated situation calculus theory) without a semantics; we are not very distressed by this, given that STRIPS systems, in their full generality, do not currently have coherent semantics anyway.

8. Two versions of STRIPS

The STRIPS systems we derive apply only to a restricted class of situation calculus action theories for which the successor state axioms have a particular syntactic form, which we now define. A successor state axiom is *strongly context free* iff it has the form:

$$\begin{aligned} Poss(a,s) \supset [F(\boldsymbol{x}, do(a,s)) \equiv \\ (\exists \boldsymbol{v}^{(1)}) a = A_1(\boldsymbol{\xi}^{(1)}) \vee \cdots \vee (\exists \boldsymbol{v}^{(m)}) a = A_m(\boldsymbol{\xi}^{(m)}) \vee \\ F(\boldsymbol{x}, s) \wedge \neg(\exists \boldsymbol{w}^{(1)}) a = B_1(\boldsymbol{\eta}^{(1)}) \wedge \\ \cdots \wedge \neg(\exists \boldsymbol{w}^{(n)}) a = B_n(\boldsymbol{\eta}^{(n)})]. \end{aligned} \tag{19}$$

Here the A and B are function symbols of sort *action, not necessarily distinct from one another.* The $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are sequences of distinct variables which *include all of the variables of* $\boldsymbol{x}$; the remaining variables of the $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are those being existentially quantified by the $\boldsymbol{v}$ and $\boldsymbol{w}$, respectively. $\boldsymbol{x}$ could be the empty sequence. Notice that strongly context free successor state axioms are special cases of context free successor state axioms defined in Section 6. The successor state axioms of our running blocks world example given below are strongly context free. The following successor state axiom is context free but not strongly context free:

$$\begin{aligned} Poss(a,s) \supset [ontable(x, do(a,s)) \equiv a = putontable(x) \vee \\ ontable(x,s) \wedge a \neq tiptable \wedge a \neq pickup(x)]. \end{aligned}$$

This is because the action *tiptable* does not have x as a parameter.

The STRIPS systems which we shall characterize will be for languages $\mathcal{L}^2$ whose only function symbols of sort *object* are constants. Therefore, consider a ground action term α, and the strongly context free successor state axiom (19) for fluent F, relativized to the initial situation S_0. How does α affect the truth value of fluent F in the successor situation $do(\alpha, S_0)$? By the unique names axioms for actions, together with the assumption that the successor state axioms are strongly context free, this relativized axiom will be logically equivalent to a sentence of the form:

$$\begin{aligned} Poss(\alpha, S_0) \supset [F(\boldsymbol{x}, do(\alpha, S_0)) \equiv \\ \boldsymbol{x} = \boldsymbol{X}^{(1)} \vee \cdots \vee \boldsymbol{x} = \boldsymbol{X}^{(m)} \vee \\ F(\boldsymbol{x}, S_0) \wedge \boldsymbol{x} \neq \boldsymbol{Y}^{(1)} \wedge \cdots \wedge \boldsymbol{x} \neq \boldsymbol{Y}^{(n)}]. \end{aligned}$$

Here the $\boldsymbol{X}$ and $\boldsymbol{Y}$ are tuples of constants of $\mathcal{L}^2$ obtained from those mentioned by the ground action term α. If we assume further that the action α is possible in the initial situation, i.e., that $\mathcal{D} \models Poss(\alpha, S_0)$, this is equivalent to:

$$F(\boldsymbol{x}, do(\alpha, S_0)) \equiv$$
$$\boldsymbol{x} = \boldsymbol{X}^{(1)} \vee \cdots \vee \boldsymbol{x} = \boldsymbol{X}^{(m)} \vee F(\boldsymbol{x}, S_0) \wedge \boldsymbol{x} \neq \boldsymbol{Y}^{(1)} \wedge \cdots \wedge \boldsymbol{x} \neq \boldsymbol{Y}^{(n)}. \tag{20}$$

Example 8.1. The following blocks world will provide a running example for the rest of this paper:

Actions

- *move*(x, y, z): move the block x from block y onto block z, provided both x and z are clear and block x is on top of block y.
- *movefromtable*(x, y): move the block x from the table onto block y, provided x is clear and on the table, and block y is clear.
- *movetotable*(x, y): move block x from block y onto the table, provided x is clear and x is on y.

Fluents

- *clear*(x, s): block x has no other blocks on top of it, in state s.
- *on*(x, y, s): block x is on (touching) block y, in state s.
- *ontable*(x, s): block x is on the table, in state s.

This setting can be axiomatized as follows:

Action precondition axioms

$$Poss(move(x, y, z), s) \equiv$$
$$clear(x, s) \wedge clear(z, s) \wedge on(x, y, s) \wedge x \neq y \wedge x \neq z \wedge y \neq z,$$

$$Poss(movefromtable(x, y), s) \equiv$$
$$clear(x, s) \wedge clear(y, s) \wedge ontable(x, s) \wedge x \neq y,$$

$$Poss(movetotable(x, y), s) \equiv clear(x, s) \wedge on(x, y, s) \wedge x \neq y.$$

Successor state axioms

$$Poss(a, s) \supset [clear(x, do(a, s)) \equiv$$
$$(\exists y, z)a = move(y, x, z) \vee (\exists y)a = movetotable(y, x) \vee$$
$$clear(x, s) \wedge \neg(\exists y, z)a = move(y, z, x) \wedge$$
$$\neg(\exists y)a = movefromtable(y, x)],$$

$$Poss(a, s) \supset [on(x, y, do(a, s)) \equiv$$
$$(\exists z)a = move(x, z, y) \vee a = movefromtable(x, y) \vee$$
$$on(x, y, s) \wedge a \neq movetotable(x, y) \wedge \neg(\exists z)a = move(x, y, z)],$$

$$Poss(a, s) \supset [ontable(x, do(a, s)) \equiv$$
$$(\exists y)a = movetotable(x, y) \vee$$
$$ontable(x, s) \wedge \neg(\exists y)a = movefromtable(x, y)].$$

Now consider the "generic" ground action $move(X, Y, Z)$. The corresponding instances of (20) for the fluents *clear*, *on* and *ontable* are logically equivalent to:

$$clear(x, do(move(X, Y, Z), S_0)) \equiv x = Y \vee clear(x, S_0) \wedge x \neq Z,$$

$$on(x, y, do(move(X, Y, Z), S_0)) \equiv$$
$$x = X \wedge y = Z \vee on(x, y, S_0) \wedge \neg[x = X \wedge y = Y],$$

$$ontable(x, do(move(X, Y, Z), S_0)) \equiv ontable(x, S_0).$$

For the generic ground actions $movefromtable(X, Y)$ and $movetotable(X, Y)$ we obtain:

$$clear(x, do(movefromtable(X, Y), S_0)) \equiv clear(x, S_0) \wedge x \neq Y,$$

$$on(x, y, do(movefromtable(X, Y), S_0)) \equiv x = X \wedge y = Y \vee on(x, y, S_0),$$

$$ontable(x, do(movefromtable(X, Y), S_0)) \equiv ontable(x, S_0) \wedge x \neq X,$$

$$clear(x, do(movetotable(X, Y), S_0)) \equiv x = Y \vee clear(x, S_0),$$

$$on(x, y, do(movetotable(X, Y), S_0)) \equiv on(x, y, S_0) \wedge \neg[x = X \wedge y = Y],$$

$$ontable(x, do(movetotable(X, Y), S_0)) \equiv x = X \vee ontable(x, S_0).$$

8.1. OCF-STRIPS: open world, context free STRIPS

In this section we characterize an open world version of STRIPS—open world in the sense that its database is a set of ground *literals* (not *atoms* with a closed world assumption, as in most versions of STRIPS), and moreover, this database need not be logically complete. In other words, a certain degree of information incompleteness is permitted. Our point of departure is an action theory $\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$, with the following properties:

1. The only function symbols of sort *object* that the second-order language $\mathcal{L}^2$ possesses are constants.[12]
2. Each situation dependent sentence of $\mathcal{D}_{S_0}$ is a ground fluent literal, i.e., of the form $F(\boldsymbol{C}, S_0)$ or $\neg F(\boldsymbol{C}, S_0)$ for fluent F and constants $\boldsymbol{C}$ of sort *object*.
3. $\mathcal{D}_{S_0}$ contains *unique names axioms* for constants of sort *object*: for each pair of distinct constant names C and C' of sort *object*, the axiom $C \neq C'$.
4. $\mathcal{D}_{S_0}$ contains no pair of complementary literals (and hence is consistent).
5. Each successor state axiom of $\mathcal{D}_{ss}$ is strongly context free.
6. We are progressing with respect to α, a ground action term, and α is possible initially:

$$\mathcal{D} \models Poss(\alpha, S_0).$$

[12] Recall that $\mathcal{L}^2$ is the language in which $\mathcal{D}$ is expressed.

7. For each fluent F, the consistency condition (14) is satisfied. It is easy (but tedious) to verify that each fluent of Example 8.1 satisfies this condition.

In keeping with our intuition that STRIPS systems are mechanisms for progressing situation calculus databases, we want now to characterize the result of progressing $\mathcal{D}_{S_0}$ under the effects of the ground action α in the case of action theories of the above kind. Our basis for this will be Theorem 6.1.

Let $\mathcal{S}$ be the following set of sentences:

1. Initialize $\mathcal{S}$ to $\{\varphi \in \mathcal{D}_{S_0} \mid \varphi \text{ is situation independent}\}$.
2. For each fluent F do (with reference to the instance (20) of F's successor state axiom):
   (a) Add to $\mathcal{S}$ the sentence $F(\boldsymbol{X}^{(i)}, do(\alpha, S_0))$, $i = 1, \ldots, m$.
   (b) For each ground instance $F(\boldsymbol{C}, S_0) \in \mathcal{D}_{S_0}$ add to $\mathcal{S}$ the sentence $F(\boldsymbol{C}, do(\alpha, S_0))$, whenever $\boldsymbol{C}$ is a tuple of constants different from each $\boldsymbol{Y}^{(i)}$, $i = 1, \ldots, n$. (Here, we invoke the unique names axioms for constants of sort *object*.)
   (c) Add to $\mathcal{S}$ the sentence $\neg F(\boldsymbol{Y}^{(i)}, do(\alpha, S_0))$, $i = 1, \ldots, n$.
   (d) For each ground instance $\neg F(\boldsymbol{C}, S_0) \in \mathcal{D}_{S_0}$ add to $\mathcal{S}$ the sentence $\neg F(\boldsymbol{C}, do(\alpha, S_0))$, whenever $\boldsymbol{C}$ is a tuple of constants different from each $\boldsymbol{X}^{(i)}$, $i = 1, \ldots, m$. (We again invoke the unique names axioms for constants of sort *object*.)

By Theorem 6.1, the resulting set $\mathcal{S}$ enjoys the property that $\mathcal{S} \cup \mathcal{D}_{una}$ is a progression of $\mathcal{D}_{S_0}$ under action α. Moreover, the situation dependent sentences of $\mathcal{S}$ are all ground literals, and $\mathcal{S}$ contains no pair of complementary literals. It follows that $\mathcal{S}$ can serve as a new initial database for the purposes of iterating the above progression mechanism.

Now we interpret the above construction of the set $\mathcal{S}$ as a STRIPS operator. Imagine suppressing the situation argument S_0 of all the ground literals of $\mathcal{D}_{S_0}$. Now ask what sequence of deletions and additions of ground literals must be performed on the situation-suppressed version of $\mathcal{D}_{S_0}$ in order to obtain the situation-suppressed version of $\mathcal{S}$ (i.e. $\mathcal{S}$ with the situation argument $do(\alpha, S_0)$ suppressed in its sentences). The deletions and additions necessary to achieve this situation-suppressed transformation of $\mathcal{D}_{S_0}$ to $\mathcal{S}$ will define the delete and add lists for the STRIPS operator α.

It is easy to see that the following deletions and additions, when applied to $\mathcal{D}_0$, the situation-suppressed version of $\mathcal{D}_{S_0}$, yields the situation-suppressed version of $\mathcal{S}$:

For each fluent F do (with reference to the instance (20) of F's successor state axiom):

1. Delete from $\mathcal{D}_0$ the sentences $\neg F(\boldsymbol{X}^{(i)})$, $i = 1, \ldots, m$.
2. Delete from $\mathcal{D}_0$ the sentences $F(\boldsymbol{Y}^{(i)})$, $i = 1, \ldots, n$.
3. Add to $\mathcal{D}_0$ the sentences $F(\boldsymbol{X}^{(i)})$, $i = 1, \ldots, m$.
4. Add to $\mathcal{D}_0$ the sentences $\neg F(\boldsymbol{Y}^{(i)})$, $i = 1, \ldots, n$.

It is now clear how to define a STRIPS system and its associated operator for α: [13]

[13] See Section 7 for the relevant definitions.

1. The language $\mathcal{L}_{STRIPS}$ is the situation-suppressed version of $\mathcal{L}^2$.[14]
2. The initial world description is $\mathcal{D}_0$.
3. $\rhd$ is ordinary logical entailment; for a world description W and sentence $\sigma \in \mathcal{L}_{STRIPS}$, $W \rhd \sigma$ iff $W \models \sigma$.
4. α's precondition is the situation-suppressed version of the right hand side of the equivalence in α's situation calculus action precondition axiom.
5. For each fluent F, include in α's add and delete lists those literals specified above for obtaining the situation-suppressed version of $\mathcal{S}$.

To our knowledge, OCF-STRIPS is the only variant of STRIPS which specifically provides for an incomplete database of ground literals, and which is provably correct with respect to a logical specification.

Example 8.2. Continuing with our blocks world example, we can "read off" the OCF-STRIPS operator schema for *move* from the instances of the successor state axioms given in Example 8.1:

$move(X, Y, Z)$ [15]

P: $clear(X) \wedge clear(Z) \wedge on(X, Y) \wedge X \neq Z \wedge X \neq Y \wedge Y \neq Z$.

D: $\neg clear(Y), clear(Z), \neg on(X, Z), on(X, Y)$.

A: $clear(Y), \neg clear(Z), on(X, Z), \neg on(X, Y)$.

The operator description schemas for *movefromtable* and *movetotable* are obtained in the same way:

$movefromtable(X, Y)$

P: $clear(X) \wedge clear(Y) \wedge ontable(X) \wedge X \neq Y$.

D: $\neg on(X, Y), ontable(X), clear(Y)$.

A: $on(X, Y), \neg ontable(X), \neg clear(Y)$.

$movetotable(X, Y)$

P: $clear(X) \wedge on(X, Y) \wedge X \neq Y$.

D: $\neg clear(Y), on(X, Y), \neg ontable(X)$.

A: $clear(Y), \neg on(X, Y), ontable(X)$.

8.2. *RCF-STRIPS: relational, context free STRIPS*

In this section, we characterize a relational version of STRIPS—relational in the sense that its database is a conventional relational database. This version of STRIPS derives from action theories $\mathcal{D}$ of the form $\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$, with the following properties:

[14] We take it as self evident what is meant formally by the language obtained by suppressing objects of sort *situation* from the language $\mathcal{L}^2$.

[15] Notice that these are *schemas*, standing for the family of operators obtained by instantiating the "variables" X, Y and Z of the schema by constants of our situation calculus language.

1. The only function symbols of sort *object* that the second-order language $\mathcal{L}^2$ possesses are constants.
2. $\mathcal{D}_{S_0}$ contains one sentence of the following form, for each fluent F:

$$F(\boldsymbol{x}, S_0) \equiv \boldsymbol{x} = \boldsymbol{C}^{(1)} \vee \cdots \vee \boldsymbol{x} = \boldsymbol{C}^{(n)}, \tag{21}$$

where the $\boldsymbol{C}^{(i)}$ are tuples of constant symbols of sort *object*. These are the only situation dependent sentences of $\mathcal{D}_{S_0}$. Notice that initial databases of this form are special cases of the relatively complete databases defined in Section 5. The case $n = 0$ is permitted, in which case this axiom is $F(\boldsymbol{x}, S_0) \equiv \mathit{false}$. For example, if an agent's hand is initially empty:

$$\mathit{holding}(x, S_0) \equiv \mathit{false}.$$

If initially, block A is on B, D is on A, C is on E, and no other block is on a block:

$$\mathit{on}(x, y, S_0) \equiv x = A \wedge y = B \vee x = D \wedge y = A \vee x = C \wedge y = E.$$

3. $\mathcal{D}_{S_0}$ contains *unique names axioms* for constants of sort *object*.
4. Each successor state axiom of $\mathcal{D}_{ss}$ is strongly context free.
5. We are progressing with respect to α, a ground action term, and α is possible initially:

$$\mathcal{D} \models \mathit{Poss}(\alpha, S_0).$$

Notice that the single sentence (21) is logically equivalent to:

$$F(\boldsymbol{C}^{(1)}, S_0), \ldots, F(\boldsymbol{C}^{(n)}, S_0), \tag{22}$$

$$\boldsymbol{x} \neq \boldsymbol{C}^{(1)} \wedge \cdots \wedge \boldsymbol{x} \neq \boldsymbol{C}^{(n)} \supset \neg F(\boldsymbol{x}, S_0). \tag{23}$$

Notice also that, given all the positive instances (22) of F, we can trivially determine the sentence (23). So it is sufficient to *represent* a database of this form (say for computational purposes) by the set of all positive instances of F. This, we claim, is what some versions of STRIPS do (but suppressing the situation argument). This is also what relational databases do; in fact, the unique names assumption together with the condition (21) on $\mathcal{D}_{S_0}$ are the defining properties for a *relational database* (Reiter [20]). The relational *tables* are just the ground instances of the fluents F. (But bear in mind that *logically*, the database consists of the table for F, together with the axiom (23) and unique names axioms.)

As we did in the previous section, we want now to characterize the result of progressing $\mathcal{D}_{S_0}$ under the effects of the ground action α in the case of action theories of the above kind. To do so, we appeal to the results in Section 5. Consider the context free successor state axiom (20) for fluent F which we relativized to the initial situation S_0. By our assumption (21) on the syntactic form of $\mathcal{D}_{S_0}$, (20) is equivalent to:

$$\begin{aligned}
&F(\boldsymbol{x}, do(\alpha, S_0)) \equiv \\
&\boldsymbol{x} = \boldsymbol{X}^{(1)} \vee \cdots \vee \boldsymbol{x} = \boldsymbol{X}^{(m)} \vee \\
&[\boldsymbol{x} = \boldsymbol{C}^{(1)} \vee \cdots \vee \boldsymbol{x} = \boldsymbol{C}^{(n)}] \wedge \boldsymbol{x} \neq \boldsymbol{Y}^{(1)} \wedge \cdots \wedge \boldsymbol{x} \neq \boldsymbol{Y}^{(n)}.
\end{aligned}$$

Let $C^{(1)}, \ldots, C^{(r)}$ be *all* the $C^{(k)}$ that are different tuples than *all* of the $Y^{(i)}$. Then, by unique names axioms for constant symbols of sort *object*, the above sentence will be logically equivalent to

$$F(x, do(\alpha, S_0)) \equiv x = X^{(1)} \vee \cdots \vee x = X^{(m)} \vee x = C^{(1)} \vee \cdots \vee x = C^{(r)}. \tag{24}$$

Let $\mathcal{S}$ be the following set of sentences:

1. Initialize $\mathcal{S}$ to $\{\varphi \in \mathcal{D}_{S_0} \mid \varphi \text{ is situation independent}\}$.
2. For each fluent F do: add the sentence (24) to $\mathcal{S}$.

The resulting set $\mathcal{S}$ enjoys the property that $\mathcal{S} \cup \mathcal{D}_{una}$ is a progression of $\mathcal{D}_{S_0}$ under action α (Theorem 5.1). Moreover, $\mathcal{S}$ has the same syntactic form as $\mathcal{D}_{S_0}$, and so can serve as a new initial database for the purposes of iterating the above progression mechanism.

Now we interpret the above construction of the set $\mathcal{S}$ as a STRIPS operator. Imagine representing the situation dependent sentences

$$F(x, S_0) \equiv x = C^{(1)} \vee \cdots \vee x = C^{(n)} \tag{25}$$

by the situation-suppressed relational database of ground instances $F(C^{(1)}), \ldots, F(C^{(n)})$. We emphasize that this representation is merely a shorthand for the sentence (25). Now ask what sequence of deletions and additions of ground literals must be performed on $\mathcal{D}_0$, the situation-suppressed relational database version of $\mathcal{D}_{S_0}$ in order to obtain the situation-suppressed relational version of $\mathcal{S}$. The deletions and additions necessary to achieve this transformation of $\mathcal{D}_0$ to the corresponding representation of $\mathcal{S}$ will define the delete and add lists for the STRIPS operator α.

It is easy to see that the following deletions and additions, when applied to $\mathcal{D}_0$, yield the situation-suppressed, relational database representation of $\mathcal{S}$:

For each fluent F do (with reference to (20)):

1. Delete from $\mathcal{D}_0$ the sentences $F(Y^{(i)})$, $i = 1, \ldots, n$.
2. Add to $\mathcal{D}_0$ the sentences $F(X^{(i)})$, $i = 1, \ldots, m$.

It is now clear how to define a STRIPS system and its associated operator for α:[16]

1. The language $\mathcal{L}_{STRIPS}$ is the situation-suppressed version of $\mathcal{L}^2$.
2. The initial world description is $\mathcal{D}_0$.
3. For a sentence $\sigma \in \mathcal{L}_{STRIPS}$, $W \rhd \sigma$ iff $\mathcal{R}(W) \models \sigma$. Here, W is a world description in relational database form for all its fluents, i.e., the only sentences in W that mention a fluent are ground atoms of that fluent. $\mathcal{R}(W)$ is the translation of the relational database part of W to its full logical form as follows: $\mathcal{R}(W)$ consists of the sentences of W that do not mention a fluent, together with those sentences of the form

$$F(x) \equiv x = C^{(1)} \vee \cdots \vee x = C^{(n)}$$

where $F(C^{(1)}), \ldots, F(C^{(n)})$ are *all* the ground instances of a fluent F in W.
4. α's precondition is the situation-suppressed version of the right hand side of the equivalence in α's situation calculus action precondition axiom.

[16] See Section 7 for the relevant definitions.

5. For each fluent F, include in α's add and delete lists those literals specified above for obtaining the situation-suppressed relational database representation of $\mathcal{S}$.

Example 8.3. Consider the same actions, fluents and axioms as in Example 8.1, except treat this setting now as an instance of an RCF-STRIPS situation calculus specification. In this case, as before, we can "read off" the RCF-STRIPS operator schema for *move* from the instances of the successor state axioms of Example 8.1:

$move(X, Y, Z)$
P: $clear(X) \wedge clear(Z) \wedge on(X, Y) \wedge X \neq Z \wedge X \neq Y \wedge Y \neq Z$.
D: $clear(Z), on(X, Y)$.
A: $clear(Y), on(X, Z)$.

The operator description schemas for *movefromtable* and *movetotable* are obtained in the same way:

$movefromtable(X, Y)$
P: $clear(X) \wedge clear(Y) \wedge ontable(X) \wedge X \neq Y$.
D: $clear(Y), ontable(X)$.
A: $on(X, Y)$.

$movetotable(X, Y)$
P: $clear(X) \wedge on(X, Y) \wedge X \neq Y$.
D: $on(X, Y)$.
A: $clear(Y), ontable(X)$.

8.3. Pednault's ADL

The only prior literature similar to our progression semantics for STRIPS-like systems is by Pednault [16, 18]. Like us, Pednault relates a STRIPS database to the initial situation of a situation calculus axiomatization. But our interpretation of such a database, namely as a situation-suppressed situation calculus *theory*, distinguishes our approach from Pednault's, in which these databases are first-order *structures*. So for Pednault, a STRIPS operator is a mapping from first-order structures to first-order structures, where this mapping is defined by the addition and deletion of tuples applied to the relations of the structure. ADL, Pednault's generalization of STRIPS, is defined by just such a mapping between structures. For us, as for Lifschitz [11], a STRIPS operator is a mapping from first-order theories to (possibly second-order) theories, where this mapping is effected by add and delete lists of *sentences* applied to the theory. The problem with the ADL view on STRIPS is that it does not provide a feasible mechanism for applying a STRIPS operator in the case that the database is a logically incomplete theory (e.g. OCF-STRIPS of Section 8.1). For in such a case, every model of this theory must be mapped by an ADL operator into its transformed structure, and it is the set of all such transformed structures which represents the effect of the ADL operator. When there are infinitely many such models, or even when they are finite in number

but plentiful, ADL becomes an unattractive STRIPS mechanism. In contrast, our focus is on STRIPS operators that apply to logical theories, and hence operate on the single sentential representations of these many models.

9. Summary and future problems

Although progression is a widespread notion in the database and AI literatures, in its full generality it is a surprisingly complex idea. This paper has explored some of the properties of progression, and related them to STRIPS systems. Here we summarize what we take to be the main contributions of the paper.

1. We have argued the need for progressing a database, both from the perspective of STRIPS, and for the purposes of cognitive robotics.
2. We have semantically defined a notion of progression, and shown that in general, to capture it, second-order logic is required. Moreover, we have shown how to determine a second-order sentence for the progression of an arbitrary finite first- (or second-) order initial database.
3. We have explored two special cases for which progression is first order definable, namely, the case of relatively complete initial databases with arbitrary successor state axioms, and the case of a limited form of open world initial database, with context free successor state axioms. In both cases, we gave efficient procedures for computing the progression. On the other hand, as Pednault has shown [16], even for context free successor state axioms, when the initial database is an arbitrary finite first-order theory, progression need not be finitely first-order axiomatizable.
4. On our view a STRIPS operator is a *mechanism* for progressing a situation calculus theory, and its semantics can best be understood with reference to a suitable situation calculus axiomatization of actions and their effects. Under this intuition, it becomes possible to formulate various STRIPS-like systems, and prove their correctness with respect to our progression semantics. In this paper we have done just that for two different STRIPS systems (OCF- and RCF-STRIPS). In this connection OCF-STRIPS is of particular interest because it provides for a (limited) form of logical incompleteness of the database. To our knowledge, OCF-STRIPS is the only variant of STRIPS which specifically provides for an incomplete database of ground literals, and which is provably correct with respect to a logical specification.
5. It is a completely mechanical process to obtain the OCF-STRIPS operators from the action precondition and successor state axioms of a situation calculus axiomatization of some domain. Similarly for RCF-STRIPS. In other words, these purely declarative situation calculus specifications can be *compiled* into appropriate STRIPS systems. Moreover, Reiter's [21] solution to the frame problem provides an algorithm for computing the successor state axioms from the effect axioms specifying the causal laws of the domain being modeled. In other words, the axiomatizer can describe the action precondition axioms, and the domain's causal laws, and have those axioms automatically transformed into suitable STRIPS operators for that domain (assuming the successor state axioms and the initial situation have the right syntactic forms).

The results of this paper suggests a variety of topics for future research:

1. There are other cases for which progression can be done in first-order logic. One such case concerns actions with finitary effects, namely, when for every fluent, the action changes the fluent's truth value at only a finite number of instances. This and other special cases of progression need to be explored.

   In this connection, Etzioni et al. [5] have recently proposed an extension of STRIPS to accommodate *sensing* actions, i.e., actions that obtain (at plan execution time) information about the world. As Levesque [9] has observed, the resulting planner suffers from a number of limitations and drawbacks, stemming primarily from the lack of a declarative specification of their system. As it happens, a situation calculus account of sensing actions already exists (Scherl and Levesque [26]). Accordingly, it should be possible to incorporate sensing actions into our notion of progression, and use this to generalize STRIPS to include such actions. It should then be possible to prove the correctness of this version of STRIPS with respect to its progression semantics, much as we did in this paper for RCF- and OCF-STRIPS.
2. We have considered only systems that compute the full result of progression. Sometimes, for example for computational purposes, it may be better to compute only that part of the progression that is relevant to the goals of interest. For example, if our blocks world includes a fluent for the colors of blocks, then there is no need to progress this fluent if our goals have nothing to do with colors. It is still an open problem how such partial progressions can be specified and computed in a principled way.
3. The connection of RCF-STRIPS to relational databases (Section 8.2) suggests a natural generalization of STRIPS operators to allow for arbitrary relational algebra operators (not just adds and deletes) in defining the operator's effects. This can indeed be done, and an appropriate semantics is defined in terms of a situation calculus axiomatization that relaxes the context free restriction on successor state axioms of Section 8.2. In this connection, Pednault's ADL [18] provides for just such a generalized relational STRIPS, but without the relational algebra.
4. In a sense, progressing a database to S_α amounts to forgetting about the initial situation and all those situations that are reachable from S_0 but not from S_α. This view of progression leads to an interesting notion of what it means for a knowledge base to forget about some of its contents that is investigated further in (Lin and Reiter [14]).

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, Daniel Marcu, and Richard Scherl. One of the referees made a number of valuable suggestions for improving the paper. This research was funded by the Government of Canada National Sciences and Engineering Research Council, and the Institute for Robotics and Intelligent Systems.

References

[1] S. Abiteboul, Updates, a new frontier, in: *Proceedings Second International Conference on Database Theory* (Springer, New York, 1988) 1–18.

[2] F. Bacchus and Q. Yang, Downward refinement and the efficiency of hierarchical problem solving, *Artif. Intell.* **71** (1994) 41–100.

[3] E.W. Dijkstra and C.S. Scholten, *Predicate Calculus and Program Semantics* (Springer, New York, 1990).

[4] K. Erol, D. Nau and V. Subrahmanian, On the complexity of domain independent planning, in: *Proceedings AAAI-92*, San Jose, CA (1992) 381–386.

[5] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh and M. Williamson, An approach to planning with incomplete information, in: *Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1992) 115–125.

[6] R.E. Fikes and N.J. Nilsson, STRIPS: a new approach to theorem proving in problem solving, *Artif. Intell.* **2** (1971) 189–208.

[7] S.C. Kleene, *Mathematical Logic* (Wiley, New York, 1967).

[8] Y. Lespérance, H.J. Levesque, F. Lin, D. Marcu, R. Reiter and R. Scherl, Foundations of a logical approach to agent programming, in: M. Wooldridge, J. Muller and M. Tambe, eds., *Intelligent Agents Volume II—Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages (ATAL-95)* (Springer, New York, 1996) 331–346.

[9] H.J. Levesque, What is planning in the presence of sensing?, Tech. Rept., Department of Computer Science, University of Toronto, Toronto, Ont. (1995).

[10] H.J. Levesque, R. Reiter, Y. Lespérance, F. Lin and R. Scherl, GOLOG: a logic programming language for dynamic domains, *J. Logic Program.* (to appear).

[11] V. Lifschitz, On the semantics of STRIPS, in: *Proceedings Workshop on Planning and Reasoning about Action*, Timberline, OR (1986) 1–9.

[12] F. Lin and R. Reiter, How to progress a database (and why) I. Logical foundations, in: *Proceedings Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Bonn (1994) 425–436.

[13] F. Lin and R. Reiter, State constraints revisited, *J. Logic Comput.* **4** (1994) 655–678.

[14] F. Lin and R. Reiter, Forget it!, in: R. Greiner and D. Subramanian, eds., *Working Notes of AAAI Fall Symposium on Relevance* (American Association for Artificial Intelligence, Menlo Park, CA, 1994) 154–159.

[15] F. Lin and R. Reiter, How to progress a database II. The STRIPS connection, in: *Proceedings IJCAI-95*, Montreal, Que. (1995) 2001–2007.

[16] E.P. Pednault, Toward a mathematical theory of plan synthesis, PhD thesis, Department of Electrical Engineering, Stanford University, Stanford, CA (1986).

[17] E.P. Pednault, Synthesizing plans that contain actions with context dependent effects, *Comput. Intell.* **4** (1988) 356–372.

[18] E.P. Pednault, ADL: exploring the middle ground between STRIPS and the situation calculus, in: *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Ont. (1989) 324–332.

[19] P. Peppas, N. Foo and M.-A. Williams, On the expressibility of propositions, *Logique Anal.* **139–140** (1992) 251–272.

[20] R. Reiter, Towards a logical reconstruction of relational database theory, in: M. Brodie, J. Mylopoulos and J. Schmidt, eds., *On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages* (Springer, New York, 1984) 191–233.

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

[22] R. Reiter, Proving properties of states in the situation calculus, *Artif. Intell.* **64** (1993) 337–351.

[23] R. Reiter, On specifying database updates, *J. Logic Program.* **25** (1995) 53–91.

[24] R. Reiter, Some soundness and completeness results for regression in the situation calculus (to appear).

[25] S.J. Rosenschein, Plan synthesis: A logical perspective, in: *Proceedings IJCAI-81*, Milan (1981) 331–337.

[26] R. Scherl and H.J. Levesque, The frame problem and knowledge-producing actions, in: *Proceedings AAAI-93*, Washington, DC (1993).

[27] L.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. Carlson, eds., *Knowledge Representation and Defeasible Reasoning* (Kluwer Academic Publishers, Boston, MA, 1990) 23–67.

[28] R. Waldinger, Achieving several goals simultaneously, in: E.W. Elcock and D. Michie, eds., *Machine Intelligence* **8** (Ellis Horwood, Chichester, 1977) 94–136.