Artificial Intelligence 111(1–2): 277–299 (1999).

From Situation Calculus to Fluent Calculus: State Update Axioms as a Solution to the Inferential Frame Problem

MICHAEL THIELSCHER[1]
Dept. of Computer Science, Dresden University of Technology, 01062 Dresden (Germany)
e-mail: mit@inf.tu-dresden.de

Abstract. Successor state axioms provide a solution to the famous Frame Problem as far as the representational aspect is concerned. Solving in classical, monotonic logic the additional inferential Frame Problem, on the other hand, was the major motivation for the development of the Fluent Calculus a decade or so ago. Yet the expressiveness of the latter in comparison to the Situation Calculus remained a largely open question until today. In this note, we derive a novel version of the Fluent Calculus by gradually applying the principle of reification to successor state axioms in order to address the inferential Frame Problem without losing the representational merits. Our approach results in a fully mechanic method for the generation of *state update axioms* from any collection of Situation Calculus-style effect axioms for deterministic actions, provided the actions do not have potentially infinitely many effects. The axiomatization thus obtained is proved essentially equivalent to the corresponding axiomatization which uses successor state axioms.

Keywords. Cognitive Robotics, Frame Problem.

1 Introduction

Research in Artificial Intelligence aims at explaining and modeling intelligent behavior in terms of computational processes [36]. The classical approach towards this goal assumes intelligent behavior to be a result of correct reasoning on correct representations. In turn, this reasoning is understood by means of formal logic [26]. In the research area of Cognitive Robotics [21, 38], this approach to AI is applied to a crucial aspect of intelligent behavior, namely, acting in a dynamic world. The famous Frame Problem [30] inevitably arises in this context. Straightforward solutions exist only for such cases where it is possible for an intelligent agent to acquire complete information about world states. The early STRIPS approach [8] and modern efficient planning techniques such as [18], for instance, are restricted to problems in which this assumption can be made. Yet most of intelligent behavior concerns acting under partial information, and this is where the Frame Problem becomes a fundamental challenge, which, thirty years after its discovery, is still in the focus of attention [40]. While achieving the ultimate goal of Cognitive Robotics—building robots equipped with high-level cognitive functions—remains a long-term research project, advances in solving the Frame Problem have already led to a wide range of applications for modeling dynamic systems, such as progressing databases [24], dynamic diagnosis [43, 31], agent programming [22], robot control [9], and planning [11, 18, 39].

[1] On leave from Darmstadt University of Technology

Much like research in AI is concerned with both understanding and reproducing intelligent behavior, the Frame Problem comes with two facets: a *representational* one, which concerns the effort needed to specify non-effects of actions, and an *inferential* one, which concerns the effort needed to actually compute these non-effects. The Fluent Calculus, introduced in [15] and christened in [5], provides an axiomatization strategy that particularly aims at both aspects. For a long time, this calculus has been viewed exclusively as a close relative of approaches to the Frame Problem which appeal to non-classical logics, namely, linearized versions of, respectively, the connection method [2, 3] and Gentzen's sequent calculus [25]. The affinity of the Fluent Calculus and these two formalisms has been emphasized by several formal comparison results. In [12], for example, the three approaches have been proved to deliver equivalent solutions to a resource-sensitive variant of STRIPS planning [8]. Yet the Fluent Calculus possesses a feature by which it stands out against the two other frameworks: It stays entirely within classical logic. Nonetheless, the expressiveness of the Fluent Calculus in relation to the mainstream calculi, and in particular to the classical Situation Calculus [27], has not yet been convincingly elaborated. The early comparison of [4], which links the aforementioned linear connection method to the Situation Calculus, covers only a restricted form of STRIPS domains and thus concerns a mere fraction of the calculi.

The purpose of this paper is to present new approach to the Fluent Calculus where we start off from the Situation Calculus in the version where successor state axioms are used as a solution to the representational aspect of the Frame Problem [34]. We illustrate how the Fluent Calculus, or a novel version thereof, can be viewed as the result of gradually improving this approach in view of the inferential aspect but without losing its representational merits. The key is to gradually apply the principle of reification, which means to use terms instead of atoms as the formal denotation of statements. Along the path leading from successor state axioms to the Fluent Calculus lies an intermediate approach, namely, the alternative formulation of successor state axioms described by [19], in which atomic fluent formulas are reified. This alternative design inherits the representational advantages and additionally addresses the inferential Frame Problem. Yet it does so only under the important restriction that complete knowledge is available of the values of the relevant fluents in the initial situation. The Fluent Calculus can then be viewed as a further improvement in that it overcomes this restriction by carrying farther the principle of reification to conjunctions of fluents.

In the following section we illustrate by means of examples how successor state axioms can thus be reified to what we call *state update axioms*. We then present a fully mechanic method to derive state update axioms from an arbitrary collection of Situation Calculus-style effect axioms for deterministic actions, provided the actions do not have potentially infinitely many effects. As the main result, the axiomatization thus obtained is proved essentially equivalent to the corresponding axiomatization using successor state axioms. We also briefly discuss how the new Fluent Calculus allows the incorporation of two important ontological extensions, namely, non-deterministic actions and ramifications, i.e., indirect effects of actions. Our results are assessed in a concluding discussion.

2 From Situation Calculus to Fluent Calculus

2.1 From Successor State Axioms (I) ...

As an example which will be used throughout the paper, we will formalize the reasoning that led to the resolution of the following little mystery:

> A reliable witness reported that the murderer poured some milk into a cup of tea before offering it to his aunt. The old lady took a drink or two and then she suddenly fell into the armchair and died an instant later, by poisoning as has been diagnosed afterwards. According to the witness, the nephew had no opportunity to poison the tea beforehand. This proves that it was the milk which was poisoned and by which the victim was murdered.

The first and fundamental challenge of formalizing reasoning about actions is to account for the fact that most properties in the real world possess just a limited period of validity. The Situation Calculus paradigm [30] is to attach a situation argument to such unstable properties, also called *fluents*, thus limiting their range of validity to a specific situation. The performance of an action then brings about a new situation in which certain fluents may no longer hold. For the formalization of the pieces of commonsense knowledge relevant to our example story, let us use the two fluents $Poisoned(x, s)$, representing the fact that x is poisoned in situation s, and $Alive(x, s)$, representing the property of x being alive in situation s.[2] Furthermore, we need the two action terms $Mix(p, x, y)$, denoting the action carried out by agent p of mixing x into y, and $Drink(p, x)$, denoting that p drinks x. Finally, let $Do(a, s)$ be a binary function whose value denotes the situation to which leads the performance of action a in situation s. With this signature and its semantics the following effect axiom formalizes the fact that if x is poisoned in situation s then y, too, is poisoned in the situation that obtains when someone mixes x into y:

$$Poisoned(x, s) \supset Poisoned(y, Do(Mix(p, x, y), s)) \tag{1}$$

This effect axiom encodes the fact that if x is poisoned then person p ceases to being among the livings after she had drunk x:

$$Alive(p, s) \wedge Poisoned(x, s) \supset \neg Alive(p, Do(Drink(p, x), s)) \tag{2}$$

These two formulas, however, do not suffice to solve the mystery due to the Frame Problem, which has been uncovered as early as in [30]. To see why, let S_0 be a constant by which we denote the initial situation, and consider the assertion,

$$\neg Poisoned(Tea, S_0) \wedge Alive(Nephew, S_0) \wedge Alive(Aunt, S_0) \tag{3}$$

Even if we added the fact that $Poisoned(Milk, S_0)$, the intended conclusion $\neg Alive(Aunt, S_2)$ does not yet follow (where $S_2 = Do(Drink(Aunt, Tea), Do(Mix(Nephew, Milk, Tea), S_0))$), because $Alive(Aunt, Do(Mix(Nephew, Milk, Tea), S_0))$ is needed for axiom (2) to apply but cannot be derived. In order to obtain this and other intuitively expected conclusions, a number of non-effect axioms (or "frame axioms") need to be supplied, like the following, which says that people survive the mixing of substances:

$$Alive(x, s) \supset Alive(x, Do(Mix(p, y, z), s)) \tag{4}$$

Now, the Frame Problem is concerned with the problems that arise from the apparent need for non-effect axioms like (4). Actually there are two aspects to this famous problem: The *representational* Frame Problem is concerned with the proliferation of all the many frame axioms. The *inferential* Frame Problem describes the computational difficulties raised by the presence of many non-effect axioms when it comes to making inferences on the basis of an axiomatization:

[2] A word on the notation: Predicate and function symbols, including constants, start with a capital letter whereas variables are in lower case, sometimes with sub- or superscripts. Free variables in formulas are assumed universally quantified.

Suppose that for the proof of a theorem some properties were needed in situations other than the ones where they are given or obtained as effects of an action (like, e.g., the property $Alive(Aunt)$, which is given wrt. S_0 but needed in the situation $Do(Mix(Nephew, Milk, Tea), S_0)$). Then one-by-one each of these properties has to be carried from the one situation to the other, and past all intermediate situations, by means of separate frame axioms.

With regard to the representational aspect of the Frame Problem, successor state axioms [34] provide a solution which is optimal in the sense that it requires no extra frame axioms at all. The key idea is to combine, in a determined fashion, several effect axioms into a single one. The result, more complex than simple effect axioms like (1) and (2) but still mentioning solely effects, is designed in such a way that it implicitly contains sufficient information also about non-changes of fluents.

The procedure by which these axioms are set up is the following. Suppose $F(\vec{x})$ is among the fluents one is interested in. On the assumption that a fixed, finite set of actions is considered relevant, it should be possible to specify with a single formula $\gamma_F^+(\vec{x}, a, s)$ all circumstances by which $F(\vec{x})$ would be caused to become true. That is to say, $\gamma_F^+(\vec{x}, a, s)$ describes all actions a and conditions relative to situation s so that $F(\vec{x})$ is a positive effect of performing a in s. For example, among the actions we considered above there is one, and only one, by which the fluent $Poisoned(x)$ is made true, namely, mixing some poisonous y into x. Hence an adequate definition of $\gamma_{Poisoned}^+(x, a, s)$ is the formula $\exists p, y[a = Mix(p, y, x) \wedge Poisoned(y, s)]$.

A dual formula, $\gamma_F^-(\vec{x}, a, s)$, defines the circumstances by which fluent $F(\vec{x})$ is caused to become false. In our example we consider no way to 'decontaminate' a substance, which is why $\gamma_{Poisoned}^-(x, a, s)$ should be equated with a logical contradiction. For our second fluent, $Alive(x)$, the situation is just the other way round: While $\gamma_{Alive}^+(x, a, s)$ is false for any instance, the appropriate definition of $\gamma_{Alive}^-(x, a, s)$ is $\exists y[a = Drink(x, y) \wedge Alive(x, s) \wedge Poisoned(y, s)]$.

On the basis of suitable definitions for both γ_F^+ and γ_F^-, a complete account can be given of how the truth value of fluent F in a new situation depends on the old one, namely,

$$F(\vec{x}, Do(a, s)) \equiv \gamma_F^+(\vec{x}, a, s) \vee [F(\vec{x}, s) \wedge \neg \gamma_F^-(\vec{x}, a, s)] \tag{5}$$

This is the general form of *successor state axioms*.[3] It says that the fluent F holds in a new situation if, and only if, it is either a positive effect of the action being performed, or it was already true and the circumstances were not such that the fluent had to become false. Although both γ^+ and γ^- talk exclusively about effects (positive and negative), a successor state axiom, by virtue of being bi-conditional, implicitly contains all the information needed to entail any non-change of the fluent in question. For whenever neither $\gamma_F^+(\vec{x}, a, s)$ nor $\gamma_F^-(\vec{x}, a, s)$ is true, then (5) rewrites to the simple equivalence $F(\vec{x}, Do(a, s)) \equiv F(\vec{x}, s)$.

The successor state axioms for our example domain, given the formulas $\gamma_{Poisoned}^+$, $\gamma_{Poisoned}^-$, γ_{Alive}^+, and γ_{Alive}^-, respectively, from above, are

$$\begin{aligned} Poisoned(x, Do(a, s)) \equiv\ & \exists p, y\,[\, a = Mix(p, y, x) \wedge Poisoned(y, s)\,] \\ & \vee\ Poisoned(x, s) \end{aligned} \tag{6}$$

$$\begin{aligned} & Alive(x, Do(a, s)) \equiv \\ & \quad Alive(x, s) \wedge \neg\exists y\,[\, a = Drink(x, y) \wedge Alive(x, s) \wedge Poisoned(y, s)\,] \end{aligned} \tag{7}$$

[3] For the sake of clarity we ignore the concept of action precondition in this paper, as it is irrelevant for our discussion (see Section 5).

The latter, for instance, suffices to conclude that $Alive(Aunt, S_0)$ is not affected by the action $Mix(Nephew, Milk, Tea)$ —assuming "unique names" for actions, i.e., $Mix(p', x', y') \neq Drink(x, y)$. Thus we can spare the frame axiom (4).

2.2 ... via Successor State Axioms (II) ...

While successor state axioms are a good way to overcome the representational Frame Problem since no frame axioms at all are required, the inferential aspect is still present. If for a proof some properties, like $Alive(Aunt)$ above, are needed in situations other than the ones where they are given or obtained, then they have to be carried through each intermediate situation by separate instances of successor state axioms.

However, it has been shown in [19] that by formulating successor state axioms in a way that is somehow dual to the scheme (5), the inferential aspect can be addressed at least to a certain extent. Central to this alternative is the representation technique of reification. It means that properties like $Poisoned(x)$ are formally modeled as terms, that is, $Poisoned$ and all other predicate symbols denoting fluents turn into function symbols. This allows for a more flexible handling of these properties within first-order logic. Let, to this end, $Holds(f, s)$ be a binary predicate representing the fact that in situation s holds the fluent f, now formally a term but still meaning a proposition. This modification is justified by a natural correspondence between a standard Situation Calculus signature and its reified version; see Appendix A.1 for the formal details.

The key to the alternative form of successor state axioms is to devise one for each action, and not for each fluent, which gives a complete account of the positive and negative effects of that action. Suppose $A(\vec{x})$ is an action, then it should be possible to specify with a single formula $\delta_A^+(\vec{x}, f, s)$ the necessary and sufficient conditions on f and s so that f is a positive effect of performing $A(\vec{x})$ in s. In our running example, the appropriate definition of $\delta_{Mix}^+(p, x, y, f, s)$, say, is $[f = Poisoned(y, s)] \wedge Holds(Poisoned(x), s)$. A dual formula, $\delta_A^-(\vec{x}, f, s)$, defines the necessary and sufficient conditions on f and s so that f is a negative effect of performing $A(\vec{x})$ in s. For instance, $\delta_{Mix}^-(p, x, y, f, s)$ should be false in any case since $Mix(p, x, y)$ has no relevant negative effect. For a suitable axiomatization of the action $Drink(p, x)$ we equate $\delta_{Drink}^+(p, x, f, s)$ with a logical contradiction and $\delta_{Drink}^-(p, x, f, s)$ with $[f = Alive(p)] \wedge Holds(Alive(p), s) \wedge Holds(Poisoned(x), s)$.

On the basis of δ_A^+ and δ_A^-, a complete account can be given of which fluents hold in situations reached by performing A, namely,

$$Holds(f, Do(A(\vec{x}), s)) \equiv \delta_A^+(\vec{x}, f, s) \vee [\, Holds(f, s) \wedge \neg \delta_A^-(\vec{x}, f, s)\,] \tag{8}$$

That is to say, the fluents which hold after performing the action A are exactly those which are among the positive effects or which held before and are not among the negative effects. The reader may contrast this scheme with (5) and in particular observe the reversed roles of fluents and actions.

Given the formulas $\delta_{Mix}^+(p, x, y, f, s)$, $\delta_{Mix}^-(p, x, y, f, s)$, $\delta_{Drink}^+(p, x, f, s)$, and $\delta_{Drink}^+(p, x, f, s)$, respectively, from above, we thus obtain these two successor state axioms of type (II):

$$\begin{aligned} Holds(f, Do(Mix(p, x, y), s)) \equiv\ & f = Poisoned(y) \wedge Holds(Poisoned(x), s) \\ & \vee\ Holds(f, s) \end{aligned} \tag{9}$$

$$\begin{aligned} & Holds(f, Do(Drink(p, x), s)) \equiv \\ & \quad Holds(f, s) \wedge \neg [\, f = Alive(p) \wedge Holds(Alive(p), s) \wedge Holds(Poisoned(x), s)\,] \end{aligned} \tag{10}$$

Notice that as before non-effects are not explicitly mentioned and no additional frame axioms are required, so the representational aspect of the Frame Problem is addressed with the alternative notion of successor state axioms just as well. The inferential advantage of the alternative design shows if we represent the collection of fluents that are true in a situation s by equating the atomic formula $Holds(f, S)$ with the conditions on f to hold in s. The following formula, for instance, constitutes a suitable description of the initial situation in our example:

$$Holds(f, S_0) \equiv f = Alive(Nephew) \vee f = Alive(Aunt) \vee f = Poisoned(Milk) \qquad (11)$$

The crucial feature of this formula is that the situation argument, S_0, occurs only once. With this representational trick it becomes possible to obtain a complete description of a successor situation in one go, that is, by singular application of a successor state axiom. Consider, for example, the axiom which specifies the effects of mixing, (9). If we substitute p, x, and y by $Nephew$, $Milk$, and Tea, respectively, and s by S_0, then we can replace the sub-formula $Holds(f, S_0)$ of the resulting instance by the equivalent disjunction as given in axiom (11). So doing yields the formula,

$$\begin{aligned} &Holds(f, Do(Mix(Nephew, Milk, Tea), S_0)) \equiv \\ &\quad f = Poisoned(Tea) \wedge Holds(Poisoned(Milk), S_0) \\ &\quad \vee f = Alive(Nephew) \vee f = Alive(Aunt) \vee f = Poisoned(Milk) \end{aligned}$$

which all at once provides a complete description of the successor situation. Given suitable axioms for equality, the above can be simplified, with the aid of (11), to

$$\begin{aligned} Holds(f, Do(Mix(Nephew, Milk, Tea), S_0)) \equiv\ & f = Poisoned(Tea) \vee f = Alive(Nephew) \\ & \vee f = Alive(Aunt) \vee f = Poisoned(Milk) \end{aligned}$$

The reader may verify that we can likewise infer the result of $Drink(Aunt, Tea)$ in the new situation by applying the appropriate instance of successor state axiom (10), which, after simplification, yields,

$$\begin{aligned} &Holds(f, Do(Drink(Aunt, Tea), Do(Mix(Nephew, Milk, Tea), S_0))) \equiv \\ &\quad f = Poisoned(Tea) \vee f = Alive(Nephew) \vee f = Poisoned(Milk) \end{aligned}$$

The alternative design of successor state axioms provides a solution to both aspects of the Frame Problem: No frame axioms at all are needed, and one instance of a single successor state axiom suffices to carry over to the next situation all unchanged fluents. However, the proposed method of inference relies on the very strong assumption that we can supply a complete account of what does and what does not hold in the initial situation. Formula (11) provides such a complete specification, because it says that any fluent is necessarily false in S_0 which does not occur to the right of the equivalence symbol. Unfortunately it is impossible to formulate partial knowledge of the initial state of affairs in a similarly advantageous fashion. Of course one can start with an incomplete specification like, for instance,

$$Holds(f, S_0) \subset [f = Alive(Nephew) \vee f = Alive(Aunt)] \wedge f \neq Poisoned(Tea)$$

which mirrors the incomplete description we used earlier (c.f. formula (3)). But then the elegant inference step from above, where we have simply replaced a sub-formula by an equivalent, is no longer feasible. In this case one is in no way better off with the alternative notion of successor state axioms; again separate instances need to be applied, one for each fluent, in order to deduce what holds in a successor situation.

2.3 ... to State Update Axioms

So far we have used reification to denote single properties by terms. The 'meta'-predicate $Holds$ has been introduced which relates a reified fluent to a situation term, thus indicating whether the corresponding property is true in the associated situation. When formalizing collected information about a particular situation S as to which fluents are known to hold in it, the various corresponding atoms $Holds(f_i, S)$ are conjuncted using the standard logical connectives. We have seen how the inferential aspect of the Frame Problem is addressed if this is carried out in a certain way, namely, by equating $Holds(f, s)$ with some suitable formula Ψ. The effects of an action a can then be specified in terms of how Ψ modifies to some formula Ψ' such that $Holds(f, Do(a, s)) \equiv \Psi'$. We have also seen, however, that this representation technique is still not sufficiently flexible in that it is impossible to construct a first-order formula Ψ so that $Holds(f, S_0) \equiv \Psi$ provides a correct incomplete specification of S_0. Yet it is possible to circumvent this drawback by carrying farther the principle of reification, to the extent that not only single fluents but also their conjunctions are formally treated as terms. Required to this end is a binary function which to a certain extent reifies the logical conjunction. This function shall be denoted by the symbol " $\circ$ " and written in infix notation, so that, for instance, the term $Alive(Nephew) \circ Poisoned(Milk)$ is the reified version of $Alive(Nephew) \wedge Poisoned(Milk)$. The use of the function " $\circ$ " is the characteristic feature of axiomatizations which follow the paradigm of Fluent Calculus. Appendix A.2 contains the formal details of how to obtain a Fluent Calculus signature from a Situation Calculus one.

The union of all relevant fluents that hold in a situation is called the *state* (of the world) in that situation. Recall that a situation is characterized by the sequence of actions that led to it. While the world possibly exhibits the very same state in different situations,[4] the world is in a unique state in each situation. A function denoted by $State(s)$ shall relate situations s to the corresponding states, which are reified collections of fluents.

Modeling entire states as terms allows the use of variables to express mere partial information about a situation. The following, for instance, is a correct incomplete account of the initial situation S_0 in our mystery story (c.f. (3)):

$$\begin{aligned} \exists z \; [\; & State(S_0) = Alive(Nephew) \circ Alive(Aunt) \circ z \; \wedge \\ & \forall z'. \; z \neq Poisoned(Tea) \circ z' \;] \end{aligned} \tag{12}$$

That is to say, of the initial state it is known that both $Alive(Nephew)$ and $Alive(Aunt)$ are true and that possibly some other facts z hold, too—with the restriction that z must not include $Poisoned(Tea)$, of which we know it is false.

The binary function " $\circ$ " needs to inherit from the logical conjunction an important property. Namely, the order is irrelevant in which conjuncts are given. Formally, order ignorance is ensured by stipulating associativity and commutativity, that is, $\forall x, y, z. (x \circ y) \circ z = x \circ (y \circ z)$ and $\forall x, y. x \circ y = y \circ x$. It is convenient to also reify the empty conjunction, a logical tautology, by a constant, which is usually denoted by $\emptyset$ and which satisfies $\forall x. x \circ \emptyset = x$. The three equational axioms, jointly abbreviated AC1, in conjunction with the standard axioms of equality entail the equivalence of two state terms whenever they are built up from an identical collection of reified fluents.[5] In addition, denials of equalities, such as in the second part of formula (12), need to be

[4] If, for example, the tea was already poisoned initially, then the state of the world prior to and after $Mix(Nephew, Milk, Tea)$ would have been the same—in terms of which of the two liquids are poisoned and who of our protagonists is alive.

[5] The reader may wonder why function " $\circ$ " is not expected to be idempotent, i.e., $\forall x. x \circ x = x$, which is yet another property of logical conjunction. The (subtle) reason for this is given below.

derivable. This requires an extension of the standard assumption of "unique names" for fluents to uniqueness of states, denoted by *EUNA* (see Appendix A.2 for the details).

The assertion that some fluent f holds (resp. does not hold) in some situation s can be formalized by $\exists z.\ State(s) = f \circ z$ (resp. $\forall z.\ State(s) \neq f \circ z$). This allows to reintroduce the *Holds* predicate, now, however, not as part of the signature but as an abbreviation for an equality sentence:

$$Holds(f, s) \stackrel{\text{def}}{=} \exists z.\ State(s) = f \circ z \tag{13}$$

In this way, any Situation Calculus assertion about situations can be directly transferred to the Fluent Calculus; e.g., the (quite arbitrary) formula $\exists x.\, Poisoned(x, S_0) \vee \neg Alive(Aunt, S_1)$ of the Situation Calculus reads $\exists x.\, Holds(Poisoned(x), S_0) \vee \neg Holds(Alive(Aunt), S_1)$ in the Fluent Calculus. Notice that *Holds* being a mere macro in the Fluent Calculus, assertions about states become pure equality sentences.

Knowledge of effects of actions is formalized in terms of specifying how a current state modifies when moving on to a next situation. The universal form of what we call *state update axiom* is

$$\Delta(s) \supset \Gamma[State(Do(A, s)), State(s)] \tag{14}$$

where $\Delta(s)$ states conditions on s, or rather on the corresponding state, under which the successor state is obtained by modifying the current state according to Γ. Typically, condition $\Delta(s)$ is a compound formula consisting of $Holds(f, s)$ atoms, as defined with the foundational axiom (13). The component Γ defines the way the state in situation s modifies according to the effects of the action under consideration. Actions may initiate and terminate properties. We will discuss the design of Γ for these two cases in turn.

If an action has a positive effect, then the fluent f which becomes true simply needs to be coupled onto the state term via $State(Do(A, s)) = State(s) \circ f$. An example is the following axiomatization of the (conditional) effect of mixing a liquid into a second one:

$$\begin{aligned} &Holds(Poisoned(x), s) \wedge \neg Holds(Poisoned(y), s) \\ &\qquad \supset State(Do(Mix(p, x, y), s)) = State(s) \circ Poisoned(y) \\ &\neg Holds(Poisoned(x), s) \vee Holds(Poisoned(y), s) \\ &\qquad \supset State(Do(Mix(p, x, y), s)) = State(s) \end{aligned} \tag{15}$$

That is to say, if x is poisoned and y is not, then the new state is obtained from the predecessor just by adding the fluent $Poisoned(y)$, else nothing changes at all and so the two states are identical. Notice that neither of the two state update axioms mentions any non-effects.

If we substitute, in the two axioms (15), p, x, and y by *Nephew*, *Milk*, and *Tea*, respectively, and s by S_0, then we can replace the term $State(S_0)$ in both resulting instances by the equal term as given in axiom (12). So doing yields,

$$\begin{aligned} \exists z\ [\ &Holds(Poisoned(Milk), S_0) \wedge \neg Holds(Poisoned(Tea), S_0) \\ &\qquad \supset State(Do(Mix(Nephew, Milk, Tea), S_0)) \\ &\qquad\quad = Alive(Nephew) \circ Alive(Aunt) \circ z \circ Poisoned(Tea) \\ &\wedge \neg Holds(Poisoned(Milk), S_0) \vee Holds(Poisoned(Tea), S_0) \\ &\qquad \supset State(Do(Mix(Nephew, Milk, Tea), S_0)) \\ &\qquad\quad = Alive(Nephew) \circ Alive(Aunt) \circ z \\ &\wedge \forall z'.\, z \neq Poisoned(Tea) \circ z'\ ] \end{aligned}$$

which implies, using the abbreviation $S_1 = Do(Mix(Nephew, Milk, Tea), S_0)$ and the correspondence (13) along with standard properties of equality and assertion (12),

$$\begin{aligned}\exists z\ [\ & Holds(Poisoned(Milk), S_0)\\ & \supset\ State(S_1) = Alive(Nephew) \circ Alive(Aunt)\\ & \qquad \circ Poisoned(Milk) \circ Poisoned(Tea) \circ z\\ & \wedge \neg Holds(Poisoned(Milk), S_0)\\ & \supset\ State(S_1) = Alive(Nephew) \circ Alive(Aunt) \circ z\\ & \wedge \forall z'.\ z \neq Poisoned(Tea) \circ z'\ ]\end{aligned}$$

In this way we have obtained from an incomplete initial specification a still partial description of the successor state, which includes the unaffected fluents $Alive(Nephew)$ and $Alive(Aunt)$. These properties thus survived the application of the effects axioms without the need to be carried over, one-by-one, by separate application of axioms.

If an action has a negative effect, then the fluent f which becomes false needs to be withdrawn from the current state $State(s)$. The scheme $State(Do(A, s)) \circ f = State(s)$ serves this purpose. Incidentally, this scheme is the sole reason for not stipulating that "$\circ$" be idempotent. For if it were, then the equation $State(Do(A, s)) \circ f = State(s)$ would be satisfied if $State(Do(A, s))$ contained f. Hence this equation would not guarantee that f becomes false. Vital for our scheme is also to ensure that state terms do not contain any fluent twice or more, i.e.,

$$\forall s, x, z.\ State(s) = x \circ x \circ z\ \supset\ x = \emptyset \tag{16}$$

These preparatory remarks lead us to the following axiomatization of the (conditional) effect of drinking:

$$\begin{aligned}& Holds(Alive(p), s) \wedge Holds(Poisoned(x), s)\\ & \qquad \supset\ State(Do(Drink(p, x), s)) \circ Alive(p) = State(s)\\ & \neg Holds(Alive(p), s) \vee \neg Holds(Poisoned(x), s)\\ & \qquad \supset\ State(Do(Drink(p, x), s)) = State(s)\end{aligned} \tag{17}$$

That is to say, if p is alive and x is poisoned, then the new state is obtained from the predecessor just by terminating $Alive(p)$, else nothing changes at all.[6]

Applying the two axioms (17) to what we have derived about the state in situation S_1 yields, setting $S_2 = Do(Drink(Aunt, Tea), S_1)$ and performing straightforward simplifications,

$$\begin{aligned}\exists z\ [\ & Holds(Poisoned(Milk), S_0)\\ & \supset\ State(S_2) \circ Alive(Aunt) = Alive(Nephew) \circ Alive(Aunt)\\ & \qquad \circ Poisoned(Milk) \circ Poisoned(Tea) \circ z\\ & \wedge \neg Holds(Poisoned(Milk), S_0)\\ & \supset\ State(S_2) = Alive(Nephew) \circ Alive(Aunt) \circ z\ ]\end{aligned}$$

This partial description[7] of the successor state again includes every persistent fluent without having applied separate deduction steps for each. The concept of state update axioms thus

[6] Actions may of course have both positive and negative effects at the same time, in which case the component Γ of a state update axiom combines the schemes for initiating and terminating fluents. This general case is dealt with in Section 3.

[7] which, since $State(S_2) = Alive(Nephew) \circ Alive(Aunt) \circ z$ implies that $Holds(Alive(Aunt), S_2)$, leads directly to the resolution of the murder mystery: Along with the statement of the witness, $\neg Holds(Alive(Aunt), S_2)$, the formula above logically entails the explanation that $Holds(Poisoned(Milk), S_0)$.

provides a solution to both the representational and the inferential aspect of the Frame Problem which is capable of dealing with incomplete knowledge about world states.

3 The General Method

Having illustrated the design and use of state update axioms by example, in this section we will present a general, fully mechanic procedure by which is generated a suitable set of state update axioms from a given collection of Situation Calculus effect axioms, like (1) and (2). One restriction turns out necessary for our method to be feasible, namely, actions may not potentially have infinitely many effects, or so-called *open* effects. An example of a violation of this assumption is the following effect axiom, which specifies an open effect: $\forall x, y, s.\ \mathit{Bomb}(x) \wedge \mathit{Nearby}(x, y, s) \supset \mathit{Destroyed}(y, \mathit{Do}(\mathit{Explodes}(x), s))$. Even after instantiating the action expression $\mathit{Explodes}(x)$ and the situation term s, the effect literal still carries a variable, y, so that the action may have an infinite number of effects. State update axioms for actions with open effects are discussed in Section 5.

We consider a standard Situation Calculus signature, which is a many-sorted first-order language that includes the sort *sit* for situations. Positive effect specifications are of the following form, where A denotes an action and F a fluent:

$$\varepsilon^{+}_{A,F}(\vec{x}, s) \ \supset\ F(\vec{y}, \mathit{Do}(A(\vec{x}), s)) \tag{18}$$

Here, ε is a first-order formula whose free variables are among $\vec{x}, s$; and $\vec{y}$ contains only variables from $\vec{x}$. Notice that it is the very last restriction which ensures that the effect specification does not describe an open effect: Except for the situation term, all arguments of the effect F are bound by the action term $A(\vec{x})$. Likewise, negative effect specifications are of the form,

$$\varepsilon^{-}_{A,F}(\vec{x}, s) \ \supset\ \neg F(\vec{y}, \mathit{Do}(A(\vec{x}), s)) \tag{19}$$

where again ε is a first-order formula whose free variables are among $\vec{x}, s$ and where $\vec{y}$ contains only variables from $\vec{x}$. Our two effect axioms at the beginning of Section 2.1, for instance, fit this scheme, namely, by equating $\varepsilon^{+}_{\mathit{Mix},\mathit{Poisoned}}(p, x, y, s)$ with $\mathit{Poisoned}(x, s)$ and $\varepsilon^{-}_{\mathit{Drink},\mathit{Alive}}(p, x, s)$ with $\mathit{Alive}(p, s) \wedge \mathit{Poisoned}(x, s)$. We assume that a given set $\mathcal{E}$ of effect axioms is *consistent* in that for all A and F the assumption of unique names entails,

$$\neg\exists \vec{x}, s\, [\, \varepsilon^{+}_{A,F}(\vec{x}, s) \wedge \varepsilon^{-}_{A,F}(\vec{x}, s)\,] \tag{20}$$

Fundamental for any attempt to solve the Frame Problem is the assumption that a given set of effect axioms is *complete* in the sense that all relevant effects of all involved actions are specified.[8] Our concern, therefore, is to derive state update axioms from a given set of effect specifications in such a way that the assumption of completeness is suitably reflected by the resulting axioms. The following instance of scheme (14) is the general form of state update axioms for deterministic actions with only direct effects:

$$\Delta(s) \ \supset\ \mathit{State}(\mathit{Do}(A, s)) \circ \vartheta^{-} = \mathit{State}(s) \circ \vartheta^{+}$$

where ϑ^{-} are the negative effects and ϑ^{+} the positive effects, respectively, of action A under condition $\Delta(s)$. The main challenge for the design of these state update axioms is to make sure

[8] If actions have additional, indirect effects, then this gives rise to the so-called Ramification Problem; see Section 4.2.

that condition Δ is strong enough for the equation in the consequent to be sound. Neither must ϑ^+ include a fluent that already holds in situation s (for this would contradict the foundational axiom about multiple occurrences, (16)), nor should ϑ^- specify a negative effect that is already false in s (for then *EUNA* implies that the equation be false). This is the motivation behind step 1 and 2 of the procedure below. The final and main step 3 reflects the fact that actions with conditional effects require more than one state update axiom, each applying in different contexts:

1. Rewrite to $\varepsilon^+_{A,F}(\vec{x},s) \wedge \neg F(\vec{y},s) \supset F(\vec{y},Do(A(\vec{x}),s))$ each positive effect axiom of the form (18).
2. Similarly, rewrite to $\varepsilon^-_{A,F}(\vec{x},s) \wedge F(\vec{y},s) \supset \neg F(\vec{y},Do(A(\vec{x}),s))$ each negative effect axiom of the form (19).
3. For each action A, let the following $n \geq 0$ axioms be all effect axioms thus rewritten (positive and negative) concerning A:

   $$\varepsilon_1(\vec{x},s) \supset F_1(\vec{y}_1,Do(A(\vec{x}),s)),\ \ldots,\ \varepsilon_m(\vec{x},s) \supset F_m(\vec{y}_m,Do(A(\vec{x}),s))$$
   $$\varepsilon_{m+1}(\vec{x},s) \supset \neg F_{m+1}(\vec{y}_{m+1},Do(A(\vec{x}),s)),\ \ldots,\ \varepsilon_n(\vec{x},s) \supset \neg F_n(\vec{y}_n,Do(A(\vec{x}),s))$$

   For any pair of subsets $\mathcal{I}_+ \subseteq \{1,\ldots,m\}$, $\mathcal{I}_- \subseteq \{m+1,\ldots,n\}$ (including the empty ones) introduce the following state update axiom:[9]

   $$\begin{array}{c} \bigwedge_{i\in\mathcal{I}_+\cup\mathcal{I}_-} HOLDS(\varepsilon_i(\vec{x},s)) \wedge \bigwedge_{j\notin\mathcal{I}_+\cup\mathcal{I}_-} HOLDS(\neg\varepsilon_j(\vec{x},s)) \\ \supset State(Do(A(\vec{x}),s)) \circ \vartheta^{\mathcal{I}_-} = State(s) \circ \vartheta^{\mathcal{I}_+} \end{array} \tag{21}$$

   where $\vartheta^{\mathcal{I}_-}$ is the term $F_1 \circ \ldots \circ F_k$ if $\{F_1,\ldots,F_k\} = \{F_i(\vec{y}_i) : i \in \mathcal{I}_-\}$ and, similarly, $\vartheta^{\mathcal{I}_+}$ is the term $F_1 \circ \ldots \circ F_k$ if $\{F_1,\ldots,F_k\} = \{F_i(\vec{y}_i) : i \in \mathcal{I}_+\}$.[10]

Step 3 blindly considers all combinations of positive and negative effects. Some of the state update axiom thus obtained may have inconsistent antecedent, in which case they can be removed. To illustrate the interaction of context-dependent positive and negative effects, we apply our procedure to two effect axioms which encode the Yale Shooting scenario [14]:[11]

$$\begin{array}{c} Loaded(x,s) \supset Dead(y,Do(Shoot(x,y),s)) \\ true \supset \neg Loaded(x,Do(Shoot(x,y),s)) \end{array}$$

After rewriting according to steps 1 and 2, step 3 produces four state update axioms, viz.

$$\begin{array}{l} \neg[\,Holds(Loaded(x),s) \wedge \neg Holds(Dead(y),s)\,] \wedge \neg[\,true \wedge Holds(Loaded(x),s)\,] \\ \quad \supset State(Do(Shoot(x,y),s)) \circ \emptyset = State(s) \circ \emptyset \\ \neg[\,Holds(Loaded(x),s) \wedge \neg Holds(Dead(y),s)\,] \wedge true \wedge Holds(Loaded(x),s) \\ \quad \supset State(Do(Shoot(x,y),s)) \circ Loaded(x) = State(s) \circ \emptyset \\ Holds(Loaded(x),s) \wedge \neg Holds(Dead(y),s) \wedge \neg[\,true \wedge Holds(Loaded(x),s)\,] \\ \quad \supset State(Do(Shoot(x,y),s)) \circ \emptyset = State(s) \circ Dead(y) \\ Holds(Loaded(x),s) \wedge \neg Holds(Dead(y),s) \wedge true \wedge Holds(Loaded(x),s) \\ \quad \supset State(Do(Shoot(x,y),s)) \circ Loaded(x) = State(s) \circ Dead(y) \end{array}$$

[9] Below, we use the notation $HOLDS(\Psi)$ to denote the formula that results from transforming a Situation Calculus-style formula Ψ into its reified counterpart, which is obtained by replacing each fluent atom $P(\vec{\tau},\sigma)$ by $Holds(P(\vec{\tau}),\sigma)$.

[10] Thus $\vartheta^{\mathcal{I}_-}$ contains the negative effects and $\vartheta^{\mathcal{I}_+}$ the positive effects specified in the update axiom. If either set is empty then the respective term is the unit element, $\emptyset$.

[11] Below, the fluents $Loaded(x)$ and $Dead(y)$ mean that gun x is loaded and animal y is dead, respectively, and $Shoot(x,y)$ represents the action of shooting with x at y.

Logical simplification of the premises of the first two axioms yields,

$$\begin{aligned} \neg Holds(Loaded(x), s) &\supset State(Do(Shoot(x, y), s)) = State(s) \\ Holds(Dead(y), s) \wedge Holds(Loaded(x), s) &\supset State(Do(Shoot(x, y), s)) \circ Loaded(x) = State(s) \end{aligned}$$

The third axiom can be abandoned because of an inconsistent antecedent, while the fourth axiom simplifies to

$$\begin{aligned} &Holds(Loaded(x), s) \wedge \neg Holds(Dead(y), s) \\ &\quad \supset State(Do(Shoot(x, y), s)) \circ Loaded(x) = State(s) \circ Dead(y) \end{aligned}$$

(The interested reader may verify that applying the general procedure to our effect axioms (1) and (2) yields four axioms which, after straightforward simplification, turn out to be (15) and (17), respectively.)

The following primary theorem for the Fluent Calculus shows that the resulting set of state update axioms correctly reflects the effect axioms if the fundamental completeness assumption is made (see the appendix for a proof).

Theorem 1 *Consider a finite set $\mathcal{E}$ of effect axioms which complies with the assumption of consistency (c.f. (20)), and let SUA be the set of state update axioms for $\mathcal{E}$. Suppose $\mathcal{M}$ is a model of $SUA \cup \{(13), (16)\} \cup EUNA$,[12] and consider a fluent term $F(\vec{\tau})$, an action term $A(\vec{\rho})$, and a situation term σ. Then $\mathcal{M} \models Holds(F(\vec{\tau}), Do(A(\vec{\rho}), \sigma))$ iff*

1. *$\mathcal{M} \models \varepsilon^+_{A,F}(\vec{\rho}, \sigma)$, for the instance $\varepsilon^+_{A,F}(\vec{\rho}, \sigma) \supset F(\vec{\tau}, Do(A(\vec{\rho}), \sigma))$ of some axiom in $\mathcal{E}$;*
2. *or $\mathcal{M} \models Holds(F(\vec{\tau}), \sigma)$ and there is no instance $\varepsilon^-_{A,F}(\vec{\rho}, \sigma) \supset \neg F(\vec{\tau}, Do(A(\vec{\rho}), \sigma))$ of an axiom in $\mathcal{E}$ such that $\mathcal{M} \models \varepsilon^-_{A,F}(\vec{\rho}, \sigma)$.*

Our theorem coincides with the main result of [34] concerning the collection of successor state axioms obtained from a given set of positive and negative effect specifications. This shows that provided no open effects occur, the state update axioms, by which is solved the additional inferential Frame Problem, express essentially the same as successor state axioms.

4 Extensions

In the following we briefly discuss how the concept of state update axioms is amenable to two important ontological extensions, namely, nondeterministic actions and ramifications.

4.1 Nondeterministic Actions

Nondeterministic actions can be very elegantly specified by means of *disjunctive* state update axioms $\Delta(s) \supset \Gamma[State(Do(A, s)), State(s)]$, where Γ is a disjunction of the possible effects, i.e., state updates, of the respective action. The following two axioms, for instance, specify the

[12] Recall that *EUNA*, the extended unique names assumption, axiomatizes equality and inequality of terms with the function " $\circ$ ".

alternative outcomes of tossing a coin x in terms of the fluent $Heads(x)$:

$$\begin{aligned}
\neg Holds(Heads(x), s) \supset\ & State(Do(Toss(x), s)) = State(s) \circ Heads(x) \\
& \vee \\
& State(Do(Toss(x), s)) = State(s) \\
Holds(Heads(x), s) \supset\ & State(Do(Toss(x), s)) \circ Heads(x) = State(s) \\
& \vee \\
& State(Do(Toss(x), s)) = State(s)
\end{aligned}$$

That is, if $Heads(x)$ is false in situation s, then it may or may not become true by performing $Toss(x)$; whereas if it is true, then it may or may not become false. Generally, disjunctive state update axioms have the same flavor as disjunctions of successor state axioms to encode nondeterministic actions in the Situation Calculus [23].

4.2 Ramifications

The Ramification Problem [10] denotes the problem of handling indirect effects of actions. These effects are not explicitly represented in action specifications but follow from general laws, so-called state constraints, describing dependencies among fluents. Approaches based on the idea of causal propagation [42, 35] are the most general solution to the Ramification Problem known today. The theory of causal relationships [42, 44] furnishes a ready approach to accommodate indirect effects in state update axioms. As an example, we consider the extension of the Yale Shooting domain by the state constraint $Walking(y) \supset \neg Dead(y)$. The state constraint itself is straightforwardly axiomatized as,

$$Holds(Walking(y), s) \supset \neg Holds(Dead(y), s) \tag{22}$$

As argued in [1], this constraint gives rise to the indirect effect that the turkey stops walking as soon as it is shot. More precisely, if both $Walking(Turkey)$ and $\neg Dead(Turkey)$ happen to be true when an action is performed which causes $Dead(Turkey)$, then this action additionally causes $\neg Walking(Turkey)$. Such indirect effects can be accounted for by the successive application of so-called causal relationships. Their axiomatization is based on defining a predicate $Causes(state, effects, new\_state, new\_effects)$, which means that in the current state $state$ the occurred effects $effects$ give rise to an additional, indirect effect resulting in the updated state new_state and the updated current effects $new\_effects$. In this way, the indirect effect in the example is accommodated via the following definition:

$$\begin{aligned}
Causes(state, effects, new_state, new_effects) \equiv\ & \exists z.\ effects = Dead(y) \circ z \wedge \\
& new_state \circ Walking(y) = state \wedge \\
& new_effects = effects \circ -Walking(y)
\end{aligned}$$

where a sub-term $-F$ represents the occurrence of a negative effect. From this definition we can derive, for instance, that whenever the turkey is dead but still walking after an action has occurred with the effects $-Loaded(Gun)$ and $Dead(Turkey)$, then $-Walking(Turkey)$ is additionally caused; that is, formally,

$$\begin{aligned}
Causes(&Dead(Turkey) \circ Walking(Turkey) \circ z,\ -Loaded(Gun) \circ Dead(Turkey), \\
&Dead(Turkey) \circ z,\ -Loaded(Gun) \circ Dead(Turkey) \circ -Walking(Turkey))
\end{aligned}$$

State update axioms which account for indirect effects are of the form,

$$\Delta(s) \supset [\, z \circ \vartheta^- = State(s) \circ \vartheta^+ \supset Ramify(z, -\vartheta^- \circ \vartheta^+, State(Do(A, s)))\,] \tag{23}$$

where

- ϑ^- are the negative direct effects;
- ϑ^+ are the positive direct effects;
- $Ramify(state, effects, new\_state)$ means that the successive application of (zero or more) causal relationships to state $state$ and effects $effects$ results in state new_state.

As in [42], the definition of the predicate $Ramify$ requires a standard second-order axiom to characterize the reflexive and transitive closure of $Causes$. The axioms which encode the underlying state constraints, such as (22), guarantee that the overall resulting state, $State(Do(A, s))$ in (23), satisfies all constraints. In this way, the state update axiom,

$$\begin{aligned} &Holds(Loaded(x), s) \wedge \neg Holds(Dead(y), s) \\ &\quad \supset\ z \circ Loaded(x) = State(s) \circ Dead(y) \\ &\qquad \supset\ Ramify(z, \text{–}Loaded(x) \circ Dead(y), State(Do(Shoot(x, y), s))) \end{aligned}$$

in conjunction with the axioms above, including the definition of $Ramify$ in terms of $Causes$, entails $Holds(Loaded(Gun), S_0) \supset \neg Holds(Walking(Turkey), Do(Shoot(Gun, Turkey), S_0))$.

5 Discussion

Starting with the concept of successor state axioms as a solution to the representational aspect of the Frame Problem, we have derived a novel version of the Fluent Calculus by gradually applying the principle of reification in order to solve the inferential Frame Problem. The intermediate approach, an alternative form of successor state axioms, roots in the axiomatization technique of [20] in a similar way as the foundations for the original successor state axioms were laid in [13, 37]. In fact, the conjunction of all axioms (8) for a domain plus a complete specification of an initial state in the flavor of axiom (11) is equivalent to Clark's completion [7] of the corresponding logic programming clauses used in [20]. A restricted version of the second form of successor state axioms has previously been used in [28] to axiomatize STRIPS domains using the Situation Calculus. The version of the Fluent Calculus we arrived at in this paper differs considerably from its roots [15], e.g. in that it exploits the full expressive power of first-order logic. In so doing it is much closer to the variant introduced in [42], but still novel is the notion of state update axioms. In particular the new function $State(s)$ lends more elegance to effect specifications and at the same time emphasizes the relation to the Situation Calculus.

We have demonstrated the expressive power of state update axioms by proving that, much like in [34], a suitable collection of these axioms can be automatically derived from a complete (wrt. the relevant fluents and actions) set of single effect axioms, provided actions have no open effects. A state update axiom formalizes an equational relation between the states at two consecutive situations. These equations being perfectly symmetric, a state update axiom can be used equally for reasoning forward and backward in time. The versatility of the Fluent Calculus promises it to be applicable to more complex problems of reasoning about actions. We have substantiated this expectation by illustrating how the concept of state update axioms is applicable to problems which involve nondeterministic actions and actions with indirect effects.

The problem of action preconditions has been ignored in this paper for the sake of clarity. Their dealing with requires no special treatment in the new Fluent Calculus since each Situation Calculus assertion about what holds in a situation corresponds directly to a Fluent Calculus

assertion via the fundamental relation (13). The restriction to actions without open effects, on the other hand, is inevitable if one aims at an explicit description of the direct effects. Open effects can only implicitly be specified in state update axioms, as is done in this example axiom:

$$\forall s, x, \vartheta^+.\ \mathit{Bomb}(x) \supset \left[\forall f, y \left[\begin{array}{c} f = \mathit{Destroyed}(y) \wedge \mathit{Holds}(\mathit{Nearby}(x, y), s) \wedge \neg \mathit{Holds}(f, s) \\ \equiv\ \exists z.\, \vartheta^+ = f \circ z \\ \supset\ \mathit{State}(\mathit{Do}(\mathit{Explodes}(x), s)) = \mathit{State}(s) \circ \vartheta^+ \end{array} \right] \right]$$

in which the positive effect of the action, ϑ^+, is defined rather than explicitly given. It lies in the nature of open effects that a suitable state update axiom provides just an implicit definition of the required update and so does no longer solve the inferential Frame Problem at the point where such an action is performed—but of course it still covers the representational aspect.

Moving from Situation Calculus to Fluent Calculus involves the introduction of the equational theory AC1. While the simple addition of equality axioms may constitute a considerable handicap for theorem proving, a variety of efficient constraint solving algorithms have been developed for the particular equational theory needed for the function "$\circ$"; for a survey see [32]. Solving the inferential Frame Problem by means of state update axioms relies on the fact that the latter always cover the entire change an action causes. The tradeoff is that the number of update axioms is, in the worst case, exponentially larger than the number of single effect axioms. However, this is perfectly acceptable as long as we can assume that actions have very few effects compared to the overall number of fluents. This assumption, in fact, plays an almost axiomatic role in commonsense reasoning about actions:

> Yet, in practice, people teach each other rather quickly what actions normally do to the world, people predict the consequences of any given action without much hustle, and AI researchers are writing languages for actions as if it is a God given truth that action representations should be compact, elegant and meaningful. Why? [...] Because the actions we normally invoke in common reasoning tasks are *local surgeries*. The world consists of a huge number of autonomous and invariant linkages [...], each corresponding to a physical process that constrains the behavior of relatively small groups of variables. ([33], p. 3.)

The essential motivation for using the Fluent Calculus is that state update axioms provide a solution not only to the representational Frame Problem, since they talk exclusively about effects, but also to the inferential Frame Problem, since one such axiom always specifies the entire relation between two consecutive situations. The inferential aspect of the Frame Problem concerns each single fluent needed by the proof of a theorem in a situation other than the one where it is given or arises. In the Situation Calculus, for instance, this holds regardless of whether successor state axioms are used for reasoning forward in time or as a basis for regression [34]. If all fluents are needed in exactly the situations they are given or obtained, then the inferential Frame Problem causes no computational burden at all and the presence of *EUNA* in the Fluent Calculus only adds to the computational complexity. However, the more fluents need to be carried unchanged through many intermediate situations, the more valuable can be a solution to the inferential Frame Problem. It remains a topic for future research to determine for which problem classes and precisely to what extent the solution to the inferential Frame Problem offered by the Fluent Calculus leads to gains in efficiency when modeling dynamic systems. This includes the hope that the Fluent Calculus can be employed to improve current techniques of planning by local search along the line of [18], which just as well have to cope with the Frame Problem and, to this end, employ techniques on which also successor state axioms are based.

Acknowledgments. The author wants to thank Steffen Hölldobler, Ray Reiter, and two anonymous reviewers for helpful comments and discussions on this paper.

References

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

[2] Wolfgang Bibel. A deductive solution for plan generation. *New Generation Computing*, 4:115–132, 1986.

[3] Wolfgang Bibel. Let's plan it deductively! *Artificial Intelligence*, 103(1–2):183–208, 1998.

[4] Wolfgang Bibel, Luis Fariñas del Cerro, Bertram Fronhöfer, and Andreas Herzig. Plan generation by linear proofs: on semantics. In *Proceedings of the German Workshop on Artificial Intelligence*, pages 49–62. Springer, Informatik Fachberichte 216, 1989.

[5] Sven-Erik Bornscheuer and Michael Thielscher. Explicit and implicit indeterminism: Reasoning about uncertain and contradictory specifications of dynamic systems. *Journal of Logic Programming*, 31(1–3):119–155, 1997.

[6] Hans-Jürgen Bürckert, Alexander Herold, Deepak Kapur, Jörg H. Siekmann, Mark E. Stickel, M. Tepp, and H. Zhang. Opening the AC-unification race. *Journal of Automated Reasoning*, 4:465–474, 1988.

[7] Keith L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.

[8] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[9] Giuseppe De Giacomo, Ray Reiter, and Mikhail Soutchanski. Execution monitoring of high-level robot programs. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 453–464, Trento, Italy, June 1998. Morgan Kaufmann.

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

[11] Cordell Green. Theorem proving by resolution as a basis for question-answering systems. *Machine Intelligence*, 4:183–205, 1969.

[12] Gerd Große, Steffen Hölldobler, and Josef Schneeberger. Linear Deductive Planning. *Journal of Logic and Computation*, 6(2):233–262, 1996.

[13] Andrew R. Haas. The case for domain-specific frame axioms. In F. M. Brown, editor, *The Frame Problem in Artificial Intelligence*, pages 343–348, Los Altos, CA, 1987. Morgan Kaufmann.

[14] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[15] Steffen Hölldobler and Josef Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8:225–244, 1990.

[16] Steffen Hölldobler and Michael Thielscher. Computing change and specificity with equational logic programs. *Annals of Mathematics and Artificial Intelligence*, 14(1):99–133, 1995.

[17] Joxan Jaffar, Jean-Louis Lassez, and Michael J. Maher. A theory of complete logic programs with equality. *Journal of Logic Programming*, 1(3):211–223, 1984.

[18] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In B. Clancey and D. Weld, editors, *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 1194–1201, Portland, OR, August 1996. MIT Press.

[19] Hesham Khalil. *Formalizing the Effects of Actions in First Order Logic.* Diplomarbeit, Intellectics, Department of Computer Science, Darmstadt University of Technology, 1996.

[20] Robert Kowalski. *Logic for Problem Solving*, volume 7 of *Artificial Intelligence Series.* Elsevier, 1979.

[21] Yves Lespérance, Hector J. Levesque, Fangzhen Lin, D. Marcu, Ray Reiter, and Richard B. Scherl. A logical approach to high-level robot programming—a progress report. In B. Kuipers, editor, *Control of the Physical World by Intelligent Agents, Papers from the AAAI Fall Symposium*, pages 109–119, New Orleans, LA, November 1994.

[22] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1–3):59–83, 1997.

[23] Fangzhen Lin. Embracing causality in specifying the indeterminate effects of actions. In B. Clancey and D. Weld, editors, *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 670–676, Portland, OR, August 1996. MIT Press.

[24] Fangzhen Lin and Ray Reiter. How to progrss a database. *Artificial Intelligence*, 92:131–167, 1997.

[25] M. Masseron, Christophe Tollu, and Jacqueline Vauzielles. Generating plans in linear logic I. Actions as proofs. *Journal of Theoretical Computer Science*, 113:349–370, 1993.

[26] John McCarthy. Programs with Common Sense. In *Proceedings of the Symposium on the Mechanization of Thought Processes*, volume 1, pages 77–84, London, November 1958. (Reprinted in: [29]).

[27] John McCarthy. *Situations and Actions and Causal Laws.* Stanford Artificial Intelligence Project, Memo 2, 1963.

[28] John McCarthy. Formalization of STRIPS in situation calculus. Technical report, Formal Reasoning Group, Department of Computer Science, Stanford University, 1985.

[29] John McCarthy. *Formalizing Common Sense.* Ablex, 1990. (Edited by V. Lifschitz).

[30] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[31] Sheila A. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 167–177, Trento, Italy, June 1998. Morgan Kaufmann.

[32] Leszek Pacholski and Andreas Podelski. Set constraints: a pearl in research on constraints. In G. Smolka, editor, *Proceedings of the International Conference on Constraint Programming (CP)*, volume 1330 of *LNCS*, pages 549–561. Springer, 1997.

[33] Judea Pearl. Causation, action, and counterfactuals. Technical Report R-223-T, Computer Science Department, University of California at Los Angeles, February 1996. (Published in *Proceedings of the Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 51–73, March 1996.).

[34] Ray Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 359–380. Academic Press, 1991.

[35] Erik Sandewall. Assessments of ramification methods that use static domain constraints. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 99–110, Cambridge, MA, November 1996. Morgan Kaufmann.

[36] Robert J. Schalkoff. *Artificial Intelligence: An Engineering Approach.* McGraw-Hill, 1990.

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

[38] Murray Shanahan. Robotics and the common sense informatic situation. In W. Wahlster, editor, *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 684–688, Budapest, Hungary, August 1996. John Wiley.

[39] Murray Shanahan. Event calculus planning revisited. In *Proceedings of the European Conference on Planning (ECP)*, volume 1348 of *LNAI*, pages 290–402. Springer, 1997.

[40] Murray Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press, 1997.

[41] John C. Shepherdson. SLDNF-resolution with equality. *Journal of Automated Reasoning*, 8:297–306, 1992.

[42] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[43] Michael Thielscher. A theory of dynamic diagnosis. *Electronic Transactions on Artificial Intelligence*, 1(4):73–104, 1997.

[44] Michael Thielscher. Reasoning about actions: Steady versus stabilizing state constraints. *Artificial Intelligence*, 104:339–355, 1998.

A Technical details

Appendix A.1 contains the formal underpinnings and justification for the alternative form of successor state axioms of Section 2.2. Appendix A.2 contains the formal underpinnings of the Fluent Calculus and the proof of the primary theorem.

A.1 Reification (I)

Let Σ be a standard signature for Situation Calculus, that is, a many-sorted first-order language with equality which includes the special sort *sit* for situations. Some predicate symbols in Σ are fluent denotations; these are of arity ≥ 1 with the last argument being of sort *sit*. The corresponding *reification* signature Σ_{Holds} is obtained from Σ by

1. replacing each $n+1$-place predicate symbol which denotes a fluent and whose argument is of sort $sorts \times sit$ by an n-place function symbol whose argument is of sort *sorts*;
2. adding a sort *fluent*, to which belong all well-sorted terms with leading function symbol obtained in step 1;
3. adding the binary predicate *Holds*, whose argument is of sort $fluent \times sit$.

The additional sort *fluent* in signature Σ_{Holds} allows formulas like $\exists f.\, Holds(f, S_0)$, which have no natural correspondence in Σ. We will discard such cases and only consider what we call *conventional* formulas over Σ_{Holds}, in which no variables of sort *fluent* occur. Then each formula ϕ over Σ has a natural, conventional counterpart over Σ_{Holds}, which is obtained by replacing, in ϕ, each fluent atom $P(\vec{\tau}, \sigma)$ by $Holds(P(\vec{\tau}), \sigma)$. Conversely, each conventional ψ over Σ_{Holds} corresponds to a formula over Σ.

This way of reifying fluents is justified (a) by the observation that each interpretation for the original signature Σ corresponds to some interpretation for the modified Σ_{Holds} and vice versa, and (b) by the fact that corresponding interpretations interpret corresponding formulas alike. To prove this formally, we define two interpretations $\mathcal{I}, \zeta$ and $\mathcal{J}, \xi$ for Σ and Σ_{Holds}, respectively, as *corresponding* iff the following holds:

1. The domains of $\mathcal{I}$ and $\mathcal{J}$ are identical except for arbitrary domain elements of sort *fluent* additionally contained in the domain of $\mathcal{J}$.
2. The variable assignments ζ and ξ are identical except for arbitrary additional assignments to variables of sort *fluent* in ξ.
3. $\mathcal{I}$ and $\mathcal{J}$ agree on the interpretation of the predicate symbols in Σ which do not denote a fluent.
4. $\mathcal{I}$ and $\mathcal{J}$ agree on the interpretation of the function symbols in Σ.
5. Whenever P is an $n+1$-place fluent denotation in Σ and $\vec{d}, d_{n+1}$ are domain elements of the right sort (that is, in particular, d_{n+1} is of sort *sit*), then the following holds: Let d be the domain element of sort *fluent* which results from applying to $\vec{d}$ the n-place function by which $\mathcal{J}$ interprets P. Then $(\vec{d}, d_{n+1}) \in P^{\mathcal{I}}$ iff $(d, d_{n+1}) \in Holds^{\mathcal{J}}$.

Now, suppose $\mathcal{I}, \zeta$ is an interpretation for Σ, then a corresponding interpretation $\mathcal{J}, \xi$ for Σ_{Holds} can be obtained by adding a domain element of sort *fluent* for each $n+1$-place fluent denotation P and each n-tuple $\vec{d}$ of the right sort, and by setting the functional assignment $P^{\mathcal{J}}$ and the relation $Holds^{\mathcal{J}}$ in the right way. Conversely, each interpretation $\mathcal{J}, \xi$ for Σ_{Holds} can be mapped onto a corresponding interpretation $\mathcal{I}, \zeta$ for Σ. The significance of this observation lies in the fact that corresponding interpretations interpret corresponding formulas alike:

Proposition 2 *Let Σ be a signature and Σ_{Holds} the corresponding reification signature. Let $\mathcal{I}, \zeta$ be an interpretation for Σ and $\mathcal{J}, \xi$ an interpretation for Σ_{Holds} so that the two correspond. Furthermore, let ϕ be formula over Σ and ψ a conventional formula over Σ_{Holds} so that the two correspond. Then $\mathcal{I}, \zeta \models \phi$ iff $\mathcal{J}, \xi \models \psi$.*

Proof: If A is a non-fluent atom over Σ, then $\mathcal{I}, \zeta \models A$ iff $\mathcal{J}, \xi \models A$ since the two interpretations agree on all relevant variable assignments, on all non-fluent predicate symbols, and on all function symbols in Σ. If $P(\vec{\tau}, \sigma)$ is a fluent atom of Σ, then $\mathcal{I}, \zeta \models P(\vec{\tau}, \sigma)$ iff $\mathcal{J}, \xi \models Holds(P(\vec{\tau}), \sigma)$ according to the definition of corresponding interpretations. Likewise, if $Holds(P(\vec{\tau}), \sigma)$ is a conventional atom over Σ_{Holds}, then $\mathcal{I}, \zeta \models P(\vec{\tau}, \sigma)$ iff $\mathcal{J}, \xi \models Holds(P(\vec{\tau}), \sigma)$. With these three base cases the claim follows by straightforward structural induction. ∎

It is an immediate consequence of these results that a set of formulas over Σ entails a formula over this signature if and only if the set of corresponding formulas over Σ_{Holds} entails the corresponding formula, and vice versa provided only conventional formulas over Σ_{Holds} are considered.

A.2 Reification (II)

Let Σ be a standard signature for Situation Calculus as above. The corresponding *Fluent Calculus* signature $\Sigma_{\mathcal{FC}}$ is obtained from Σ by

1. replacing each $n+1$-place predicate symbol which denotes a fluent and whose argument is of sort $sorts \times sit$ by an n-place function symbol whose argument is of sort $sorts$;
2. adding the constant $\emptyset$ and the binary function symbol $\circ$;
3. adding a sort *fluent*, to which belong all well-sorted terms with leading function symbol obtained in step 1, and a sort *state*, which is the least set to which belong the constant $\emptyset$, each *fluent*, and each $t_1 \circ t_2$ where t_1, t_2 are of sort *state*;
4. adding the unary function *State*, whose argument is of sort *sit*.

Notice that a Fluent Calculus signature does not include the predicate *Holds*, which is merely a macro standing for an equality sentence (c.f. (13)).

Fundamental for any Fluent Calculus axiomatization is the axiom set *EUNA* (the *extended unique names-assumption*) [16]. Its definition relies on a complete AC1-unification algorithm (see, e.g., [6]), and it comprises the following equational axioms:

1. The axioms AC1 for $\circ$ and $\emptyset$,

$$\begin{aligned} (x \circ y) \circ z &= x \circ (y \circ z) \\ x \circ y &= y \circ x \\ x \circ \emptyset &= x \end{aligned}$$

All variables are universally quantified.

2. For any two terms t_1 and t_2 of sort other than *state* and with variables $\vec{x}$,

   (a) if t_1 and t_2 are not unifiable, then

   $$\neg\exists\vec{x}.\ t_1 = t_2$$

   (b) if t_1 and t_2 are unifiable with *mgu* θ, then

   $$\forall\vec{x}\ [\ t_1 = t_2\ \supset\ \exists\vec{y}.\ \theta_=\ ]$$

   where $\vec{y}$ denotes the variables which occur in $\theta_=$ but not in $\vec{x}$.[13]

3. For any two terms t_1 and t_2 of sort *state* and with variables $\vec{x}$,

   (a) if t_1 and t_2 are not AC1-unifiable, then

   $$\neg\exists\vec{x}.\ t_1 = t_2$$

   (b) if t_1 and t_2 are AC1-unifiable with the complete set of unifiers $cU_{\mathrm{AC1}}(t_1, t_2)$, then

   $$\forall\vec{x}\ \left[\ t_1 = t_2\ \supset \bigvee_{\theta\in cU_{\mathrm{AC1}}(t_1,t_2)} \exists\vec{y}.\ \theta_=\ \right]$$

   where $\vec{y}$ denotes the variables which occur in $\theta_=$ but not in $\vec{x}$.

The axioms of item 3, in conjunction with the standard uniqueness of names-assumption in item 2, ensure that *EUNA* is *unification complete* [17, 41] wrt. state terms and the equational theory AC1. The latter axiomatizes the arbitrary re-arranging of the fluent terms that occur in a state term; hence, the following observation, which will be needed below, is a consequence of *EUNA* being AC1-unification complete:

Observation 3 *Let $\mathcal{I}, \zeta$ be an interpretation for $\Sigma_{\mathcal{FC}}$ such that $\mathcal{I}, \zeta \models EUNA$, and consider two state terms t_1 and t_2. Then $\mathcal{I}, \zeta \models \exists z.\ t_1 = t_2 \circ z$ iff each fluent term occurs $n \geq m$-times in t_1 if it occurs $m \geq 1$-times in t_2.*

A.3 Proof of Theorem 1

We are now prepared to prove our main result.

Theorem 1 *Consider a finite set $\mathcal{E}$ of effect axioms which complies with the assumption of consistency (c.f. (20)), and let SUA be the set of state update axioms for $\mathcal{E}$. Suppose $\mathcal{M}$ is a model of $SUA \cup \{(13), (16)\} \cup EUNA$, and consider a fluent term $F(\vec{\tau})$, an action term $A(\vec{\rho})$, and a situation term σ. Then $\mathcal{M} \models Holds(F(\vec{\tau}), Do(A(\vec{\rho}), \sigma))$ iff*

1. *$\mathcal{M} \models \varepsilon^+_{A,F}(\vec{\rho}, \sigma)$, for the instance $\varepsilon^+_{A,F}(\vec{\rho}, \sigma) \supset F(\vec{\tau}, Do(A(\vec{\rho}), \sigma))$ of some axiom in $\mathcal{E}$;*
2. *or $\mathcal{M} \models Holds(F(\vec{\tau}), \sigma)$ and there is no instance $\varepsilon^-_{A,F}(\vec{\rho}, \sigma) \supset \neg F(\vec{\tau}, Do(A(\vec{\rho}), \sigma))$ of an axiom in $\mathcal{E}$ such that $\mathcal{M} \models \varepsilon^-_{A,F}(\vec{\rho}, \sigma)$.*

[13] By $\theta_=$ we denote the equational formula $x_1 = r_1 \wedge \ldots \wedge x_n = r_n$ constructed from the substitution $\theta = \{x_1 \mapsto r_1, \ldots, x_n \mapsto r_n\}$.

Proof. According to (13), $\mathcal{M} \models \mathit{Holds}(F(\vec{\tau}), \mathit{Do}(A(\vec{\rho}), \sigma))$ stands for

$$\mathcal{M} \models \exists z\, [\, \mathit{State}(\mathit{Do}(A(\vec{\rho}), \sigma)) = F(\vec{\tau}) \circ z\,]$$

From Observation 3 it follows that the latter holds iff $F(\vec{\tau})$ is contained in $\mathit{State}(\mathit{Do}(A(\vec{\rho}), \sigma))$. The state update axioms (21) in SUA for action $A(\vec{x})$ are designed in such a way that each distribution of truth values of the members of $\{\mathit{HOLDS}(\varepsilon_i(\vec{x}, s)) : 1 \leq i \leq n\}$ occurs as antecedent of one, and only one, axiom. Hence, there is a unique state update axiom whose antecedent, if instantiated with $A(\vec{\rho})$ and σ, is true in $\mathcal{M}$. We distinguish three cases:

1. If $\mathcal{E}$ contains some $\varepsilon^+_{A,F}(\vec{x}, s) \supset F(\vec{y}, \mathit{Do}(A(\vec{x}), s))$ such that $\mathcal{M} \models \varepsilon^+_{A,F}(\vec{\rho}, \sigma)$ and $\{\vec{x} \setminus \vec{\rho}\}$ binds $\vec{y}$ to $\vec{\tau}$, and if $\mathcal{M} \models \neg \mathit{Holds}(F(\vec{\tau}), \sigma)$, then $F(\vec{y}) \in \{F_i(\vec{y}_i)\}_{i \in \mathcal{I}_+}$ (wrt. the state update axiom which applies; c.f. (21)). Moreover, consistency of $\mathcal{E}$ ensures that $F(\vec{y}) \notin \{F_i(\vec{y}_i)\}_{i \in \mathcal{I}_-}$. Hence, $F(\vec{\tau})$ occurs in $\vartheta^{\mathcal{I}_+}$ but not in $\vartheta^{\mathcal{I}_-}$ and so is contained in $\mathit{State}(\mathit{Do}(A(\vec{\rho}), \sigma))$ according to Observation 3.

2. If $\mathcal{E}$ contains some $\varepsilon^-_{A,F}(\vec{x}, s) \supset \neg F(\vec{y}, \mathit{Do}(A(\vec{x}), s))$ such that $\mathcal{M} \models \varepsilon^-_{A,F}(\vec{\rho}, \sigma)$ and $\{\vec{x} \setminus \vec{\rho}\}$ binds $\vec{y}$ to $\vec{\tau}$, and if $\mathcal{M} \models \mathit{Holds}(F(\vec{\tau}), \sigma)$, then $F(\vec{y}) \in \{F_i(\vec{y}_i)\}_{i \in \mathcal{I}_-}$. Moreover, consistency of $\mathcal{E}$ ensures that $F(\vec{y}) \notin \{F_i(\vec{y}_i)\}_{i \in \mathcal{I}_+}$. Hence, $F(\vec{\tau})$ occurs in $\vartheta^{\mathcal{I}_-}$ but not in $\vartheta^{\mathcal{I}_+}$ and so is not contained in $\mathit{State}(\mathit{Do}(A(\vec{\rho}), \sigma))$ since it does not occur twice in $\mathit{State}(\sigma)$ according to foundational axiom (16) and Observation 3.

3. If no effect axiom of $\mathcal{E}$ at all for $A(\vec{\rho})$ and $F(\vec{\tau})$ applies in situation σ, or if a positive one applies with $F(\vec{\tau})$ being contained in $\mathit{State}(\sigma)$ already, or a negative one applies with $F(\vec{\tau})$ not present in $\mathit{State}(\sigma)$ already, then $F(\vec{\tau})$ does not occur in $\{F_i(\vec{y}_i)\}_{i \in \mathcal{I}_+}\{\vec{x} \setminus \vec{\rho}\}$ nor in $\{F_i(\vec{y}_i)\}_{i \in \mathcal{I}_-}\{\vec{x} \setminus \vec{\rho}\}$. Hence, $F(\vec{\tau})$ does not occur in $\vartheta^{\mathcal{I}_+}$ nor in $\vartheta^{\mathcal{I}_-}$ and so is contained in $\mathit{State}(\mathit{Do}(A(\vec{\rho}), \sigma))$ iff so it is in $\mathit{State}(\sigma)$ according to Observation 3.

■