On the Role of Possibility in Action Execution and Knowledge in the Situation Calculus

Vahid Vaezian(✉) and James P. Delgrande

School of Computing Science, Simon Fraser University,
Burnaby, BC V5A 1S6, Canada
{vvaezian,jim}@cs.sfu.ca

Abstract. In the Situation Calculus the term $do(a, s)$ denotes the successor situation to s, resulting from *performing* (i.e. executing) the action a. In other words, it is assumed that actions always succeed. If action a is not possible in situation s, then the action still succeeds but the resulting situation is not physically realizable. We will argue that consequences of this definition of $do(a, s)$ puts some limitations on applicability of the Situation Calculus. In this paper, we view $do(a, s)$ slightly differently which results in a more general form for successor state axioms. The new framework not only has all the benefits of the current version of the Situation Calculus but also offers several advantages. We suggest that it is more intuitive than the traditional account. As well, it leads to a more general solution to the projection problem. Last, it leads to a more general formalization of knowledge in the Situation Calculus.

1 Introduction

The Situation Calculus ([2,4]) is a formalism designed for reasoning about action. In the axiomatization, $do(a, s)$ denotes the result of executing action a in situation s, even if executing a is not possible in s. In other words, it is assumed that actions always succeed. If action a is not possible in situation s, then a still succeeds but the resulting situation is not physically realizable. Let's call this the *success assumption.* To take care of the case where a is not possible, the tree of situations is "pruned" by a separate executability condition, which limits things to only those situations which the actions "actually" succeed.

Although a great deal has been achieved with the current version of the Situation Calculus (from now on we call it the traditional framework), we argue that the success assumption leads to some limitations. In this paper, we define $do(a, s)$ differently by relaxing the success assumption and explore the impacts of this change in the theory of the Situation Calculus. In the new definition actions have no effect if their preconditions are not met. This leads to a more general form of the successor state axioms which follows from reconsidering the frame problem when unexecutable actions (i.e. those actions whose preconditions are not met) are involved.

In Sect. 2 we review the main concepts of the Situation Calculus. In Sect. 3 we make explicit the success assumption and discuss its consequences. In Sect. 4 we discuss the new definition of $do(a, s)$, present a more general form for successor state axioms and discuss the benefits of the proposed framework. In the last section we conclude and present directions for future work.

2 Background

The language of the Situation Calculus is a many-sorted first order language. There are three sorts: *action* for actions, *situation* for situations, and *object* for everything else. A distinguished constant S_0 represents the initial situation, and a distinguished function symbol *do* represents the execution of an action. A situation is a finite sequence of actions starting from the initial situation. A binary predicate symbol $\sqsubset$ defines an ordering relation on situations. A *fluent* is a fact whose truth value may vary from situation to situation; formally a fluent is a predicate that takes a situation as the final argument. An action then takes a situation to another situation in which the action has been executed. A situation calculus action theory includes an *action precondition* axiom for each action symbol, a *successor state axiom* for each fluent symbol, as well as *unique name axioms* and the *foundational axioms* of the Situation Calculus. Action precondition axioms are represented by the binary predicate *Poss*. Successor state axiom for a (relational) fluent F has the form

$$F(\boldsymbol{x}, do(a, s)) \equiv \gamma_F^+(\boldsymbol{x}, a, s) \vee [(F(\boldsymbol{x}, s) \wedge \neg\gamma_F^-(\boldsymbol{x}, a, s))] \qquad (1)$$

where $\gamma_F^+(\boldsymbol{x}, a, s)$ and $\gamma_F^-(\boldsymbol{x}, a, s)$ represent the conditions under which action a affects the value of fluent F positively or negatively.

3 The Traditional Definition of $\boldsymbol{do(a, s)}$

In the Situation Calculus, the term $do(a, s)$ denotes “the successor situation to s, resulting from *performing* the action a” [4]. In this definition, actions always succeed. In other words, in the situation denoted by $do(a, s)$ all the (conditional) effects of action a hold, even if it is not possible to perform action a in s. If performing a is not possible in situation s then $do(a, s)$ and subsequent situations are what Reiter calls “ghost” situations. In these cases the actions still succeed, in that the action effects hold in the resulting situation, but the resulting situation is not physically realizable. The focus is then put on the *executable* situations (i.e. those action histories in which it is actually possible to perform the actions one after the other). For example in planning in the Situation Calculus, where we have a goal statement G and want to find a plan that satisfies G, we prove $Axioms \models (\exists s).executable(s) \wedge G(s)$ which only deals with executable situations. As another example, in Golog, *primitive* and *test* actions are defined as

Primitive actions: $Do(a, s, s') \overset{def}{=} Poss(a[s], s) \wedge s' = do(a[s], s)$.

Test actions: $Do(\phi?, s, s') \overset{def}{=} \phi[s] \wedge s' = s$.

Note that executability is included in the definition of primitive actions, and test actions are always possible. More complex actions then are defined on top of these two kinds of actions, and they inherit executability as well.

This definition of $do(a, s)$, through Reiter's solution to frame problem [4], results in the successor state axiom (1) which in it truth or falsity of a fluent after an action is independent of whether the action is executable or not. For example if we have $holding(x, do(a, s)) \equiv a = pickup(x) \lor holding(x, s)$ then $holding(x, do(pickup(x), s))$ is always true for all x and s no matter executing the action $pickup(x)$ is possible in s or not.

In the sequel we discuss some consequences of the success assumption.

1. Projection Problem. Consider the example where an agent intends to execute a pickup action of some object followed by a move action to the next room, but the pickup is not possible, say as a result of the object being glued to the floor. In the traditional framework, following the pickup-and-move sequence, in the resulting situation the agent is holding the object and is in the next room. This is clearly impossible and the (separate) executability condition rules out such a (ghost) situation. As a result we cannot formalize these scenarios in the traditional framework. Clearly a more desirable outcome of this pickup-and-move sequence is that the agent is in the next room and the object's location is unchanged. This will be the result in our proposed framework.

2. Representing Knowledge. Formalizing knowledge is an important aspect of reasoning about change; there have been several variants of formalization of knowledge in the Situation Calculus ([1,4,5]). The most recent of these has been used as the standard representation in the literature; but as we will see all these variants have some issues and can be used only in a restricted way. These approaches differ on how they formalize the accessibility relation. We now summarize how knowledge is formalized in Situation Calculus based on [5].

The accessibility relation is represented by a relational fluent K. $K(s', s)$ denotes "situation s' is accessible from situation s". Knowledge is then defined naturally using the K fluent: ($\mathbf{Knows}(\phi, s) \overset{def}{=} (\forall s').K(s', s) \supset \phi[s']$) where ϕ is a *situation suppressed* expression and $\phi[s]$ denotes the formula obtained from ϕ by restoring situation variable s into all fluent names mentioned in ϕ.

The successor state axiom for the fluent K is

$$\begin{aligned} K(s'', do(a, s)) \equiv (\exists s').s'' = do(a, s') \land K(s', s) \\ \land Poss(a, s') \land \textsc{sr}(a, s) = \textsc{sr}(a, s') \end{aligned} \tag{2}$$

where SR is *sensing result function*, and formalizes the result of a sense action.

However there is a problem in formalizing knowledge in the traditional framework, as illustrated in the following example in the blocks world domain.

Example 1. Consider a robot which can pickup objects. There are two blocks A and B. Block A is known to be on the table, but the agent does not know whether B is on A or on the table. Nonetheless in the Scherl-Levesque approach [5], after a $pickup(A)$ action the agent believes that it is holding A, even though picking up A is not possible in one of the initial situations.

Formally, we have two initial situations S_0 and S_1. There are three fluents:

$holding(x, s)$: The robot is holding object x, in situation s.
$clear(x, s)$: There is no block on top of x, in situation s.
$on(x, y, s)$: x is on (touching) y, in situation s.

We have only one action ($pick\ up(x)$). Its action precondition axiom is:

$$Poss(pickup(x), s) \equiv clear(x, s) \wedge (\forall y)\neg holding(y, s)$$

Successor state axioms:

$$holding(x, do(a, s)) \equiv a = pickup(x) \vee holding(x, s)$$
$$clear(x, do(a, s)) \equiv [(\exists y).on(y, x, s) \wedge a = pickup(y)] \vee clear(x, s)$$
$$on(x, y, do(a, s)) \equiv on(x, y, s) \wedge a \neq pickup(x)$$

S_0: $(\forall x)\neg holding(x, S_0)$, $on(B, A, S_0)$, $on(A, Table, S_0)$, $clear(B, S_0)$.
S_1: $(\forall x)\neg holding(x, S_1)$, $on(A, Table, S_1)$, $on(B, Table, S_1)$, $clear(A, S_1)$, $clear(B, S_1)$.

The initial accessibility relations: $K(S_0, S_0), K(S_1, S_0), K(S_1, S_1), K(S_0, S_1)$.

Note that $pickup(A)$ is a physical action, so the accessibility relations are preserved. Also note that it is possible to pick up A in S_1 while it is not possible in S_0. Therefore we expect that in a "correct" formalization the accessibility relations and possible worlds after $pickup(A)$ be as shown in Fig. 1.

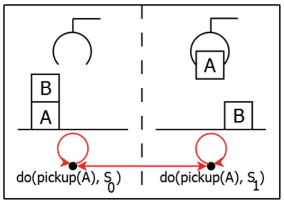

Fig. 1. Desired accessibility relations and possible worlds after action $pickup(A)$

But in the traditional framework, after a $pickup(A)$ action the formulas $holding(A, do(pickup(A), S_0))$ and $holding(A, do(pickup(A), S_1))$ hold; so no matter how the accessibility relation gets updated (i.e. no matter which formulation of K fluent we opt) the agent will believe that it is holding A after $pickup(A)$.

In addition to this general problem the Scherl-Levesque approach has also a problem with updating the accessibility relation. Note that using (2) the only accessibility relations we get are $K(do(pickup(A), S_1), do(pickup(A), S_1))$ and $K(do(pickup(A), S_1), do(pickup(A), S_0))$ (because of $\neg Poss(pickup(A), S_0)$).

Among other proposed successor state axioms for the K fluent, [1] has similar problems. The one suggested in [4] which using the standard terminology is

$$K(s'', do(a,s)) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \wedge \textsc{sr}(a,s) = \textsc{sr}(a,s') \tag{3}$$

returns the desired accessibility relations after *pickup*(A), but the general problem discussed above remains (the agent believes it is holding A after *pickup*(A)).

The main reason behind the general problem discussed above is that in the traditional framework ghost situations cannot represent possible worlds. Note that in the traditional framework, when action a is not possible in situation s, $do(a,s)$ is a ghost situation and can only represent an imaginary world where action a was successfully executed (although it was not possible) in the world represent by situation s. In our proposed framework we will not have ghost situations and the problem with formalizing knowledge will be fixed.

Note that assuming there is no unexecutable action possible in the model is often too restrictive. The reason is that when dealing with knowledge we have multiple initial situations. Assuming that all actions are executable in every situation starting from any of the initial situations is a very strong assumption.

4 An Alternative Definition of *do*(a,s)

Let $do(a,s)$ denote "the successor situation to s, resulting from *attempting* the action a". An *attempt* to do an action is different from *performing* (i.e. executing) an action in the sense that it is not assumed that it succeeds. If an action is executable then it has its expected effects, otherwise nothing happens (i.e. a null action is executed). This is reasonable because for example if *pickup*(x) is unexecutable (say, because x is glued to the floor), after (attempting) this action, it is reasonable to assume that nothing happens.

In [3], Reiter presented a solution to the frame problem building on the previous works of Davis, Haas and Schubert. He then developed a form for successor state axioms using this solution. If we follow a similar pattern in the new framework we will obtain the following form for successor state axioms[1]:

$$\begin{aligned} F(\boldsymbol{x}, do(a,s)) \equiv\ & (Poss(a,s) \wedge \gamma_F^+(\boldsymbol{x},a,s)) \\ & \vee [F(\boldsymbol{x},s) \wedge (\neg\gamma_F^-(\boldsymbol{x},a,s) \vee \neg Poss(a,s))] \end{aligned} \tag{4}$$

It is important to note that this successor state axiom gives Reiter's successor state axiom (1) as a special case when we have $Poss(a,s) \equiv \top$.

We now describe some advantages of the new framework and show that the new framework does not suffer from the aforementioned limitations.

Projection Problem. In the new framework we can solve the more general form of the problem where the sequence of actions does not have to be executable. Reconsidering the example discussed before, using the new form of successor

[1] The discussion and proof will be given in the full paper.

state axiom, after the sequence of pickup-and-move actions, the agent is in room 2 not holding any object, and the object is still in room 1, as desired.[2]

Representing Knowledge. The ability of our framework to formalize the result of unexecutable actions (by means of the new form of successor state axiom) enables it to regard situations as "states" of the world. In other words, when action a is possible in situation s, the situation $do(a, s)$ represents the world resulting from executing action a in the world represented by situation s, and when action a is not possible, the situation $do(a, s)$ represents the world resulting from the failed attempt to execute action a (which as we assumed has the same properties as s). This advantage will be useful for formalizing knowledge.

Reconsider Example 1. The new successor state axioms are

$$\begin{aligned} holding(x, do(a, s)) &\equiv [a = pickup(x) \wedge Poss(a, s)] \vee holding(x, s) \\ clear(x, do(a, s)) &\equiv [(\exists y).on(y, x, s) \wedge a = pickup(y) \wedge Poss(a, s)] \vee clear(x, s) \\ on(x, y, do(a, s)) &\equiv on(x, y, s) \wedge (a \neq pickup(x) \vee \neg Poss(a, s)) \end{aligned}$$

First note that the new axiomatization entails $\neg holding(A, do(pickup(A), S_0)$. Determining the value of other fluents, we see that in our framework the possible worlds after action $pickup(A)$ are as shown in Fig. 1 (the desired results).

For accessibility relation we use (3). Note that the difference is that for fluents other than K we are using the new form of successor state axiom. In our framework possibility of actions has been considered in successor state axioms of fluents (excluding the K fluent). These fluents characterize the possible worlds. Therefore there is no need to mention $Poss$ in the fluent K which characterizes the accessibility relation between possible worlds. Using this formulation of K and the new form of successor state axiom for other fluents, we will get the expected results after actions $pickup(A)$.[3]

5 Conclusion

We have provided a more nuanced and expressive version of the Situation Calculus by presenting a more general form for successor state axioms which stems from a different definition of $do(a, s)$.

We described some advantages of the new framework. In the traditional framework we can solve projection problem but only for executable situations. We can regard situations as possible worlds but only when the situation is executable. We showed that the current formalization of knowledge works only for executable situations. In our framework we don't have these limitations and it allows us to utilize the power of the Situation Calculus in a broader area. Studying other impacts of the new framework is subject of the future research.

[2] A formal account of the problem will be given in the full paper.

[3] The complete description will be given in the full paper.

References

1. Levesque, H., Pirri, F., Reiter, R.: Foundations for the situation calculus. Electron. Trans. Artif. Intell. **2**(3–4), 159–178 (1998)
2. McCarthy, J.: Situations, actions and causal laws. Technical report, Stanford University (1963)
3. Reiter, R.: The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In: Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy (1991)
4. Reiter, R.: Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems. MIT Press, Cambridge (2001)
5. Scherl, R.B., Levesque, H.J.: Knowledge, action, and the frame problem. Artif. Intell. **144**(1), 1–39 (2003)