Noname manuscript No.
(will be inserted by the editor)

The Situation Calculus: A Case for Modal Logic

Gerhard Lakemeyer

Received: date / Accepted: date

Abstract The situation calculus is one of the most established formalisms for reasoning about action and change. In this paper we will review the basics of Reiter's version of the situation calculus, show how knowledge and time have been addressed in this framework, and point to some of the weaknesses of the situation calculus with respect to time. We then present a modal version of the situation calculus where these problems can be overcome with relative ease and without sacrificing the advantages of the original.

Keywords Situation Calculus · Epistemic Logic · Dynamic Logic

1 Introduction

The situation calculus is one of the most established formalisms for reasoning about action and change. It is almost as old as the field of Artificial Intelligence itself and was originally proposed by McCarthy [37] as a dialect of first-order logic. But it had a major impact only after Reiter proposed his solution to the frame problem [43], which was the start of a whole research agenda culminating in the publication of his book [44]. An important milestone was the development of the action programming language Golog [34], which is based on the situation calculus and which led to applications in robotics [8,32] and the semantic web [38], among other things.

Situations share many properties with possible worlds, but in contrast to modal logic, these are reified in the situation calculus. Properties of situations are captured axiomatically, and there is no special semantics, that is, ordinary Tarskian models suffice. As pointed out in [28], this approach is fine if we are just interested in what a theory entails. A prominent example is the *projection* task, where a domain theory about a dynamic world is given and one asks questions about what will be true after a number of actions are executed. Projection is a key mechanism underlying planning

G. Lakemeyer
Dept. of Computer Science
RWTH Aachen University
Germany
E-mail: gerhard@cs.rwth-aachen.de

and languages like Golog. Questions other than entailment are not so easy to deal with in the situation calculus. One example is: if *Theory1* entails *Formula1*, is it also the case that *Theory2* entails *Formula2*? We will see an example of this later (Theorem 3 on page 12). This was the motivation in [28] to come up with a new modal version of the situation calculus, where situations are banned from the language and the new logic called $\mathcal{ES}$ is given a possible-world semantics. It is shown that $\mathcal{ES}$ retains all the advantages of Reiter's situation calculus but where questions like the above become much easier to answer.

The purpose of this paper is to make this case for a modal situation calculus again and support it with more evidence. For that we will start by reviewing Reiter's original situation calculus with a focus on its most successful application, namely solving the projection task. We will then highlight how features like knowledge and time, which have long been studied in the modal logic community, are treated in the situation calculus. In the context of time we will see more evidence that the situation calculus is problematic when it comes to expressing temporal notions such as branching time in a natural way. Then we will introduce the modal situation calculus $\mathcal{ES}$ and show that these problems can be overcome with relative ease there.

2 The Situation Calculus

We begin with the basic situation calculus as defined by Reiter and add epistemic and temporal features as we go along. The situation calculus is a sorted second-order[1] language with equality. The logical connectives are $\neg, \wedge$, and $\forall$. We will freely use $\vee, \exists, \supset$, and $\equiv$, which are understood as the usual abbreviations. There are predicate and function symbols of every arity and three sorts: situations, actions, and objects. Situations are understood as histories of actions starting in an initial situation denoted by the constant S_0. A special binary function $do(a, s)$ is used to refer to the situation which results from performing action a in s. For example,

$$do(\mathit{putonfloor}(B), do(\mathit{pickup}(B), S_0))$$

denotes the situation after the agent picks up object B in S_0 and then puts it on the floor. In other words, we can get from one situation to another by performing an action and *do* records the history of actions that lead to the situation referred to by the *do*-term. To enhance readability, we will sometimes write $do([a_1, \ldots, a_n], s)$ instead of $do(a_n, do(a_{n-1}, \ldots, do(a_1, S_0)\ldots)$. *Fluents* are used to talk about what is true at a given situation. These are simply predicates and functions whose last argument is a situation term. For example,

$$\mathit{Holding}(B, do(\mathit{pickup}(B), S_0))$$

may be read as "the agent is holding object B after picking it up in S_0." There are two special predicates $\mathit{Poss}(a, s)$, indicating that action a is possible in situation s, and $s \sqsubset s'$, which says that s precedes s', that is, s' can be reached from s by a sequence of actions. We will also write $s \sqsubseteq s'$ as shorthand for $s \sqsubset s' \vee s = s'$. For the purposes of this paper, we assume that all predicates of the language are fluents and that all functions are rigid, that is, they have no situation arguments.

[1] As we will see below, only a tiny bit of second-order logic is actually used, namely to define the set of all situations.

The intended use of the situation calculus is that a domain expert would axiomatize the relevant aspects of an application, including its dynamics, and then use some automated reasoning mechanism to arrive at interesting conclusions about the domain, in particular, about what holds after a number of actions have been performed. Among other things, axioms are needed which describe the effects of actions, their preconditions, and what is true initially. Here is an example of an effect axiom, which says that, provided an object is within reach, the agent will hold it in its hand after picking it up.

$$\mathit{WithinReach}(x, s) \land a = \mathit{pickup}(x) \supset \mathit{Holding}(x, \mathit{do}(a, s)).$$

Throughout we often leave out universal quantifiers. The rule is that all free variables are implicitly universally quantified, unless stated otherwise. We use s for situation variables, a for actions, and others like x and y for objects.

What we have seen so far is no more than a variant of McCarthy's original proposal from fifty years ago. In the early days, perhaps the strongest argument in favour of using the situation calculus for problem solving came out of the work by Cordell Green [22]. The only problem was that it did not quite work. This was because getting the right inferences turned out to be very difficult computationally, if not impossible. Among the reasons for this early failure perhaps the following stand out: automated reasoning methods like resolution were still in their infancy and the frame problem was unsolved. The latter refers to the fact that any logical theory of action not only needs to have axioms which describe the effects of actions but also their non-effects. For example, after the robot has picked up a blue book and then moves to another room, we would like to conclude that the book is not only still in the robot's possession but that it has remained blue, as moving does not normally change the colour of objects. What makes this a problem is that there usually are far more non-effects than effects and having to write them down explicitly is not only cumbersome but makes the reasoning task even more daunting. From this dilemma two important streams of research emerged. For one, people interested in planning gave up on the idea of using a very expressive representation language and turned to STRIPS [21].[2] For another, the frame problem was a major impetus for nonmonotonic reasoning. The idea was quite intuitive and appealing in that non-effects were seen as default conclusions, hence obviating the necessity to explicitly represent them (see [47] for a detailed discussion). Alas, nonmonotonic reasoning turned out to be even more complex than classical logical reasoning. In any case, after the early failures most people believed (and some still do) that the situation calculus is not useful as the basis for building agents which reason about action.

2.1 Basic Action Theories

This was essentially the situation until the early nineties, when Ray Reiter proposed a monotonic solution to the frame problem by introducing what he calls *successor state axioms*, one for each fluent, which provide necessary and sufficient conditions for the values of fluents after any action. These together with precondition axioms and axioms describing what is true initially (in S_0) make up what Reiter calls *basic action theories.*

[2] In fact, an early version of the famous SRI robot Shakey used the situation calculus for planning, but it was later dropped in favour of STRIPS.

We begin with a very brief review of the foundational axioms Σ for the situation calculus from [44].

1. $do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2$;
2. $\forall Q.[Q(S_0) \wedge \forall s, a.[Q(s) \supset Q(do(a, s))] \supset \forall s.Q(s)$;
3. $s \sqsubset do(a, s') \equiv s \sqsubseteq s'$;
4. $\neg s \sqsubset S_0$.

(1) is a unique names axiom for situations; the second-order axiom; (2) is similar to the induction axiom for the natural numbers and defines the set of all situations to be exactly those reachable from S_0 by a sequence of actions; (3) defines $\sqsubset$ as reachability (by a sequence of actions) between situations; (4) says that no situation precedes S_0.

Essentially, these axioms make sure that the space of situations forms a tree rooted in S_0, where the nodes of the tree are the situations and the edges represent actions connecting situations with their successors. Note that, despite the induction axiom, there may be uncountably many situations in case the set of actions is uncountable.

Precondition axioms have the form

$$Poss(A(\mathbf{x}, s)) \equiv \Pi_A(\mathbf{x}, s),$$

which specifies under which conditions action A with arguments $\mathbf{x}$is executable in situation s. Π_A is required to be *uniform* in the situation variable s. This means, roughly, that s is the only situation term occurring in Π_A, and only as the last argument of a fluent, and s is free everywhere.

Successor state axiom define, for each fluent F, what the truth value of F will be after an action has occurred. It has the form

$$F(\mathbf{x}, do(a, s)) \equiv \Phi_F(\mathbf{x}, a, s),$$

where Φ_F is again uniform in s.

A basic action is then defined as

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

where Σ is the set of foundational axioms, $\mathcal{D}_{ap}$ the set of action preconditions, one for each action type, $\mathcal{D}_{ss}$ the set of successor state axioms, one for each fluent, $\mathcal{D}_{S_0}$ consists of sentences uniform in S_0 describing the initial situation, and $\mathcal{D}_{una}$ consists of unique name axioms for actions.

As an illustration, consider the following blocks-world example of a robot with three actions *pickup*, *putontable*, and *putonfloor* with the obvious connotation. The fluents are $Holding(x, s)$, which is true when the robot is holding an object in its gripper, $OnTable(x, s)$ and $OnFloor(x, s)$, which are true when object x is on the table or on the floor, respectively. As action preconditions ($\mathcal{D}_{ap}$) we have the following sentences:

$Poss(pickup(x), s) \equiv \forall z \neg Holding(z, s)$
$Poss(putontable(x), s) \equiv Holding(x, s)$
$Poss(putonfloor(x), s) \equiv Holding(x, s)$

In other words, the robot can pick up an object if it is not holding anything, and it can put away an object if it is currently holding it. The successor state axioms ($\mathcal{D}_{ss}$) are these sentences:

$Holding(x, do(a, s)) \equiv [a = pickup(x) \vee$
$\quad Holding(x, s) \wedge a \neq putontable(x) \wedge a \neq putonfloor(x)]$

$$\mathit{OnTable}(x, \mathit{do}(a, s)) \equiv [a = \mathit{putontable}(x) \lor \mathit{OnTable}(x, s) \land a \neq \mathit{pickup}(x)]$$
$$\mathit{OnFloor}(x, \mathit{do}(a, s)) \equiv [a = \mathit{putonfloor}(x) \lor \mathit{OnFloor}(x, s) \land a \neq \mathit{pickup}(x)]$$

Let us consider the axiom for *Holding*. It says that after performing an action the robot is holding object x if the action is *pickup*(x), or if it was holding it before and did not put the object away. Finally, we need to describe the initial state ($\mathcal{D}_{S_0}$):

$$\mathit{OnTable}(x, S_0) \equiv (x = B) \lor (x = C)\ \forall x \neg \mathit{Holding}(x, S_0)$$

In other words, we assume that there are exactly two objects B and C on the table initially and the robot is not holding anything.

Given a basic action theory like the above one can then answer questions about what holds after some actions have occurred. This is also called the *projection problem* and lies at the heart of planning but also of action programming languages like Golog [34]. For example, we obtain

$$\mathcal{D} \models \mathit{OnFloor}(B, \mathit{do}([\mathit{pickup}(B), \mathit{putonfloor}(B)], S_0),$$

that is, object B is indeed on the floor after the robot picked it up and put it there.

In principle, projections could be computed from $\mathcal{D}$ using automated reasoning methods, but this is unlikely to be feasible in general. Also, keep in mind that $\mathcal{D}$ is second order, if only to a small degree. Perhaps the most important contribution of Reiter's work on the situation calculus was that he showed that there is a much better and more efficient way to compute projection using *regression.* Without going into too much detail, the idea of regression is that any fluent $F(\mathbf{t}, \mathit{do}(A, \sigma))$ is replaced by $\Phi_F(\mathbf{t}, A, \sigma)$, the R.H.S. of the corresponding successor state axiom with appropriate variable substitutions. Similarly, occurrences of *Poss* are also replaced by their definitions in $\mathcal{D}_{ap}$. The result is a formula where the level of nesting of *do*-terms is reduced by one. If σ is a ground situation term this can be iterated until the resulting formula only mentions S_0.

Theorem 1 (Reiter) *Let ϕ be a formula whose situation terms are all ground. Then there is a formula ϕ' uniform in S_0 (obtained by regression) such that*

$$\mathcal{D} \models \phi \quad \textit{iff} \quad \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \phi'.$$

As an illustration, consider our example basic action theory and

$$\phi = \mathit{Holding}(B, \mathit{do}(\mathit{pickup}(B), S_0)) \land \mathit{Poss}(\mathit{pickup}(B), S_0).$$

Then the regression of ϕ (after simplification) is $\phi' = \forall x. \neg \mathit{Holding}(x, S_0)$, which obviously follows from $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$.

Note that the R.H.S. of the theorem does not mention the foundational axioms, thus making the projection problem amenable to existing first-order theorem-proving technology. With further restrictions such as $\mathcal{D}_{S_0}$ consisting only of literals and the closed-world assumption, which are routinely made in Prolog implementations, this often becomes efficiently computable.

Besides knowledge and time, which we will turn to in a moment, there have been a number of other extensions of the situation calculus, dealing with issues like concurrency [44] or probabilities [1]. Perhaps the most important use of the situation calculus has been within the framework of the Golog family of languages like [34, 19, 7]. Golog combines features from traditional imperative programming language with planning, and its semantics is completely specified within the situation calculus. In particular, the actions referred to in Golog programs obtain their meaning from a basic action theory as described above. Golog has been applied to the control of real robots [8, 17, 9]. See also [32] for a general story about applying the situation calculus to cognitive robotics.

2.2 Knowledge in the situation calculus

So far the situation calculus allows us to talk about what is true in the world and how it changes. We now add an agent's epistemic state to the picture as proposed by Scherl and Levesque [46] and building on earlier ideas by Moore [39]. The idea is to add a special binary fluent K to the language, where $K(s', s)$ should be read as "situation s' is (epistemically) accessible from s." Knowledge is then defined as truth in all accessible worlds, and we write

$$\mathit{Knows}(\phi(\mathit{now}), s) \overset{def}{=} \forall s'.K(s', s) \supset \phi[s'].$$

Here $\phi(\mathit{now})$ is a formula uniform in the special situation variable now.[3] For example, $\mathit{Knows}(\mathit{OnTable}(A, \mathit{now}), S_0)$ stands for $\forall s'.K(s', S_0) \supset \mathit{OnTable}(A, s')$.

For those familiar with possible-world semantics all this should look awfully familiar, and indeed, what we have done here is simply reify the accessibility relation of possible-world semantics within the situation calculus.[4] However, there is still one question that needs to be addressed: what kinds of situations should be accessible at any given moment? Or, put differently, given that we already have infinitely many situations at our disposal, can we choose arbitrarily among them and make them epistemically accessible? The answer, it seems, is no. To see why consider an agent's knowledge who has not performed any actions. Intuitively, whatever situations the agent considers possible initially, these should themselves be initial situations. After all, even if the agent has no knowledge about the world initially, it should know that it has not done anything yet. But so far, the only initial situation is S_0. It seems the only way out of this dilemma is to extend the space of situations and allow other initial situations (and their successors) besides S_0. The following two axioms take care of all this:

2′ $\forall Q.\forall s.[\mathit{Ini}(s) \supset Q(s)] \wedge \forall a, s.[Q(s) \supset Q(do(a, s))] \supset \forall s.Q(s)$,
where $\mathit{Ini}(s) \overset{def}{=} \forall a \forall s'.\, s \neq do(a, s')$;
5. $K(s', s) \supset [\mathit{Ini}(s) \equiv \mathit{Ini}(s')]$.

Axiom 2′ replaces Axiom 2 and is the new induction axiom, making sure that every situation is rooted in some initial situation. Axiom 5 formalizes what we just said, namely that initially only initial situations are accessible.

What is still left to do is a specification of how the K-fluent and hence the agent's knowledge evolves when actions are performed. To this end, Scherl and Levesque [46] came up with a successor state axiom for K, which in its simplest form looks like this:

$$K(s'', do(a, s)) \equiv \exists s'.s'' = do(a, s') \wedge K(s', s) \wedge \mathit{Poss}(a, s').$$

[3] The notion of *uniform in now* can be extended to formulas with nested occurrences of *Knows*, but we leave out the details here.

[4] As a minor quirk, note that the order of the arguments of K is reversed compared to the standard use.

In other words, the accessible situations accessible after doing an action are precisely the successors of those situations accessible before doing the action, provided the execution of the action is possible.[5] One of the nice properties of this axiom is that it makes sure that whatever general property the accessible situations have initially, these will be preserved by actions. In particular, this is true for reflexive, transitive, symmetric, or Euclidean relations. So if an agent is fully introspective initially (K is transitive and Euclidean), then this remains true after any sequence of actions.

3 Time

There are essentially two approaches to modelling time in the situation calculus, one adds time explicitly as a new kind of individuals, and another one makes use of the implicit temporal structure the tree of situations provides. While the former may be a bit unusual for readers from the temporal logic community, the latter will be rather familiar.

Treating time points as first-class citizens was done in [40,44,20]. In its most basic form, actions are fitted with an extra time argument in the last position ranging over the reals. For example, $\mathit{pickup}(x,t)$ should be understood as "the agent picks up object x at time t." To make the formalism easier to work with, two functions *time* and *start* are introduced with

$$\mathit{time}(A(\mathbf{x},t)) = t \quad \text{and} \quad \mathit{start}(\mathit{do}(a,s)) = \mathit{time}(a)$$

as new foundational axioms. This allows a simple definition of executability of an action that takes into account temporal ordering:

$$\mathit{executable}(a,s) \equiv \mathit{Poss}(a,s) \wedge \mathit{start}(s) \leq \mathit{time}(a),$$

that is, an action is executable in s if it is possible and its execution time does not precede the start time of s.

Given an explicit notion of time, it seems desirable to allow actions with durations. Although actions in the situation calculus really are discrete and instantaneous, Pinto showed that it is not that difficult to model durative actions using a little trick. Consider, for example, the action of going to location $\langle x,y\rangle$. Instead of a non-temporal primitive action $\mathit{goto}(x,y)$, Pinto suggested to use two actions $\mathit{startgoto}(x,y,t_0)$ and $\mathit{endgoto}(x,y,t_1)$ with the obvious understanding that the goto starts at time t_0 and ends at t_1. All that is needed in addition is another fluent $\mathit{going}(s)$ which is false initially, set true by *startgoto* and false again by *endgoto*. This way it is easy to prevent the start of another *goto*-action before the other has ended. With a little extra effort, the model of time can be adapted further to allow fluents to change continuously with the passage of time [20]. This is particularly useful in applications like robotics.

While the explicit use of time in the situation calculus has been around for more than 10 years, the implicit use which relies on the precedence ordering inherent in situations is fairly recent. It was introduced by Gabaldon [18] and later used by Bienvenue et al. [5]. Roughly, the idea is to introduce familiar temporal operators like *eventually* or *until,* which are used to specify temporal constraints or preferences on situations.

[5] Scherl and Levesque also considered sensing or knowledge-producing actions, which slightly complicates the successor state axiom for K, but we gloss over those details here.

They define what they call *basic desire formulas (BDFs)*, which are formed from situation-suppressed first-order formulas not mentioning *Poss*, $\sqsubset$, and equality expressions with situation arguments, the usual logical connectives, and temporal operators $\boldsymbol{F}$ (eventually), $\boldsymbol{X}$ (next), and $\boldsymbol{U}$ (until). There is also a new predicate $occ(a)$, which is true when a is the next action to occur.

Formally, given two situations s and s' such that s' precedes s ($s' \sqsubseteq s$), $\phi[s', s]$ is is inductively defined as follows:

- $F(\mathbf{t}, now)[s', s] \overset{def}{=} F(\mathbf{t}, s')$ for fluent F
- $t_1 = t_2[s', s] \overset{def}{=} t_1 = t_2$
- $(\neg\phi)[s', s] \overset{def}{=} \neg\phi[s', s]$
- $(\phi \wedge \psi)[s', s] \overset{def}{=} \phi[s', s] \wedge \psi[s', s]$
- $(\forall x.\phi)[s', s] \overset{def}{=} \forall x \phi[s', s]$
- $occ(a)[s', s] \overset{def}{=} do(a, s') \sqsubseteq s$
- $\boldsymbol{F}\phi[s', s] \overset{def}{=} \exists s_1 . s' \sqsubseteq s_1 \sqsubseteq s \wedge \phi[s_1, s]$[6]
- $\boldsymbol{X}\phi[s', s] \overset{def}{=} \exists a . do(a, s') \sqsubseteq s \wedge \phi[do(a, s'), s]$
- $\phi \boldsymbol{U} \psi[s', s] \overset{def}{=} \exists s_1 . s' \sqsubseteq s_1 \sqsubseteq s \wedge \psi[s_1, s] \wedge \forall s_2 . s' \sqsubseteq s_2 \sqsubset s_1 \supset \phi[s_2, s]$

As in temporal logic, other operators are definable in terms of the above. For example, $\boldsymbol{G}\phi$ (always ϕ) is $\neg\boldsymbol{F}\neg\phi$. To see what can be expressed with this language, consider the following example, adapted from [5], about the preferences of an agent making dinner plans:

$$\begin{array}{c}[\exists x, y . \boldsymbol{F}(occ(orderTakeout(x, y))) \vee \\ \boldsymbol{F}(occ(orderRestaurant(x, y)))] \quad \wedge \\ \boldsymbol{G}(\neg\exists x . occ(eat(x)) \wedge type(x) = chinese)).\end{array}$$

Roughly, the agent would like to either order takeout food or something at a restaurant, but in any case, she wants to avoid eating Chinese food.

One use of BDFs is to control the search for a plan (situation) satisfying a goal $G(s)$, that is, instead of proving $\mathcal{D} \models \exists s . G(s)$ one tries to establish $\mathcal{D} \models \exists s . \phi[S_0, s] \wedge G(s)$. As shown in [2,18], this can significantly improve the performance of forward-search planners.

While it seemed almost effortless to define LTL operators in the situation calculus, there is a catch. Notice that the definitions require two situations s and s' such that $s' \sqsubseteq s$. In temporal logic terms this means that we restrict ourselves to finite paths, and so we cannot express, for example, that a certain property repeats itself infinitely often. And what about branching time? After all, given the tree-like nature of situations, one would think that the logic lends itself most naturally to the definition of branching-time operators as, say, in CTL* [15]. Curiously, this does not seem to be the case. To be fair, some things can be said easily such as

$$\forall s . now \sqsubseteq s \supset \phi,$$

which can be read as "from now on ϕ always holds." But this refers to all situations on all paths starting at *now*, where a path is understood as one of the branches of the situation tree rooted in *now*. It is much more difficult to express properties that

[6] We write $s' \sqsubseteq s_1 \sqsubseteq s$ as shorthand for $s' \sqsubseteq s_1 \wedge s_1 \sqsubseteq s$.

hold along a particular path. For example, it is not immediately obvious how to say in the situation calculus that there is a path such that eventually ϕ becomes true on this path. In fact, it seems that we need to resort to second-order logic to formalize such path quantifiers.

While this can be done in principle, it seems somewhat dubious that we have to resort to higher order logic to express such simple concepts. In fact, one could argue that this is just another symptom of a deeper problem with the situation calculus: by giving access to situations in the language, it simply becomes too fine-grained and unwieldy. Notice also that in practice very little of this very expressive language is used. For example, while there is no restriction in principle on how to use the K-fluent, the only real use is in terms of the knowledge operator of modal logic.

This suggests that perhaps we can get by with a less expressive modal language, which is sufficient to capture Reiter-style basic action theories but at the same time becomes much more manageable and where temporal concepts can be embedded in a style similar to existing temporal logic. Indeed, such a logic was recently introduced [28–30], and we will present it in the next section. Moreover, we will show how branching time operators can be added and discuss related approaches from the modal logic community.

4 The Modal Situation Calculus $\mathcal{ES}$

The language is a first-order modal dialect with equality and sorts of type object and action. Unlike other languages, the language includes (countably many) standard names for both objects and actions. These can be thought of as special extra constants that satisfy the unique name assumption and an infinitary version of domain closure. This idea was adapted from [31]. Its main advantage is that this allows quantification to be understood substitutionally. The language also contains fluent predicates and rigid functions.[7]

4.1 The Language

Definition 1 The symbols of $\mathcal{ES}$ are taken from the following vocabulary:

- first-order variables: $x_1, x_2, \ldots, y_1, y_2, \ldots, a_1, a_2, \ldots$;
- standard names: $n_1, n_2, \ldots$ for objects and actions;
- rigid function symbols of arity k: $g_1^k, g_2^k, \ldots$;
- fluent predicate symbols of arity k: $F_1^k, F_2^k, \ldots$;
- connectives and other symbols: $=$, $\wedge$, $\neg$, $\forall$, *Know*, $\Box$, round and square parentheses, period, comma.

We assume that the fluent predicates include the special predicate *Poss*.

Definition 2 The *terms* of the language are of sort *action* or *object*, and form the least set of expressions such that

1. Every standard name and first-order variable is a term of the corresponding sort;

[7] In [29,30], a full version of $\mathcal{ES}$ is considered, with rigid predicates, fluent functions, as well as fluent and rigid second-order variables.

2. If $t_1, \ldots, t_k$ are terms and h is a k-ary function symbol then $h(t_1, \ldots, t_k)$ is a term of the same sort as h.

By a *primitive term* we mean one of the form $h(n_1, \ldots, n_k)$ where h is a function symbol and all of the n_i are standard names.

Definition 3 The *well-formed formulas* of the language form the least set such that

1. If $t_1, \ldots, t_k$ are terms, and F is a k-ary predicate symbol then $F(t_1, \ldots, t_k)$ is an (atomic) formula;
2. If t_1 and t_2 are terms, then $(t_1 = t_2)$ is a formula;
3. If t is an action term and α is a formula, then $[t]\alpha$ is a formula;
4. If α and β are formulas, and v is a variable, then the following are also formulas: $(\alpha \wedge \beta)$, $\neg\alpha$, $\forall v.\, \alpha$, $\Box\alpha$, $\mathit{Know}(\alpha)$.

We read $[t]\alpha$ as "α holds after action t", and $\Box\alpha$ as "α holds after any sequence of actions." As usual, we treat $(\alpha \vee \beta)$, $(\alpha \supset \beta)$, $(\alpha \equiv \beta)$, and $\exists v.\, \alpha$, as abbreviations. To ease notation, we leave the type of variables implicit. We reserve the symbol a to denote a variable of type action.

We call a formula without free variables a *sentence.* By a *primitive sentence* we mean a formula of the form $F(n_1, \ldots, n_k)$ where F is a predicate symbol and all of the n_i are standard names. A formula with no *Know* operators is called *objective.* A formula with no fluent, $\Box$, or $[t]$ operators outside the scope of a *Know* is called *subjective.* A formula with no *Know*, $\Box$, or $[t]$ is called a *fluent* formula.

4.2 The semantics

The main purpose of the semantics we are about to present is to give meaning to fluents, which may vary as the result of actions and whose values may be unknown. Intuitively, to determine whether or not a sentence α is true after a sequence of actions z has been performed, we need to specify two things: a world w and an epistemic state e. We write $e, w, z \models \alpha$. A world determines truth values for the primitive sentences and co-referring standard names for the primitive terms after any sequence of actions. An epistemic state is defined by a set of worlds, as in possible-world semantics. More precisely, let $\mathcal{N}$ denote the set of all standard names and $\mathcal{Z}$ the set of all finite sequences of standard action names, including $\langle\,\rangle$, the empty sequence. Then

– a world $w \in W$ is any function from the primitive sentences and $\mathcal{Z}$ to $\{0, 1\}$, and from the primitive terms to $\mathcal{N}$ (preserving sorts).[8]
– an epistemic state $e \subseteq W$ is any set of worlds.

We extend the idea of co-referring standard names to arbitrary ground terms as follows. Given a term t without variables and a world w we define $|t|_w$ (read: the co-referring standard name for t given w) by:

1. If $t \in \mathcal{N}$, then $|t|_w = t$;
2. $|g(t_1, \ldots, t_k)|_w = w[g(n_1, \ldots, n_k), z]$, where $n_i = |t_i|_w$.

[8] Since we only deal with rigid terms, there is no need to include action sequences for those.

To interpret formulas with free variables, we proceed as follows. First-order variables are handled substitutionally using the standard names.

Finally, to be compatible with the original situation calculus where the values of rigid terms are always known, we introduce the notion of compatibility $\simeq$ between worlds:

1. $w' \simeq w$ iff w' and w agree on the value of every primitive rigid term;[9]

Putting all these together, here is the semantic definition of truth. Given a sentence α, $e \subseteq W$ and $w \in W$, we define $e, w \models \alpha$ (read: α is true) as $e, w, \langle\rangle \models \alpha$, where for any $z \in \mathcal{Z}$ we have:

1. $e, w, z \models F(t_1, \ldots, t_k)$ iff $w[F(n_1, \ldots, n_k), z] = 1$, where $n_i = |t_i|_w$;
2. $e, w, z \models (t_1 = t_2)$ iff n_1 and n_2 are identical, where $n_i = |t_i|_w$;
3. $e, w, z \models [t]\alpha$ iff $e, w, z \cdot n \models \alpha$, where $n = |t|_w$;
4. $e, w, z \models (\alpha \wedge \beta)$ iff $e, w, z \models \alpha$ and $e, w, z \models \beta$;
5. $e, w, z \models \neg\alpha$ iff $e, w, z \not\models \alpha$;
6. $e, w, z \models \forall x.\, \alpha$ iff $e, w, z \models \alpha^x_n$, for every std. name n of the right sort;
7. $e, w, z \models \Box\alpha$ iff $e, w, z \cdot z' \models \alpha$, for every $z' \in \mathcal{Z}$;
8. $e, w, z \models \mathit{Know}(\alpha)$ iff $e, w', z \models \alpha$, for every $w' \in e$ such that $w' \simeq w$.

When α is *objective* (has no *Know* operators), we can leave out the e and write $w \models \alpha$. Similarly, when α is *subjective*, we can leave out the w and write $e \models \alpha$. When Σ is a set of sentences and α is a sentence, we write $\Sigma \models \alpha$ (read: Σ logically entails α) to mean that for every e and w, if $e, w \models \alpha'$ for every $\alpha' \in \Sigma$, then $e, w \models \alpha$. Finally, we write $\models \alpha$ (read: α is valid) to mean $\{\} \models \alpha$.

4.3 Knowledge

At this point we will not go into a detailed discussion of the properties of $\mathcal{ES}$. Instead we will focus on knowledge as a first example of how the semantics of $\mathcal{ES}$ allows us to prove properties with relative ease. A more complete picture of $\mathcal{ES}$ will emerge later when we establish a formal connection with Reiter's situation calculus.

The interpretation of knowledge in $\mathcal{ES}$ is just a special case of possible-world semantics [27,25]. In particular, as we model knowledge as a set of "worlds", it is not surprising that we obtain the usual properties of *weak S5* or *K45* [16]. Since we assume a fixed universe of discourse, the Barcan formula for knowledge (Property 4 of the following theorem) and its existential version (Property 5) hold as well. Moreover, these properties hold after any number of actions have been performed.

Theorem 2

1. $\models \Box(\mathit{Know}(\alpha) \wedge \mathit{Know}(\alpha \supset \beta) \supset \mathit{Know}(\beta))$;
2. $\models \Box(\mathit{Know}(\alpha) \supset \mathit{Know}(\mathit{Know}(\alpha)))$;
3. $\models \Box(\neg\mathit{Know}(\alpha) \supset \mathit{Know}(\neg\mathit{Know}(\alpha)))$;
4. $\models \Box(\forall x.\mathit{Know}(\alpha) \supset \mathit{Know}(\forall x.\alpha))$;
5. $\models \Box(\exists x.\mathit{Know}(\alpha) \supset \mathit{Know}(\exists x.\alpha))$.

Proof: The proof is fairly straightforward. Here we only consider (3.) and (4.).

[9] In [29], compatibility was extended to account for sensing actions as well, an issue we ignore here, just as we did earlier when introducing the original situation calculus.

3. Let $e, w, z \models \neg \mathit{Know}(\alpha)$. Thus for some w', $w' \simeq w$, $w' \in e$ and $e, w', z \not\models \alpha$. Let w'' be any world such that $w'' \simeq w'$ and $w'' \in e$. Clearly, $e, w'', z \models \neg \mathit{Know}(\alpha)$. Since $w'' \simeq w$, $e, w, z \models \mathit{Know}(\neg \mathit{Know}(\alpha))$ follows.
4. Let $e, w, z \models \forall x.\mathit{Know}(\alpha)$. Hence for all $n \in \mathcal{N}$ of the right sort, $e, w, z \models \mathit{Know}(\alpha^x_n)$ and thus for all $w' \simeq w$, if $w' \in e$ then for all $n \in \mathcal{N}$ of the right sort, $e, w, z \models \alpha^x_n$, from which $e, w, z \models \mathit{Know}(\forall x.\alpha)$ follows. □

We remark that the converse of the Barcan formula (Property 4) holds as well. However, note that this is not the case for Property 5: $\Box(\mathit{Know}(\exists x.\alpha) \supset \exists x.\mathit{Know}(\alpha))$ is not valid in general. Despite the fact that quantification is understood substitutionally, knowing that someone satisfies α does not entail knowing who that individual is, just as it should be.

The following property refers to what is sometimes called the *determinacy of knowledge.*

Theorem 3 *Suppose α is an objective sentence and β is an objective formula with one free variable x, such that $\models \mathit{Know}(\alpha) \supset \exists x.\mathit{Know}(\beta)$. Then for some standard name n, $\models \mathit{Know}(\alpha) \supset \mathit{Know}(\beta^x_n)$.*

Proof Suppose not. Then for every n (of the right sort), $\mathit{Know}(\alpha)$ does not entail $\mathit{Know}(\beta^x_n)$, and so, by the Lemma below, α does not entail β^x_n. So for every n, there is a world w_n such that $w_n \models (\alpha \wedge \neg\beta^x_n)$. Let $e = \{w_n \mid n \text{ a standard name}\}$. Then we have that $e \models \mathit{Know}(\alpha)$ and for every standard name n, $e \models \neg\mathit{Know}(\beta^x_n)$, and so $e \models \forall x.\neg\mathit{Know}(\beta)$. This contradicts the fact that $\mathit{Know}(\alpha)$ entails $\exists x.\mathit{Know}(\beta)$.

Lemma 1 *If α and β are objective, and $\models (\alpha \supset \beta)$, then $\models (\mathit{Know}(\alpha) \supset \mathit{Know}(\beta))$.*

Proof Suppose that some $e \models \mathit{Know}(\alpha)$. Then for every $w \in e$, $w \models \alpha$. Then for every $w \in e$, $w \models \beta$. Thus $e \models \mathit{Know}(\beta)$.

For a modal logician neither the result nor the proof may seem surprising. What is noteworthy though is that in the original situation calculus the proof requires a complicated multi-page argument involving Craig's Interpolation Lemma [45]. This is perhaps another indication that a modal situation calculus may be easier to work with.

5 The Connection with Reiter's Situation Calculus

In [28] it was shown how to define basic action theories and regression in a way very similar to what we saw in Section 2.1. We will not repeat that exercise here except to note that it is the ability to quantify over action modalities which allows us to model successor state axioms with ease in the new language. For example, in our blocks world domain the successor state axiom for *OnFloor* would now be this:

$$\begin{aligned}\Box\forall a, x.[a]\,\mathit{OnFloor}(x) \equiv a = \mathit{putonfloor}(x) \vee \\ \mathit{OnFloor}(x) \wedge a \neq \mathit{pickup}(x).\end{aligned}$$

Note the use of $\Box$ and $[a]$ here, which allows us to state how the fluent changes or does not change after an arbitrary action is performed in any situation.

Despite this nice correspondence between $\mathcal{ES}$ and Reiter's situation calculus regarding the representation of action theories, there is still the question of correctness. Or, how do the valid sentences of $\mathcal{ES}$ relate to what we can conclude from Reiter's foundational axioms? In [29] it is shown that this relationship is indeed tight. They first present a translation from formulas in $\mathcal{ES}$ to the situation calculus and then show that a sentence is valid in $\mathcal{ES}$ iff its translation follows from Reiter's foundational axioms plus axioms which account for the fact that $\mathcal{ES}$ requires a countably infinite domain and that equality is interpreted as identity. As this result provides a nice bridge between the modal situation calculus and the original, we will restate it here and begin with the translation.

5.1 The translation

Before describing Υ, we present the translation from $\mathcal{ES}$ into the situation calculus. In the simplest case, the idea is that a formula like $\mathit{OnTable}(B)$ will be mapped to the situation calculus formula $\mathit{OnTable}(B, S_0)$, where we have restored the distinguished situation term S_0 for the fluent. Similarly, the formula $[\mathit{pickup}(B)]\neg(\mathit{OnTable}(B))$ will be mapped to $\neg(\mathit{OnTable}(\mathit{do}(\mathit{pickup}(B), S_0)))$. So $\mathcal{ES}$ formulas can be thought of as "situation-suppressed" (in situation-calculus terminology) and the * mapping restores the situation argument to the fluents, leaving the rigids unchanged.

More precisely, we have the following:

Definition 4 Let α be any term or formula of $\mathcal{ES}$ without standard names. The expression α^* is defined as $\alpha[S_0]$ where, for any situation term σ, $\alpha[\sigma]$ is defined inductively by:

1. $v[\sigma]$, where v is a first-order variable, is v;
2. $g(t_1, \ldots, t_k)[\sigma]$, where g is a rigid function is $g(t_1, \ldots, t_k)$;
3. $F(t_1, \ldots, t_k)[\sigma]$, where F is a fluent predicate is $F(t_1, \ldots, t_k, \sigma)$;
4. $(t_1 = t_2)[\sigma]$ is $(t_1 = t_2)$;
5. $([t]\alpha)[\sigma]$ is $\alpha[\mathit{do}(t[\sigma], \sigma)]$;
6. $(\alpha \wedge \beta)[\sigma]$ is $(\alpha[\sigma] \wedge \beta[\sigma])$
7. $(\neg\alpha)[\sigma]$ is $\neg\alpha[\sigma]$;
8. $(\forall v. \alpha)[\sigma]$ is $\forall v. \alpha[\sigma]$;
9. $(\Box\alpha)[\sigma]$ is $\forall s'(\sigma \sqsubseteq s' \supset \alpha[s'])$;
10. $\mathit{Know}(\alpha)[\sigma]$ is $\mathit{Knows}(\alpha[\mathit{now}], \sigma)$.

Note that the translation of $\Box\alpha$ introduces quantification over situations, where the introduced variable s' is assumed to be one that does not appear in situation term σ.

5.2 The axioms and the embedding theorem

The axioms we assume in Υ are the following:

1. domain of objects is countably infinite;[10]
2. domain of actions is countably infinite (as above);
3. equality is the identity relation:
   $\forall x \forall y. (x = y) \equiv \forall Q(Q(x) \equiv Q(y))$.

[10] Since the axioms defining a countably infinite set are somewhat cumbersome and would only distract at this point, we defer them to the end of the discussion of Υ.

4. the K predicate:[11] $\forall s' \forall s.\, K(s', s) \;\equiv\; \forall P(\ldots \supset P(s', s))$,
   where the ellipsis stands for the universal closure of
   $[K(s_1, S_0) \wedge \mathit{Ini}(s_2) \;\supset\; P(s_1, s_2)] \quad \wedge$
   $[P(s_1, s_2) \supset P(do(a, s_1), do(a, s_2))]$.

Axioms (1) and (2) talk about the cardinality of the set of objects and actions respectively: they are both countable and infinite. The countability aspect is not very controversial. In the first-order case, every satisfiable set of sentences is satisfiable in a countable domain, and we do not expect users of the situation calculus to use second-order logic to defeat this. Note that this does not rule out having theories that talk about real numbers or other continuous phenomena; it simply rules out using second-order logic to force the interpretations of these theories to be uncountable. We can, however, imagine contexts where finiteness might be desirable. In such cases, we can introduce a new predicate O and instead of asserting that there are finitely many objects, assert that there are finitely many objects in O.

As for axiom (3), it is hard imagining anyone taking the negation of this one seriously. The usual first-order axiomatization of equality is often enough, but the intent is invariably for the equality symbol to be understood as the identity relation, which this second-order axiom ensures.

Finally axiom (4) is a second order definition of the K predicate in terms of the value it has at S_0. This is just another way of capturing the successor state axiom for K introduced by Scherl and Levesque [46], and the added machinery to make *Knows* be a weak-S5 operator [26]. Other knowledge operators are possible in the situation calculus, but weak-S5 and its extensions (such as strong-S5) are the most often used.

The last missing piece are the axioms asserting the countability of objects and actions. Here is one way of specifying these for objects:

$$\begin{aligned}
&\exists Q.\, \forall x Q(x) \;\wedge\; \mathit{Inf}(Q) \;\wedge\; \mathit{Cnt}(Q) \quad \text{where} \\
&\qquad \mathit{Cnt}(Q) \overset{def}{=} \forall Q'\, (Q' \subsetneq Q \;\wedge\; \mathit{Inf}(Q') \supset Q \le Q') \\
&\qquad \mathit{Inf}(Q) \overset{def}{=} \exists Q'.\, Q' \subsetneq Q \;\wedge\; Q \le Q' \\
&\qquad Q \subsetneq Q' \overset{def}{=} \forall x.\, Q(x) \supset Q'(x) \;\wedge\; \exists x.(Q(x) \;\wedge\; \neg Q'(x)) \\
&\qquad Q \le Q' \overset{def}{=} \exists R.\, \forall x\, (Q(x) \supset \exists y\, Q'(y) \wedge R(x, y)) \\
&\qquad\qquad\qquad\quad \wedge\; \forall x, x', y.\, R(x, y) \wedge R(x', y) \supset x = x'.
\end{aligned}$$

Starting from the bottom, $Q \le Q'$ says that the set Q' is no smaller than Q, that is, there is a 1-1 mapping from Q to Q'; $Q \subsetneq Q'$ says that Q is a proper subset of Q'; *Inf*(Q) says that Q is infinite if it contains a proper subset Q' which is no smaller than Q itself; *Cnt*(Q) says that Q is countable if every infinite proper subset is no smaller than Q; finally, the first line says that the set of all objects is countably infinite (here x is assumed to be of type object). To assert the same for actions we simply add another axiom of the form

$\exists Q.\, \forall a Q(a) \;\wedge\; \mathit{Inf}(Q) \;\wedge\; \mathit{Cnt}(Q)$ where a is of type action.

With all the axioms in place we can now state the embedding theorem:

Theorem 4 (Lakemeyer and Levesque) *Let α be any basic sentence of $\mathcal{ES}$ without standard names. Then*

$$\alpha \textit{ is valid} \quad \textit{iff} \quad \Sigma \cup \Upsilon \models_{\mathrm{FOL}} \alpha^*.$$

[11] We let *Ini*(t) be an abbreviation for the situation calculus formula $\forall a \forall s (t \neq do(a, s))$. In this version of the axiom, we ignore the correspondence between K and *Poss*.

Recently [30], the authors obtained an even stronger result in that the equivalence holds when Σ is replaced by an axiom defining $\sqsubseteq$. In other words, as far as the fragment of the original situation calculus as defined by $\mathcal{ES}$ is concerned, Reiter's foundational axioms play essentially no role.

A sentence which has no straightforward counterpart in $\mathcal{ES}$ is

$$\forall s \exists s'. s \sqsubseteq s' \land (s \neq s') \land F(s) \equiv F(s'),$$

which says that from every situation another situation is reachable that agrees on the truth value of F. However, as we showed in [29], this sentence can be expressed if we add to $\mathcal{ES}$ an explicit encoding of action sequences. Indeed, with this trick we were able to come up with a backward translation from the situation calculus to $\mathcal{ES}$, which covers the entire *rooted* situation calculus with knowledge. Here rooted means that quantification over situations is restricted to formulas of the form $\forall s.\sigma \sqsubseteq s \supset \alpha$, where σ is a situation term. Note, in particular, that the rooted situation calculus without knowledge is equivalent with Reiter's original version (simply take σ to be S_0). So, with a little bit of extra effort, $\mathcal{ES}$ has almost the same expressive power as all of the situation calculus.

5.3 Branching time

Let us now see how to integrate branching time into $\mathcal{ES}$. We start with the definition of a temporal sub-language.

Definition 5 The set of temporal formulas is the least set which includes the fluent formulas of $\mathcal{ES}$, is closed under $\neg, \land$, and $\forall$, and for any temporal formulas ϕ and ψ and action a, $\boldsymbol{E}\phi, \boldsymbol{X}\phi, \boldsymbol{F}\phi, \phi\boldsymbol{U}\psi$, and $occ(a)$ are in the set.

Compared to BDFs (see page 7) the only new operator is the path quantifier $\boldsymbol{E}$, and $\boldsymbol{E}\phi$ should be read as "ϕ is true on some path." We will also write $\boldsymbol{A}\phi$ as an abbreviation for $\neg\boldsymbol{E}\neg\phi$ and, as before, $\boldsymbol{G}\phi$ for $\neg\boldsymbol{F}\neg\phi$. Here is an example of what can be expressed in the new language:

$$\boldsymbol{E}(\forall x.\boldsymbol{G}(\mathit{OnTable}(x) \supset \boldsymbol{F}\mathit{OnFloor}(x))),$$

which can be read as "there is a path such that whenever an object is on the table, then it will be on the floor some time after that." Another example is

$$\boldsymbol{EG}(\forall a.occ(a) \supset \mathit{Poss}(a)),$$

which can be read as "there is a path along which all actions are executable."

In contrast to BDFs, whose semantics is given in terms of situation calculus formulas, we will give meaning to the new operators in terms of the worlds we introduced for $\mathcal{ES}$. For that we first need to introduce the notion of a *path,* which we take to be an infinite sequence of standard action names. We denote paths by π and let Π be the set of all paths. If z is a finite sequence of action names and π a path, then $z \cdot \pi$ is the path generated by the concatenation of z and π.

The truth of a temporal formula is defined with respect to a world, a finite sequence of actions z and a path π. Intuitively, z tells us how far world w has evolved already, and π tell us which actions are still to come. (As we left out knowledge from temporal formulas, there is no need to include an epistemic state e.)

1. $w, z, \pi \models F(t_1, \ldots, t_n)$ iff $w[F(n_1, \ldots, n_k), z] = 1$, where $n_i = |t_i|_w$;
2. $w, z.\pi \models (t_1 = t_2)$ iff n_1 and n_2 are identical, where $n_i = |t_i|_w$;
3. $w, z.\pi \models occ(t)$ iff $z = z' \cdot n$ and $n = |t|_w$;
4. $w, z, \pi \models (\phi \wedge \psi)$ iff $w, z, \pi \models \phi$ and $w, z, \pi \models \psi$;
5. $w, z, \pi \models \neg\phi$ iff $w, z, \pi \not\models \phi$;
6. $w, z, \pi \models \forall x.\, \phi$ iff $w, z, \pi \models \phi^x_n$, for every std. name n of the right sort;
7. $w, z, \pi \models \boldsymbol{E}\phi$ iff $w, z, \pi' \models \phi$ for some path π';
8. $w, z, \pi \models \boldsymbol{X}\phi$ iff $w, z \cdot n, \pi' \models \phi$ for $\pi = n \cdot \pi'$;
9. $w, z, \pi \models \boldsymbol{F}\phi$ iff $w, z \cdot z', \pi' \models \phi$ for some z' such that $\pi = z' \cdot \pi'$;
10. $w, z, \pi \models \phi \boldsymbol{U} \psi$ iff $w, z \cdot z', \pi' \models \psi$ for some z' such that $\pi = z' \cdot \pi'$ and for all $z'' \neq z'$ and z''' with $z' = z'' \cdot z'''$, $\quad w, z \cdot z'', z''' \cdot \pi' \models \phi$.

What is still missing is an embedding of the temporal language into $\mathcal{ES}$. For simplicity we will assume that all temporal expressions mentioned in $\mathcal{ES}$ are of the form $\boldsymbol{E}\phi$. In other words, the language of temporal $\mathcal{ES}$ is defined exactly as in Definition 3 except that we add:

5. If $\boldsymbol{E}\phi$ is a temporal formula, then it is also a formula (of temporal $\mathcal{ES}$).

To give meaning to the extended language, we only need to add a rule to the original semantics of $\mathcal{ES}$ that deals with temporal formulas:

9. $e, w, z \models \boldsymbol{E}\phi$ iff $w, z, \pi \models \phi$ for some path π.

Logical entailment and validity for temporal $\mathcal{ES}$ are defined as before.

We will not go into a detailed discussion of the properties of temporal formulas except to note that the temporal operators together with the special *occ* predicate are powerful enough to simulate the action operators of $\mathcal{ES}$. For example, if ϕ is a fluent sentence and t a closed action term, then the following sentences are easily seen to be valid:

1. $\Box\phi \equiv \boldsymbol{AG}\phi$;
2. $[t]\phi \equiv \boldsymbol{EX}\, occ(t) \wedge \phi$.

So it seems that, if we were to extend temporal formulas with epistemic operators, we could recreate all of $\mathcal{ES}$ within the temporal fragment alone.

5.4 Related work

This paper has started from the original situation calculus and then recast a significant fragment of it within modal logic. Another partial translation from the situation calculus to modal logic in the context of belief change is given by Demolombe [12]. In contrast to $\mathcal{ES}$, the work is purely axiomatic and no quantification over action is considered. In [6], the situation calculus is embedded in Hybrid Logic [6], a variant of modal logic which was inspired by the work on tense logic by Prior [42]. In contrast to [12] and $\mathcal{ES}$, however, situations remain part of the language.

Instead of starting with the situation calculus and then moving towards modal logic, the opposite direction has also been pursued. Starting with dynamic logic [23], features were added to close the gap between modal logic and the situation calculus. In early work, Castilho et al. [10] add a dependence relation between actions and fluents to partially address the frame problem. In [12] Reiter-style regression is incorporated into dynamic logic. The authors also consider formulas of the form $\forall a.[a]\alpha$ in a way similar to $\mathcal{ES}$, but remain propositional otherwise. In [24] knowledge is added to dynamic logic, and in [14] the regression idea of [12] is extended to this more expressive language. The authors also obtain optimality results which make use of a reduction to public announcement logic [3,36].

6 Conclusions

Starting with Reiter's influential take on the situation calculus, we discussed how some issues dear to the heart of modal logicians, like knowledge and time, are represented in this formalism. We observed that, despite the obvious branching time structure of situations, it is not that easy to actually introduce branching time operators into the situation calculus. This led us to a recent modal variant, where situations are banned from the language. Advantages are that it captures most of what the situation calculus is actually used for and combines it with the crispness of modal logic. In particular, it turned out to be fairly straightforward to add branching time to the new logic. Recently, Classen and Lakemeyer [11] used these ideas to prove properties of non-terminating Golog programs.

As pointed out at the end of the previous section, there has been other work capturing aspects of the situation calculus within modal logic, mostly with modal logic being the starting point. Johan van Benthem recently also observed that there is a definite trend of convergence between the two camps, despite their "cultural" differences [4]. Perhaps the present paper can provide additional material for building this bridge.

Acknowledgements

This paper owes much if not all to my long-time collaborator Hector Levesque. It was him who had the initial idea to turn Reiter's situation calculus into the modal logic $\mathcal{ES}$.

References

1. F. Bacchus, J.Y. Halpern, and H. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence 111(1-2)*, (1999)
2. F.Bacchus, F. Kabanza, Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2), 123–191 (2000)
3. A. Baltag, L. S. Moss, and S. Solecki, The logic of public announcements, common knowledge, and private suspicions. Proc. of the *7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-VII)*, 43–56 (1998)
4. J. van Benthem, Situation calculus meets modal logic. In L. Morgenstern, V. Lifshitz, and Sheila McIlraith (Eds.), Special Issue in Honor of John McCarthy, *Artificial Intelligence*, Elsevier, (to appear)
5. M. Bienvenu, Ch. Fritz, and S. A. McIlraith, Planning with Qualitative Temporal Preferences. *Proc. of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR'2004)*, AAAI Press, 134–144 (2006)
6. P. Blackburn, J. Kamps, and M. Marx, Situation calculus as hybrid logic: First steps. In P. Brazdil and A. Jorge (Eds.) *Progress in Artificial Intelligence,* Lecture Notes in Artificial Intelligence 2258, Springer Verlag, 253-260 (2001)

7. C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proc. of AAAI-00*, 355–362. AAAI Press (2000)
8. W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2) (2000)
9. A. Carbone, A. Finzi, A. Orlandini, F. Pirri, and G. Ugazio, Augmenting situation awareness via model-based control in rescue robots, In *Proc. of IROS-2005,* Edmonton, Canada, (2005)
10. M. A. Castilho, O. Gasquet, and A. Herzig, Formalizing Action and Change in Modal Logic I: the frame problem. *Journal of Logic and Computation*, 9(5), 701–735 (1999)
11. J. Classen and G. Lakemeyer, A Logic for Non-Terminating Golog Programs. In *Proc. of the Eleventh Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, AAAI Press, 589–599 (2008)
12. R. Demolombe, Belief change: from Situation Calculus to Modal Logic. *Journal of Applied Non-Classical Logics*, 13(2), 187-198 (2003)
13. R. Demolombe, A. Herzig, and I. Varzinczak, Regression in Modal Logic. *Journal of Applied Non-Classical Logics,* 13(2), 165–18 (2003)
14. H. van Ditmarsch, A. Herzig, and T. de Lima, Optimal regression for reasoning about knowledge and actions. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, 1070–1075 (2007)
15. E. Emerson and J. Y. Halpern, "Sometimes" and "not never" revisited: on branching versus linear time temporal logic, *J. of the ACM*, 33(1), 151–178 (1986)
16. R. Fagin, J. Halpern, Y. Moses and M. Vardi, *Reasoning about Knowledge.* MIT Press, Cambridge (1995)
17. A. Ferrein, C. Fritz, and G. Lakemeyer. On-line decision-theoretic Golog for unpredictable domains. In S. Biundo, T. Frhwirt, and G. Palm, editors, *KI2004: Advances in Artificial Intelligence*. Springer (2004)
18. A. Gabaldon, Precondition Control and the Progression Algorithm. *Proc. of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR'2004)*, AAAI Press, 634–643 (2004)
19. G. De Giacomo, Y. Lesperance, and H. J Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121:109–169 (2000)
20. H. Grosskreutz and G. Lakemeyer. cc-Golog – An Action Language with Continous Change. *Logic Journal of the IGPL* (2002)
21. R. Fikes, N. J. Nilsson, STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Proc. of the Second International Joint Conference on Artificial Intelligence (IJCAI-71)*, 608–620 (1971)
22. C. Green, Application of Theorem Proving to Problem Solving. *Proc. of the 1st International Joint Conference on Artificial Intelligence (IJCAI-69)*, Washington, DC, 219–240 (1969)
23. D. Harel, Dynamic Logic, in D. Gabbay and F. Guenther (Eds.), *Handbook of Philosophical Logic*, Vol. 2, D. Reidel Publishing Company, 497–604 (1984)
24. A. Herzig, J. Lang, D. Longin, and T. Polacsek, A logic for planning under partial observability. In *Proc. AAAI-2000*, AAAI Press (2000)
25. J. Hintikka,, *Knowledge and Belief.* Cornell University Press, Ithaca (1962)
26. G. Hughes, and M. Cresswell, *An Introduction to Modal Logic.* Methuen and Co., London (1968)
27. S. A. Kripke, Semantical considerations on modal logic. *Acta Philosophica Fennica* 16, pp. 83–94 (1963)
28. G. Lakemeyer and H. J. Levesque, Situations si, situation terms no. In *Proc. of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*, AAAI Press, 516–526 (2004)
29. G. Lakemeyer and H. J. Levesque, A useful fragment of the situation calculus. In *Proc. of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, 490–496 (2005)
30. G. Lakemeyer and H. J. Levesque, A semantic characterization of a useful fragment of the situation calculus with knowledge. In L. Morgenstern, V. Lifshitz, and Sheila McIlraith (Eds.), Special Issue in Honor of John McCarthy, *Artificial Intelligence,* Elsevier (to appear)
31. H. J. Levesque and G. Lakemeyer, *The Logic of Knowledge Bases*, MIT Press (2001)

32. H. J. Levesque and G. Lakemeyer, Cognitive Robotics. In Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, *Handbook of Knowledge Representation,* Elsevier, 869–886 (2008)
33. H. J. Levesque, F. Pirri, and R. Reiter, Foundations for the Situation Calculus. Linköping Electronic Articles in Computer and Information Science, 3(18) (1998)
34. H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl., Golog: A logic programming language for dynamic domains. *Journal of Logic Programming,* 31, 59–84 (1997)
35. F. Lin and R. Reiter, How to Progress a Database. *Artificial Intelligence*, 92, pp.131-167 (1997)
36. C. Lutz, Complexity and succinctness of public announcement logic. Proc. of the *Fifth International Joint Conference On Autonomous Agents and Multiagent Systems*, 137–143 (2006)
37. J. McCarthy, *Situations, Actions and Causal Laws.* Technical Report, Stanford University, 1963. Also in M. Minsky (ed.), *Semantic Information Processing*, MIT Press, Cambridge, MA, 410–417 (1968)
38. S. McIlraith and T. C. Son, Adapting Golog for Composition of Semantic Web Services, *Proc. of the Eighth International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, San Francisco, 482–493 (2002)
39. R. C. Moore, A formal theory of knowledge and action. J. R. Hobbs and R. C. Moore (eds.), *Formal Theories of the Commonsense World.* Ablex, Norwood, NJ, 319–358 (1985)
40. J. Pinto, Integrating discrete and continuous change in a logical framework. *Computational Intelligence, 14(1)* (1997)
41. V. R. Pratt, Semantical considerations on Floyd-Hoare logic. In *Proc. 17th Ann. IEEE Symp. on Foundations of Computer Science,* IEEE Computer Society Press, 109–121 (1976)
42. A. Prior, *Past, Present and Future.* Oxford University Press (1967)
43. R. Reiter, The Frame Problem in the Situation Calculus: A simple Solution (sometimes) and a Completeness Result for Goal Regression. In V. Lifschitz (ed.), *Artificial Intelligence and Mathematical Theory of Computation*, Academic Press, pp. 359–380 (1991)
44. R. Reiter, *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems.* MIT Press (2001)
45. R. Reiter, On knowledge-based programming with sensing in the situation calculus. *ACM Transactions on Computational Logic,* 433-457 (2001)
46. R. B. Scherl and H. J. Levesque, Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1-2), 1-39 (2003)
47. M. Shanahan, *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press (1997)