A semantic characterization of a useful fragment of the situation calculus with knowledge

Gerhard Lakemeyer\textsuperscript{a,\textdagger}, Hector J. Levesque\textsuperscript{b}

\textsuperscript{a} Dept. of Computer Science, RWTH Aachen, 52056 Aachen, Germany
\textsuperscript{b} Dept. of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G6

\textbf{A R T I C L E   I N F O}

Article history:
Available online 3 April 2010

Keywords:
Knowledge representation
Reasoning about action

\textbf{A B S T R A C T}

The situation calculus, as proposed by McCarthy and Hayes, and developed over the last decade by Reiter and co-workers, is reconsidered. A new logical variant called ES is proposed that captures much of the expressive power of the original, but where certain technical results are much more easily proved. This is illustrated using two existing non-trivial results: the determinacy of knowledge theorem of Reiter and the regression theorem, which reduces reasoning about the future to reasoning about the initial situation. Furthermore, we show the correctness of our approach by embedding ES in Reiter’s situation calculus.

\textcopyright{} 2010 Elsevier B.V. All rights reserved.

1. Introduction

Among the many contributions of John McCarthy, the formalism of the situation calculus has proved to be an extremely useful tool for reasoning precisely about action and change. It was originally proposed in [29,30] as a dialect of first-order logic. A second-order refinement of the language, developed by Reiter and his colleagues [36], forms the theoretical and implementation foundation for Golog [27], a language for the high-level control of robots and other agents (see, for example, [2,31]). Over the past decade, a number of extensions have been proposed to deal with issues such as time, natural actions, knowledge of agents, numerical uncertainty, or utilities (see [36] and the references therein).

As a formalism, the situation calculus is based on axioms. In Reiter’s formulation, which is also our starting point, these take the form of so-called basic action theories. These consist of a number of foundational axioms, which define the space of situations, unique-name axioms for actions, axioms describing action preconditions and effects, and axioms about the initial situation.

What makes basic action theories particularly useful is the formulation of action effects in terms of successor state axioms, which not only provide a simple solution to the frame problem [35] but also allow the use of regression-based reasoning, which has been used in planning [8] and forms the core of every Golog interpreter, for example. Derivations using regression are simple, clear, and computationally feasible.

Since the situation calculus is defined axiomatically, no special semantics is needed. Tarskian models suffice, provided they satisfy the foundational axioms. When the focus is on logical entailments, which is the case in the execution of Golog programs, for example, this approach seems perfectly adequate.

However, when we wish to consider theoretical questions about basic action theories that are not direct entailment questions, problems arise. For example, suppose we are doing an analysis of our system, and want to know, if whenever Theory\textsuperscript{1} entails Formula\textsuperscript{1}, is it also true that Theory\textsuperscript{2} entails Formula\textsuperscript{2}? Here we can run into serious complications in an
axiomatic setting unless there are ways to take derivations of Formula1 from Theory1 and convert them into derivations of Formula2 from Theory2. Similar issues arise with consistency questions.

For instance, consider the epistemic extension of the situation calculus, as introduced by Moore and later extended by Scherl and Levesque [32,39]. If Know(A) entails (Know(B) ∨ Know(C)) in this theory, is it also true that Know(A) entails Know(B) or Know(A) entails Know(C)? For restricted A, B, C, the answer is yes, but the proof by Reiter requires a multi-page argument using considerable proof-theoretic machinery, including Craig’s Interpolation Lemma [37].

One might wonder whether a semantic proof using Tarski structures would be any easier. The answer, in short, is no. The problem is that different Tarski structures can have different domains and considerable effort is required to standardize the domains, identify the situations, and amalgamate multiple structures into a single structure that satisfies the foundational axioms. While certainly possible, the argument is again long and complicated.

In contrast, in the epistemic logic KLC [22], the semantic proof of the above determinacy of knowledge theorem is simple, clear and direct. One reason for this is the use of a semantic formulation involving possible worlds for knowledge [12,7]. Typical of these formalisms, situations and possible worlds are not reified in the language itself. Beyond this, however, a major factor in the simplicity of proofs in KLC (and its extension, OL) is the use of standard names, which allows a substitutional interpretation of the first-order quantifiers. While there have been philosophical arguments against substitutional quantification [18], our experience has been that its technical simplicity has been of tremendous help in tackling issues such as quantifying-in [15], which are rarely addressed in other formalisms.

Since KLC only deals with static knowledge bases, an amalgamation of KLC and the situation calculus was previously proposed [19]. However, this formalization kept situations reified, did not allow substitutional quantification, and the definition of knowledge required second-order logic, all of which again complicated the proofs considerably, even semantic ones.

In this paper, we propose a rather different amalgamation of KLC and the situation calculus called ES. The idea is to keep the simplicity of KLC, and while dropping some of the expressiveness of the ordinary situation calculus, retain its main benefits, like successor state axioms to solve the frame problem and regression-based reasoning. In particular, we will use a possible-world semantics where situations are part of the semantics but do not appear as terms in the language. In order to represent what is true in a situation after a number of actions have occurred, we use special modal operators. For example, we will have formulas like those of standard dynamic logic [33,10], such as

\[ \text{[forward]} \text{[forward]} \text{distance} = 4 \]

to say that a robot is four units away from a wall after moving forward twice (see Fig. 1 for an illustration). In contrast to other modal approaches such as [3,11,5] but similar to [6], we also allow formulas of the form \( \forall \alpha \exists \alpha \text{([distance = x])} \equiv \phi \), where modalities contain (action) variables. This feature will be key in reconstructing Reiter’s basic action theories in our language. Moreover, unlike standard modal logics (including dynamic logics), we will be able to use a substitutional interpretation for first-order quantifiers. This is perhaps the main reason why we cannot afford situation terms as part of our language. The epistemic situation calculus requires us to consider an uncountable number of initial situations (see [26] for a second-order foundational axiom that makes this explicit). In a language with only countably many situation terms, this would preclude a substitutional interpretation of quantifiers.

Yielding much simpler proofs (like the determinacy of knowledge and the correctness of regression) still leaves open the question of the overall correctness of the approach. In other words, is ES really a faithful reconstruction of the situation calculus? We will prove that it is by providing an embedding of ES in Reiter’s version of the situation calculus, showing that the valid sentences of ES can be cast as entailments in Reiter’s original version (modulo some modest assumptions). This shows that ES is a notational variant for a fragment of the situation calculus that can be given a clean and workable semantics. In addition, this result allows us to automatically transfer results obtained for ES to that fragment of the situation calculus, which is expressive enough to formulate basic action theories, and more.

The rest of the paper is organized as follows. In the next section we introduce the syntax and semantics of ES and discuss some of the properties of knowledge. In Section 3, we introduce the ES-version of Reiter’s basic action theories.

---

1 Roughly speaking, this amounts to assuming at the outset that the domain of quantification is countably infinite and that there is a set of special constants called standard names uniquely denoting each element of the domain. A first-order universal sentence then ends up being true iff every instance of the sentence, where a standard name substitutes for the variable, is true.
followed by regression theorems for the non-epistemic and epistemic case. Section 5 provides the results on embedding $\mathcal{ES}$ in the situation calculus. We end the paper with a discussion of related work, only-knowing, and concluding remarks.

2. The logic $\mathcal{ES}$

The language is a second-order modal dialect with equality and sorts of type object and action. Before presenting the formal details, here are the main features:

- **Standard names**: Unlike other languages (but similar to $\mathcal{KL}$), 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 allows first-order quantification to be understood substitutionally. Equality can also be given a simpler treatment: every ground term will have a coreferring standard name, and two terms are considered identical if their coreferring standard names are identical.
- **Fluent and rigid functions and predicates**: The language also contains both fluent and rigid predicate and function symbols. Fluents vary as the result of actions and have values that may be unknown, but rigids do not. These are present in the original situation calculus, of course. For example, we might have a sentence like this in the situation calculus:

  \[
  \text{Fragile}(c) \land \neg \text{Broken}(c, S_0).
  \]

Here we can see that the first predicate is rigid and the second one is fluent just by seeing if the last argument is a situation. In our case, we do not have situation terms, and so we will need to distinguish the two sorts of predicates syntactically and semantically. Furthermore, for second-order quantification, we will need to distinguish rigid and fluent predicate variables as well.\(^2\)
- **Knowledge and truth**: The language includes a modal operator Know for knowledge. This allows us to distinguish between sentences that are true and sentences that are known (by some implicit agent). For example, we can model situations where a robot is close to a wall but does not yet know it. We can also model situations where a robot has false beliefs about its world or how its world changes. The connection between knowledge and truth is made with sensing. Every action is assumed to have a binary sensing result and after performing the action, the agent learns that the action was possible (as indicated by the Poss predicate) and whether the sensing result for the action was 1 or 0 (as indicated by the SF predicate).

2.1. The language

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

- first-order variables: $x_1, x_2, \ldots, y_1, y_2, \ldots, q_1, q_2, \ldots$;
- fluent second-order variables of arity $k$: $f^1_k, f^2_k, \ldots$;
- rigid second-order variables of arity $k$: $Q^1_k, Q^2_k, \ldots$;
- standard names: $n_1, n_2, \ldots$ for objects and actions;
- fluent function symbols of arity $k$: $f^1_k, f^2_k, \ldots$; for example, distance;
- rigid function symbols of arity $k$: $g^1_k, g^2_k, \ldots$; for example, forward;
- fluent predicate symbols of arity $k$: $F^1_k, F^2_k, \ldots$; for example, Broken;
- rigid predicate symbols of arity $k$: $C^1_k, C^2_k, \ldots$; for example, BrotherOf;
- connectives and other symbols: $\equiv, \land, \neg, \forall, \text{Know, \Box, round and square parentheses, period, comma}$.

We assume that all action function symbols are rigid and that the fluent predicates include the special predicates Poss and SF.

**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;
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 (fluent or rigid) function symbol and all of the $n_i$ are standard names.

\(^2\) We follow Reiter in including both types of symbols. It is possible to live with just fluents, however, and treat rigids as fluents that happen not to change.
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 \( H \) is a k-ary predicate symbol then \( H(t_1, \ldots, t_k) \) is an (atomic) formula;
2. If \( t_1, \ldots, t_k \) are terms, and \( V \) is a k-ary second-order variable, then \( V(t_1, \ldots, t_k) \) is an (atomic) formula;
3. If \( t_1 \) and \( t_2 \) are terms, then \( (t_1 = t_2) \) is a formula;
4. If \( t \) is an action term and \( \alpha \) is a formula, then \( [t] \alpha \) is a formula;
5. If \( \alpha \) and \( \beta \) are formulas, \( v \) is a first-order variable, and \( V \) is a second-order variable, then the following are also formulas: \( (\alpha \land \beta), \neg \alpha, \forall v. \alpha, \forall \forall \alpha, \square \alpha. \) Know(\alpha).

We read \([t] \alpha\) as "\( \alpha \) holds after action \( t \)", and \( \Box \alpha \) as "\( \alpha \) holds after any sequence of actions." So, for example, here is a successor state axiom in this language (we follow the usual convention of having free variables universally quantified from the outside):

\[
\Box([a] \text{Broken}(x) \equiv (a = \text{drop}(x) \land \text{Fragile}(x)) \lor (\text{Broken}(x) \land a \neq \text{repair}(x)))
\]

In English: after any sequence of actions, an object \( x \) will be broken after doing action \( a \) iff \( a \) is the dropping of \( x \) when \( x \) is fragile or \( x \) was already broken and \( a \) is not the action of repairing it.

As usual, we treat \( (\alpha \lor \beta), (\alpha \supset \beta), (\alpha \equiv \beta), \exists v. \alpha, \) and \( \exists \forall \alpha \) 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 use \( a_n^m \) to mean formula \( \alpha \) with all free occurrences of variable \( x \) replaced by name \( n \). We call a formula without free variables a sentence. By a primitive sentence we mean a formula of the form \( H(n_1, \ldots, n_k) \) where \( H \) is a (fluent or rigid) predicate symbol and all of the \( n_i \) are standard names.

In the following, we will sometimes refer to special sorts of first-order formulas and use the following terminology:

- a formula with no \( \Box \) operators is called bounded;
- a formula with no \( \Box \) or \([t]\) operators is called static;
- a formula with no \( \text{Know} \) operators is called objective;
- a formula with no fluent, \( \Box \), or \([t]\) operators outside the scope of a \( \text{Know} \) is called subjective;
- a formula with no \( \text{Know}, \Box, [t] \), Poss, or \( \text{SF} \) is called a fluent formula.\(^3\)

2.2. The semantics

The main purpose of the semantics we are about to present is to be precise about how we handle fluents, which may vary as the result of actions and whose values may be unknown. Intuitively, to determine whether or not a sentence \( \alpha \) 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 \( w, e, z \models \alpha \). A world determines truth values for the primitive sentences and coreferring 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 and \( \mathcal{Z} \) to \( \mathcal{N} \) (preserving sorts), and satisfying the rigidity constraint: if \( g \) is a rigid function or predicate symbol, then for all \( z \) and \( z' \) in \( \mathcal{Z} \), \( w[g(n_1, \ldots, n_k), z] = w[g(n_1, \ldots, n_k), z'] \);
- an epistemic state \( e \subseteq W \) is any set of worlds.

We extend the idea of coreferring standard names to arbitrary ground terms as follows. Given a term \( t \) without variables, a world \( w \), and an action sequence \( z \), we define \( [t]_{e,w} \) (read: the coreferring standard name for \( t \) given \( w \) and \( z \)) by:

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

So to find a coreferring standard name for \( h(t_1, \ldots, t_k) \), we find coreferring names for the \( t_i \) recursively, and then use the function \( w \) on the resulting primitive term.

\(^3\) In the situation calculus, these correspond to formulas that are uniform in some situation term.
To interpret formulas with free variables, we proceed as follows. First-order variables are handled substitutionally using the standard names. To handle the quantification over second-order variables, we use second-order variable maps defined as follows:

The second-order primitives are formulas of the form $V(n_1, \ldots, n_k)$ where $V$ is a (fluent or rigid) second-order variable and all of the $n_i$ are standard names. A variable map $u$ is a function from worlds, second-order primitives, and $Z$ to $\{0, 1\}$, satisfying the rigidity constraint: if $Q$ is a rigid second-order variable, then for all $w$ and $w'$ in $W$, and all $z$ and $z'$ in $Z$, $u[w, Q(n_1, \ldots, n_k), z] = u[w', Q(n_1, \ldots, n_k), z']$.

Let $u$ and $u'$ be variable maps, and let $V$ be a (fluent or rigid) second-order variable; we write $u' \sim_V u$ to mean that $u$ and $u'$ agree except perhaps on the second-order primitives involving $V$.

Finally, to interpret what is known after a sequence of actions has taken place, we define $w' \simeq_z w$ (read: $w'$ agrees with $w$ agree on the sensing throughout action sequence $z$) inductively by the following:

1. $w' \simeq_\emptyset w$ iff $w'$ and $w$ agree on the value of every primitive rigid term and sentence;
2. $w' \simeq_{z \cdot n} w$ iff $w' \simeq_z w$, $w'[\text{Poss}(n), z] = 1$, and
   \[ w'[\text{SF}(n), z] = w[\text{SF}(n), z]. \]

Note that $\simeq_z$ is not quite an equivalence relation because of the use of Poss here. As will become clearer, this is because we are insisting that the agent comes to believe that Poss was true after performing an action, even in those "non-legal" situations where the action was not possible in reality.4

Putting all these together, here is the semantic definition of truth. Given a sentence $\alpha$ of ES, an epistemic state $e \in W$ and a world $w \in W$, we define $e, w \models \alpha$ (read: $\alpha$ is true at $e$ and $w$) as $e, w, (\langle \rangle, u \models \alpha$ for any second-order variable map $u$, where for any $z \in Z$ we have:

1. $e, w, z, u \models H(t_1, \ldots, t_k)$ iff $w[H(n_1, \ldots, n_k), z] = 1$, where $n_i = [t_i]_w$;
2. $e, w, z, u \models V(t_1, \ldots, t_k)$ iff $u[w, V(n_1, \ldots, n_k), z] = 1$, where $n_i = [t_i]_w$;
3. $e, w, z, u \models (t_1 = t_2)$ iff $n_1$ and $n_2$ are identical, where $n_i = [t_i]_w$;
4. $e, w, z, u \models \exists ! \alpha$ iff $e, w, z \cdot n, u \models \alpha$, where $n = [t^2]_w$;
5. $e, w, z, u \models (\alpha \land \beta)$ iff $e, w, z, u \models \alpha$ and $e, w, z, u \models \beta$;
6. $e, w, z, u \models \neg \alpha$ iff $e, w, z, u \models \alpha$;
7. $e, w, z, u \models \forall x. \alpha$ iff $e, w, z, u \models \alpha^n$, for every standard name $n$ of the right sort;
8. $e, w, z, u \models \forall V. \alpha$ iff $e, w, z, u \models \alpha$, for every $u \sim_V u$;
9. $e, w, z, u \models \Box \alpha$ iff $e, w, z \cdot z', u \models \alpha$, for every $z' \in Z$;
10. $e, w, z, u \models \text{Know}(\alpha)$ iff $e, w', z, u \models \alpha$, for every $w' \in e$ such that $w' \simeq_z w$.

When $\alpha$ is objective (has no Know operators), we can leave out the $e$ and write $w \models \alpha$. Similarly, when $\alpha$ is subjective, we can leave out the $w$ and write $e \models \alpha$. When $\Sigma$ is a set of sentences and $\alpha$ is a sentence, we write $\Sigma \models \alpha$ (read: $\Sigma$ logically entails $\alpha$) 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: $\alpha$ is valid) to mean $\langle \rangle \models \alpha$.

2.3. Knowledge

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

The interpretation of knowledge in ES is just a special case of possible-world semantics [17,12]. One minor feature worth noting is that we do not simply require truth in all elements of $e$, the given set of "possible worlds," as in KL. In fact, $e$ represents the initial state of knowledge, and as knowledge is acquired though action, some of those initial worlds will no longer be considered possible. This is reflected in the $\simeq_z$ relation. In a nutshell, we look for truth in all elements of $e$ that agree with the real world $w$ in terms of sensing. It will then follow that after doing a sequence of actions, the agent will know the correct values of the sensing results in the real world (and everything it can conclude from that).

Regarding the more traditional logical properties of knowledge, it is not surprising that we obtain the usual properties of weak S5 or K45 [7]. 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.

---

4 An alternate account that would state that the agent learns the true value of Poss (analogous to SF) is a bit more cumbersome, but would allow $\simeq_z$ to be a full equivalence relation.
Then we have that

**Proof.**

for situations \([36]\). In fact, for this and the next section we will have no use for second-order logic at all and only consider

the notion of situations our basic action theories do not require foundational axioms like Reiter’s second-order induction axiom

involving Craig’s Interpolation Lemma.

**Theorem 1.**

Suppose not. Then for every \(\alpha, \beta \models \text{Know}(\alpha) \land \text{Know}(\alpha \supset \beta) \Rightarrow \text{Know}(\beta)\).

**Proof.** Let \(e, w, z \models \text{Know}(\alpha) \land \text{Know}(\alpha \supset \beta)\). Then for all \(w' \simeq w, w' \in e\) then \(e, w', z \models (e, \alpha, \beta)\). Hence, \(e, w', z \models \beta\) and, therefore, we have that \(e, w, z \models \text{Know}(\beta)\).

**Theorem 2.** Suppose \(\alpha\) is an objective sentence and \(\beta\) is an objective formula with one free variable \(x\), such that \(\models \text{Know}(\alpha) \supset \text{∃}x. \text{Know}(\alpha)\). Then for some standard name \(n\), \(\models \text{Know}(\alpha) \supset \text{Know}(\beta_n)\).

**Proof.** Suppose not. Then for every \(n\) (of the right sort), \(\text{Know}(\alpha)\) does not entail \(\text{Know}(\beta_n)\), and so, by the lemma below, \(\alpha\) does not entail \(\beta_n\). So for every \(n\), there is a world \(w_n\) such that \(w_n \models (\alpha \land \neg \beta_n)\). Now let \(e = \{w_n \mid n \text{ a standard name}\}\). Then we have that \(e \models \text{Know}(\alpha)\) and for every standard name \(n\), \(e \models \neg \text{Know}(\beta_n)\), and so \(e \models \forall x. \neg \text{Know}(\beta)\). This contradicts the fact that \(\text{Know}(\alpha)\) entails \(\exists x. \text{Know}(\beta)\). □

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

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

This proof is exactly as it would be in \(\mathcal{KL}\). Again it is worth noting that the proof of this theorem in the ordinary situation calculus (for the simpler case involving disjunction rather than existential quantification) is a multi-page argument involving Craig’s Interpolation Lemma.

3. **Basic action theories**

Let us now consider the equivalent of basic action theories of the situation calculus. Since in our logic there is no explicit notion of situations our basic action theories do not require foundational axioms like Reiter’s second-order induction axiom for situations [36]. In fact, for this and the next section we will have no use for second-order logic at all and only consider the first-order fragment of \(\mathcal{ES}\).

**Definition 4.** Given a set of fluents \(F\), a set \(\Sigma \subseteq \mathcal{ES}\) of sentences is called a basic action theory over \(F\) iff \(\Sigma = \Sigma_{\text{una}} \cup \Sigma_0 \cup (\Sigma_{\text{pre}} \cup \Sigma_{\text{post}} \cup \Sigma_{\text{sense}})\) where \(\Sigma\) mentions only fluents in \(F\) and

1. \(\Sigma_{\text{una}}\) is a set of unique names axioms for action functions;
2. \(\Sigma_0\) is any set of fluent sentences;
3. \(\Sigma_{\text{pre}}\) is a singleton sentence of the form \(\Box \text{Poss}(a) \equiv \pi\), where \(\pi\) is a fluent formula\(^5\);

\(^5\) We assume that \(\Box\) has lower syntactic precedence than the logical connectives, so that \(\Box \text{Poss}(a) \equiv \pi\) stands for \(\forall \alpha. \Box \text{Poss}(a) = \pi\) .
4. $\Sigma_{\text{post}}$ is a set of sentences of the form $\Box \lbrack a \rbrack F(\vec{v}) \equiv \gamma_F$ or $\Box \lbrack a \rbrack f(\vec{v}) = y \equiv \gamma_f$, one for each relational fluent $F$ and functional fluent $f$, respectively, and where $\gamma_F$ and $\gamma_f$ are fluent formulas$^6$.

5. $\Sigma_{\text{sense}}$ is a sentence exactly parallel to the one for Poss of the form $\Box SF(a) \equiv \varphi$, where $\varphi$ is a fluent formula.

The idea here is that $\Sigma_0$ expresses what is true initially (in the initial situation). $\Sigma_{\text{pre}}$ is one large precondition axiom, and $\Sigma_{\text{post}}$ is a set of successor state axioms, one per fluent, which incorporate the solution to the frame problem proposed by Reiter [35]. Here we follow the convention of [39] that every action returns a sensing result. $\Sigma_{\text{sense}}$ then captures the outcome of sensing actions. For actions like forward, which do not return any useful sensing information, SF can be defined to be vacuously true (see below for an example)$^7$.

Since an agent's beliefs may differ from what is true, we will, in general, need two basic action theories: $\Sigma$ for what is true in the world, including its dynamics, and $\Sigma'$ for what the agent believes to be true. Except for the unique names assumption for actions, the two are allowed to differ arbitrarily and even contradict each other to allow for false beliefs.

A state of affairs can then be characterized by sentences of the form $\Sigma$ to denote what is actually true, Know($\Sigma'$) to denote what the agent believes to be true, and perhaps a set of sentences $\neg$Know($\phi$) to capture some of the agent's ignorance$^8$.

We will be interested in the what is entailed by such theories.

As an example, imagine a robot that lives in a 1-dimensional world, and that can move towards or away from a fixed wall. The robot also has a sonar sensor that tells it when it gets close to the wall, say, less than 10 units away. See Fig. 1.

We will be interested in the what is entailed by such theories.

Let us consider informally how sensing relates knowledge to truth here. We start in some initial epistemic state $e$ and world $w$. Initially, before any actions have taken place, the action sequence $z$ is $\langle \rangle$. We might suppose that $w[\text{distance}, \langle \rangle] = 6$ as in the diagram. If the robot does not know where it is, there may be a $w^* \in e$ where $w^*[\text{distance}, \langle \rangle] = 13$. Now suppose the robot performs a sonar action. In this case, we would expect that $w[SF(\text{sonar}), \langle \rangle] = 1$, but $w^*[SF(\text{sonar}), \langle \rangle] = 0$. In other words, if the sonar is doing its job, in $w$ it would tell us that the robot is close to the wall and in $w^*$ it would tell us that the robot is far from the wall. So if we now let $z = (\text{sonar})$, we see that $w^* \not\equiv_x w$, since they disagree on the SF value. In fact, for every $w'$ such that $w' \equiv_x w$, we will have that $w'[SF(\text{sonar}), \langle \rangle] = 1$. Since the definition of Know uses $\equiv_x$, when we consider what is known after doing the sonar action, the robot will believe (correctly) that it is close to the wall: $e, w, (\text{sonar}) \models \text{Know(distance < 10)}$.

Let us now make all this precise. We begin our formalization by writing preconditions for the three actions:

\[\Box \text{Poss}(a) \equiv\]
\[a = \text{forward} \land \text{distance } > 0\quad \checkmark\]
\[a = \text{backward} \land \text{TRUE}\quad \checkmark\]
\[a = \text{sonar} \land \text{TRUE}.\]

In other words, while backward and sonar are always possible, forward is executable only when the robot is not already at the wall. Next, we define the sensing results for the actions:

\[\Box SF(a) \equiv\]
\[a = \text{forward} \land \text{TRUE}\quad \checkmark\]
\[a = \text{backward} \land \text{TRUE}\quad \checkmark\]
\[a = \text{sonar} \land \text{distance } < 10.\]

Since backward and forward are not expected to return any useful sensing information, SF is vacuously true for them, while SF(sonar) says that the sonar returns 1 precisely when the distance to the wall is less than 10. Finally, we write a successor state axiom for our only fluent:

\[\Box \lbrack a \rbrack (\text{distance } = x) \equiv\]
\[a = \text{forward} \land \text{distance } = x + 1\quad \checkmark\]

---

$^6$ The $\{\}$ construct has higher precedence than the logical connectives. So $\Box \lbrack a \rbrack F(\vec{v}) \equiv \gamma_F$ abbreviates $\forall a. \Box \lbrack a \rbrack F(\vec{v}). \equiv \gamma_F$.

$^7$ In this paper we restrict ourselves to sensing truth values. See [39] for how to handle arbitrary values.

$^8$ When we use $\Sigma$ as part of a sentence we mean the conjunction of all the finitely many sentences contained in $\Sigma$.

$^9$ Here and below, we use simple arithmetic involving $\prec, +$, and $-$, which can easily be defined in second-order terms with the standard names acting as natural numbers. We omit the details.
\[ a = \text{backward} \land \text{distance} = x - 1 \lor \]
\[ a \neq \text{forward} \land a \neq \text{backward} \land \text{distance} = x. \]

In other words, the distance to the wall increases or decreases by 1 depending on whether backward or forward is executed, or it remains as before for all other actions.

Now we are ready to consider some specifics having to do with what is true initially by defining an action theory. Let Close stand for the formula “distance < 10.” Let \( \phi \) denote the conjunction of the sentences above. We assume that \( \phi \) is true and the robot knows it. We also assume the robot is located initially 6 units away from the wall, but that the robot has no idea where it is. So, we let \( \Sigma = \{ \phi \} \cup \{ \text{distance} = 6 \} \) and \( \Sigma' = \{ \phi \} \). Then we get this:

**Example 1.** The following are logical entailments of

\[ \Sigma \land \text{Know}(\Sigma') \land \forall x. \neg \text{Know}(\text{distance} \neq x): \]

1. \( \text{Close} \land \neg \text{Know}(	ext{Close}) \land [\text{forward}] \neg \text{Know}(	ext{Close}) \)
   the robot is close to the wall, but does not know it, and continues not to know it after moving forward;
2. \([\text{sonar}] (\text{Know}(	ext{Close}) \land [\text{forward}] \text{Know}(	ext{Close})) \)
   after reading the sonar, the robot knows it is close, and continues to know it after moving forward;
3. \([\text{sonar}] [\text{backward}] [-\text{Know}(	ext{Close})] \)
   after reading the sonar and then moving backward, the robot no longer knows that it is close to the wall;
4. \([\text{forward}] [\text{sonar}] \text{Know}(	ext{Close}) \)
   after moving forward and then reading the sonar, the robot knows that it is close to the wall;
5. \([\text{sonar}] [\text{forward}] [\text{backward}] \text{Know}(	ext{Close}) \)
   after reading the sonar, moving forward, and then backward, the robot knows that it is still close to the wall;
6. \([\text{sonar}] \text{Know}[\text{forward}] [\text{Close}] \)
   after reading the sonar, the robot knows that it will remain close after moving forward;
7. \(\neg \text{Know}([\text{sonar}] \text{Know} [\text{Close}]) \)
   the robot does not know initially that it will know that it is close after reading the sonar;
8. \(\text{Know}([\text{sonar}] [\text{Know} [\text{Close}]) \lor \text{Know}(\neg \text{Close})) \)
   the robot does know initially that after reading the sonar, it will then know whether or not it is close to the wall;
9. \(\neg \text{Know}([\text{sonar}] [\text{backward}] \neg \text{Know} [\text{Close}]) \)
   the robot knows initially that it will not know that it is close after reading the sonar and moving backwards.

**Proof.** The proofs of these are similar. Here we will only do item 3. Let \( z = [\text{sonar} \cdot \text{backward}] \), and suppose that \( e, w \models \Sigma \land \text{Know}(\Sigma') \land \forall x. \neg \text{Know}(\text{distance} \neq x) \); we must show that \( e, w, z \models \neg \text{Know}(	ext{Close}) \). Because \( e \models \forall x. \neg \text{Know}(\text{distance} \neq x) \), there exists \( w' \in e \) such that \( w' \approx_1 z \) and \( w'[\text{distance}, \{ 1 \}] = 9 \). Since \( 9 < 10 \), we also have that \( w' \approx_2 z \). However, \( w'[\text{distance}, z] = 10 \). So there exists \( w' \in e \) such that \( w' \approx_2 z \) and \( w', z \models \neg \text{Close} \). Therefore, \( e, w, z \models \neg \text{Know}(	ext{Close}) \). \( \square \)

4. **Projection by regression**

The examples of the previous section all involve projection as a fundamental reasoning task, that is, determining what holds after a number of actions have occurred, as in

\[ \Sigma \land \text{Know}(\Sigma') \land \forall x. \neg \text{Know}(\text{distance} \neq x) \models [\text{sonar} [\text{backward}] \neg \text{Know} [\text{Close}] \]. \]

When we are not concerned with knowledge, things are somewhat simpler as we only need a single basic action theory as in

\[ \Sigma \models [\text{forward}] [\text{backward}] \neg \text{Close} \].

For this simpler case, Reiter [36] showed how successor state axioms allow the use of regression to solve this reasoning task for certain \( \alpha \) (which he called the regressable formulas) and which, roughly, correspond to bounded objective formulas in \( ES \).

The idea is to successively replace fluents in \( \alpha \) by the right-hand side of their successor state axioms until the resulting sentence contains no more actions, at which point one need only check whether that sentence follows from the sentences in the initial theory. Later Scherl and Levesque [39] extended these results to handle knowledge in the situation calculus. In this section we will show how these ideas carry over to \( ES \), beginning with the non-epistemic fragment of the language.

4.1. **Regression objective formulas**

Here we consider regression to determine entailments of the form \( \Sigma \models \alpha \), where \( \Sigma \) is a basic action theory and \( \alpha \) is any bounded objective sentence. To start with we assume, from now on, that all basic action theories and queries are rectified, that is, that each quantifier has a distinct variable. This is needed for regression to work properly and later for the
translation of $ES$ to the situation calculus. To simplify the formal details, we will define regression only for formulas in the following normal form $NF$.

**Definition 5.** A sentence $\alpha$ is in $NF$ if it is rectified and every function symbol $f$ in $\alpha$ occurs only in equality expressions of the form $(f(n_1, \ldots, n_k) = n)$, where the $n_i$ and $n$ here are either variables or standard names.

It is easy to show that every sentence can be transformed into an equivalent one in $NF$ and the transformation is linear in the size of the original sentence. For example, if $b$ is a rigid constant and $f$ a functional fluent, then the normal form of $F(f(b))$ is $3x,y. (b = x) \land (f(x) = y) \land F(y)$. Note that, for any formula in $NF$, if a term $t$ appears in $[t]$ or as an argument to a function or predicate, then $t$ is either a variable or a standard name. In the following we will make use of sequences which consist of action variables or action standard names. We will reserve the symbol $r$ to denote such sequences. (We continue to use $z$ to denote the special case where all elements of the sequence are standard names.)

In our account, any bounded, objective sentence $\alpha$ in $NF$ is considered regressable. By the transformation above any bounded, objective sentence becomes regressable by first converting it into $NF$ and then applying regression to the result.

**Definition 6.** We define $R[\alpha]$, the regression of $\alpha$ wrt $\Sigma$, to be $R[\langle \cdot \rangle, \alpha]$, where for any sequence $r$ consisting of action variables or standard names, $R[r, \alpha]$ is defined inductively on $\alpha$ by:

1. $R[r, \forall x\alpha] = \forall xR[r, \alpha]$;
2. $R[r, (\alpha \land \beta)] = (R[r, \alpha] \land R[r, \beta])$;
3. $R[r, \neg \alpha] = \neg R[r, \alpha]$;
4. $R[r, [t]\alpha] = R[r \cdot t, \alpha]$;
5. $R[r, \text{Poss}(t)] = R[r, \pi_{G}^{a}]$;
6. $R[r, \text{SF}(t)] = R[r, \phi_{F}^{a}]$;
7. $R[r, G(t_1, \ldots, t_k)] = G(t_1, \ldots, t_k)$ for rigid predicate $G$;
8. $R[r, F(t_1, \ldots, t_k)]$ for fluent predicate $F$ is defined inductively on $r$ by:
   (a) $R[r, F(t_1, \ldots, t_k)] = F(t_1, \ldots, t_k)$;
   (b) $R[r \cdot t, F(t_1, \ldots, t_k)] = R[r, G^{a}_{F}t_1 \ldots t_k]$;
9. $R[r, (t_1 = t_2)] = (t_1 = t_2)$ if $t_1$ and $t_2$ do not mention functional fluents;
10. $R[r, (f(n_1, \ldots, n_k) = n)]$ for functional fluent $f$ is defined inductively by:
   (a) $R[r, (f(n_1, \ldots, n_k) = n)] = (f(n_1, \ldots, n_k) = n)$;
   (b) $R[r \cdot t, (f(n_1, \ldots, n_k) = n)] = \exists y.(y\gamma_{f}^{a}n_1 \ldots n_k \land (y = n))$.

Note that this definition uses the right-hand sides of the precondition, successor state, and sense condition axioms from $\Sigma$.

It is not hard to show that $R$ always transforms a bounded objective formula into a fluent formula.

**Lemma 2.** Let $\alpha$ be a bounded objective formula and $r$ a sequence of action variables or standard names. Then there is a unique fluent formula $\phi$ such that $R[r, \alpha] = \phi$.

**Proof.** The proof is simple but tedious and we will skip the details here. Perhaps the only interesting aspect is the structure of the proof itself, which is also used in other proofs of properties of regression below. First, the lemma is proved for static formulas only. This is achieved by an induction on the length of $r$ and a sub-induction on the length of $\alpha$, counting the number of logical operators and where occurrences of $\text{Poss}(t)$ and $\text{SF}(t)$ are counted as the length of $\pi_{G}^{a} + 1$ and $\phi_{F}^{a} + 1$, respectively. Note, in particular, that the induction is well-behaved because the formulas $\pi$, $\varphi$, $\gamma_{F}$, and $\gamma_{f}$ are themselves fluent formulas, that is, they are static and mention neither $\text{Poss}$ nor $\text{SF}$.

Having proved the lemma for static $\alpha$, the case of bounded formulas is established by another simple induction on the number of $[t]$-operators in $\alpha$. $\square$

Using the semantics of $ES$, we will now reprove Reiter’s Regression Theorem, and show that it is possible to reduce reasoning with formulas that contain $[t]$ operators to reasoning with fluent formulas in the initial state.

We begin by defining for any world $w$ and basic action theory $\Sigma$ another world $w_{\Sigma}$ which is like $w$ except that it satisfies the $\Sigma_{\text{pre}}$, $\Sigma_{\text{sense}}$, and $\Sigma_{\text{post}}$ sentences of $\Sigma$.

**Definition 7.** Let $w$ be a world, $z \in Z$, and $\Sigma$ a basic action theory with fluents $\mathcal{F}$. Then $w_{\Sigma}$ is a world satisfying the following conditions:

---

10 See also the proof of Lemma 6 below, where this is needed to establish the induction for $\forall$. 

1. For h ∈ ℱ (predicate or function), wΣ[h(n₁, . . . , nₖ), z] = w[h(n₁, . . . , nₖ), z];
2. For predicate F ∈ ℱ, wΣ[F(n₁, . . . , nₖ), z] is defined inductively:
   (a) wΣ[F(n₁, . . . , nₖ), ()] = w[F(n₁, . . . , nₖ), ()];
   (b) wΣ[F(n₁, . . . , nₖ), z ⋅ m] = 1 iff wΣ, z = (γ₁y₁v₁) ⋯ (γₙyₙvₙ);
3. For function f ∈ ℱ, wΣ[f(n₁, . . . , nₖ), z] is defined inductively:
   (a) wΣ[f(n₁, . . . , nₖ), ()] = w[f(n₁, . . . , nₖ), ()];
   (b) wΣ[f(n₁, . . . , nₖ), z ⋅ m] = n iff wΣ, z = (γ₁y₁v₁) ⋯ (γₙyₙvₙ);
4. wΣ[Poss(n), z] = 1 iff wΣ, z = π₁z;
5. wΣ[SF(n), z] = 1 iff wΣ, z = ϕ₁z.

Note that this again uses the π, γ, and ϕ formulas from Σ. Then we get the following simple lemmas:\(^1\)

**Lemma 3.** For any w, wΣ exists and is uniquely defined.

**Proof.** wΣ clearly exists. The uniqueness follows from the fact that π and ϕ are fluent formulas and that for all fluents in ℱ, once their initial values are fixed, then the values after any number of actions are uniquely determined by Σ\(_{post}\). □

**Lemma 4.** If w |= Σ\(_{una}\) ∪ Σ then wΣ |= Σ.

**Proof.** Directly from the definition of wΣ, we have that wΣ |= ∀a □ Poss(a) ≡ π, wΣ |= ∀a □ SF(a) ≡ ϕ, wΣ |= ∀a ∀x □ [a]F(x) ≡ γ\(_{f}\), and wΣ |= ∀a ∀x □ y □ [a]f(x) = y ≡ γ\(_{f}\). □

**Lemma 5.** If w |= Σ then wΣ = wΣ.

**Proof.** If w |= ∀a □ Poss(a) ≡ π, w |= ∀a □ SF(a) ≡ ϕ, w |= ∀a ∀x □ [a]F(x) ≡ γ\(_{f}\), and w |= ∀a ∀x □ y □ [a]f(x) = y ≡ γ\(_{f}\), then w satisfies the definition of wΣ. □

The following property of regression is used to prove the main lemma needed for the Regression Theorem. Given a sequence of action variables or standard names r, let r\(_{n}\) denote r with all occurrences of variable x replaced by standard name n.

**Lemma 6.** For any bounded objective formula α and sequence of action variables or standard names r, R[r, α\(_{n}\)] = R[r\(_{n}\), α\(_{n}\)].

**Proof.** The proof is long but simple and follows the structure of Lemma 2. Here we only consider static α and three cases: fluent predicates, assuming that the lemma holds for |r| = k − 1, Poss and ∀, assuming in the sub-induction that the lemma holds for formulas of length m − 1.

1. Let r = r₁ · t. Then R[r, F(t)]\(_{n}\) = R[r₁, γ₁F(t)]\(_{n}\) (def. of R) = R[r₁, (γ₁F(t)]\(_{n}\) (by induction) = R[r₁, (γ₁F(t)]\(_{n}\) (since x not in γ₁) = R[t₁, F(t)]\(_{n}\).
2. R[r, Poss(t)]\(_{n}\) = R[r, π₁t]\(_{n}\) (definition of R) = R[r₁, π₁t]\(_{n}\) (by induction, as π₁t is of length m − 1) = R[r₁, π₁t]\(_{n}\) (since π does not mention x) = R[t₁, Poss(t)]\(_{n}\).
3. R[r, ∀y.α\(_{n}\)] = (∀y. R[r, α])\(_{n}\)) = ∀y. R[r, α\(_{n}\)] (since x ≠ y) = ∀y. R[r, α\(_{n}\)] (by induction on |α|) = R[r, (∀y.α\(_{n}\))]. □

**Lemma 7.** Let α be any bounded, objective sentence in NF and z ∈ Z.

Then w |= R[z, α] iff wΣ, z = α.

**Proof.** As before, the proof is rather straightforward and uses the same induction scheme as Lemma 2. Assuming the lemma holds for z of length k − 1, we only consider two cases, atoms with functional fluents and ∀.

1. Note that, by the definition of NF, ground atoms mentioning functional fluents have the form f(n₁, . . . , nₖ) = n, where n and n₁ are standard names. Then:
   (a) wΣ, z ⋅ m = f(n₁, . . . , nₖ) = n iff (by definition of wΣ);
   (b) wΣ, z = ∃y.(γ₁y)v₁ ⋯ (γₙy)vₙ ∧ y = n iff (by induction);
   (c) wΣ, z = R[z, y]fₘ₁ ⋯ fₙᵣ y = n iff (by definition of R);
   (d) wΣ, z = R[z, f(n₁, . . . , nₖ) = n].
2. w |= R[z, ϕ₁α] iff w |= ∀x. R[z, α] iff w |= R[z, α\(_{n}\)] for all n of the right sort iff (by Lemma 6), w |= R[z, α\(_{n}\)] for all n iff (by sub-induction on |α|), wΣ, z = α\(_{n}\) for all n iff wΣ, z = ∀x.α. □

\(^1\) As we only consider first-order sentences here, the second-order variable map u is dropped everywhere.
Theorem 3 (Objective regression). Let $\Sigma = \Sigma_{una} \cup \Sigma_0 \cup \Sigma_{pre} \cup \Sigma_{post} \cup \Sigma_{sense}$ be a basic action theory and let $\alpha$ be an objective, bounded sentence. Then $R[\alpha]$ is a fluent sentence and satisfies

$$\Sigma \models \alpha \iff \Sigma_{una} \cup \Sigma_0 \models R[\alpha].$$

Proof. Suppose $\Sigma_{una} \cup \Sigma_0 \models R[\alpha]$. We prove that $\Sigma \models \alpha$. Let $w$ be any world such that $w \models \Sigma$. Then, $w \models \Sigma_{una} \cup \Sigma_0$, and so $w \models R[\alpha]$. By Lemma 5, $w_\Sigma = w$, and so $w \models \alpha$.

Conversely, suppose $\Sigma \models \alpha$. We prove that $\Sigma_0 \models R[\alpha]$. Let $w$ be any world such that $w \models \Sigma_{una} \cup \Sigma_0$. From Lemma 4, $w_\Sigma \models \Sigma$, and so $w \models \alpha$. By Lemma 7, $w \models R[\alpha]$.

Note that the conciseness of this proof depends crucially on the fact that Lemma 7 is proved by induction over sentences, which is possible only because quantification is interpreted substitutionally.

4.2. Regressing knowledge

Let us now turn to the more general case of regression for bounded sentences which may refer to the agent’s knowledge. As we discussed in Section 3, this means that we need to consider two basic action theories $\Sigma$ and $\Sigma'$ for what is true in the world and for what the agent believes, respectively.

The following theorem can be thought of as a successor-state axiom for knowledge, which will allow us to extend regression to formulas containing $\text{Know}$. Note that, in contrast to the successor state axioms for fluents, this is a $\text{Theorem}$ of the logic not a stipulation as part of a basic action theory:

Theorem 4. $\Box [\alpha] \text{Know}(\alpha) \equiv$

$$\text{SF}(\alpha) \land \text{Know}(\text{Poss}(\alpha)) \land \text{SF}(\alpha) \supset \Box [\alpha] \lor$$

$$\neg \text{SF}(\alpha) \land \text{Know}(\text{Poss}(\alpha)) \land \neg \text{SF}(\alpha) \supset \Box [\alpha] \alpha.$$

Proof. For both directions of the equivalence we will only consider the case where $\neg \text{SF}(n)$ holds for an arbitrary action name $n$. The other case is completely analogous.

To prove the only-if direction, let $e, w, z \models [n] \text{Know}(\alpha^n_2)$ for action name $n$. We write $\alpha'$ for $\alpha^n_2$. Suppose $e, w, z \models \neg \text{SF}(n)$. It suffices to show that $e, w, z \models \text{Know}(\text{Poss}(n)) \land \neg \text{SF}(n) \supset [n] \alpha'$. So suppose $w' \sim_z w$, $w' \in e$, $w' \text{Poss}(n), z \models 1$, and $w' \text{SF}(n), z \models 0$. Thus $w'[\text{SF}(n), z] = w[\text{SF}(n), z]$ and, hence, $w' \sim_z w$. Since $e, w, z \models [n] \text{Know}(\alpha')$ by assumption, $e, w', z \models \alpha'$, from which $e, w', z \models [n] \alpha'$ follows.

Conversely, let $e, w, z \models \neg \text{SF}(n) \land \text{Know}(\text{Poss}(n)) \land \neg \text{SF}(n) \supset [n] \alpha'$. We need to show that $e, w, z \models [n] \text{Know}(\alpha')$, that is, $e, w, z \models \text{Know}(\alpha')$. Let $w' \sim_{z,n} w$ and $w' \in e$. Then $w'[\text{Poss}(n), z] = 1$ and $w'[\text{SF}(n), z] = w[\text{SF}(n), z] = 0$ by assumption. Hence $e, w', z \models \text{Poss}(n) \land \neg \text{SF}(n)$. Therefore, by assumption, $e, w', z \models \alpha'$, from which $e, w, z \models [n] \text{Know}(\alpha')$ follows.

11. $R[\Sigma', \Sigma, r, \text{Know}(\alpha)]$ is defined inductively on $r$ by:

(a) $R[\Sigma', \Sigma, \langle \rangle, \text{Know}(\alpha)] = \text{Know}(R[\Sigma', \Sigma', \langle \rangle, \alpha]);$

(b) $R[\Sigma', \Sigma, r \cdot t, \text{Know}(\alpha)] = R[\Sigma', \Sigma, r, \beta^n_2]$, where $\beta$ is the right-hand side of the equivalence in Theorem 4.

---

12 Note that by this account, after performing an impossible action, the agent believes that the action was possible. This is perhaps undesirable, but this anomaly arises only in non-legal situations.

13 This is like [19] but in contrast to Scherl and Levesque [39], who can only handle true belief. While we allow for false beliefs, we continue to use the terms knowledge and belief interchangeably.
For simplicity, we write $R[\alpha]$ instead of $R[\Sigma', \Sigma, (\cdot), \alpha]$. Next we extend Lemma 4 to knowledge, where $e_\Sigma = \{ w_\Sigma \mid w \in e \}$ for a given epistemic state $e$ and basic action theory $\Sigma'$:

**Lemma 8.** If $e \models \text{Know}(\Sigma_{\text{una}} \cup \Sigma_0)$ then $e_\Sigma \models \text{Know}(\Sigma)$.

**Proof.** Let $e \models \text{Know}(\Sigma_{\text{una}} \cup \Sigma_0)$, that is, for all $w$, if $w \in e$ then $w \models \Sigma_{\text{una}} \cup \Sigma_0$. We need to show that for all $w$, if $w \in e_\Sigma$ then $w \models \Sigma$.

Let $w \in e_\Sigma$. By definition, there is a $w' \in e$ such that $w = w'_\Sigma$. Since $w' \models \Sigma_{\text{una}} \cup \Sigma_0$, by Lemma 4, $w'_\Sigma \models \Sigma$, that is, $w \models \Sigma$. □

We now turn to the generalization of Lemma 7 for knowledge.

**Lemma 9.** $e, w \models R[\Sigma', \Sigma, z, \alpha]$ iff $e_{\Sigma'}, w_{\Sigma}, z \models \alpha$.

**Proof.** The proof is by induction on $z$ with a sub-induction on $\alpha$.

Let $z = \langle \cdot \rangle$. The proof for $\text{Poss}$, $SF$, fluent atoms, and the connectives $\neg$, $\land$, and $\lor$ is exactly analogous to Lemma 7. For formulas $\text{Know}(\alpha)$ we have:

- $e_{\Sigma'} \models \text{Know}(\alpha)$ iff
  - for all $w \in e_{\Sigma'}$, $e_{\Sigma'}$, $w \models \alpha$ iff (by definition of $e_{\Sigma'}$),
  - for all $w \in e$, $e_{\Sigma'}$, $w_{\Sigma'} \models \alpha$ iff (by induction),
  - for all $w \in e$, $w, \models R[\Sigma', \Sigma, (\cdot), \alpha]$ iff
    - $e \models \text{Know}(R[\Sigma', \Sigma, (\cdot), \alpha]$ (by definition of $R$),
    - $e \models R[\Sigma', \Sigma, (\cdot), \text{Know}(\alpha)]$.

This concludes the base case $z = \langle \cdot \rangle$.

Now consider the case of $z \cdot n$, which again is proved by a sub-induction on $\alpha$. The proof is exactly like the sub-induction for the base case except for $\text{Know}$, for which we have the following:

- $e_{\Sigma'}$, $w_{\Sigma'}$, $z \cdot n \models \text{Know}(\alpha)$ iff (by Theorem 4),
- $e_{\Sigma'}$, $w_{\Sigma'}$, $z \models \beta^\alpha_0$ (where the $\beta$ is from Theorem 4)
  - iff (by (main induction)),
- $e, w \models R[\Sigma', \Sigma, z, \beta^\alpha_0]$ iff (by definition of $R$),
- $e, w \models R[\Sigma', \Sigma, z \cdot n, \text{Know}(\alpha)]$,

which completes the proof. □

Finally, here is the general regression theorem:

**Theorem 5** (Generalized regression). Let $\Sigma$ and $\Sigma'$ be basic action theories, and $\alpha$ be a bounded sentence. Then $R[\alpha]$ is a static sentence and satisfies

$$\Sigma \land \text{Know}(\Sigma') \models \alpha \iff \Sigma_{\text{una}} \cup \Sigma_0 \land \text{Know}(\Sigma_{\text{una}} \cup \Sigma'_0) \models R[\alpha].$$

**Proof.** To prove the only-if direction, let us suppose that $\Sigma \land \text{Know}(\Sigma') \models \alpha$ and that $e, w \models \Sigma_{\text{una}} \cup \Sigma_0 \land \text{Know}(\Sigma_{\text{una}} \cup \Sigma'_0)$. Thus $w \models \Sigma_{\text{una}} \cup \Sigma_0$ and, by Lemma 4, $w_\Sigma \models \Sigma$. Also, $e \models \text{Know}(\Sigma_{\text{una}} \cup \Sigma'_0)$ and thus, by Lemma 8, $e_{\Sigma'} \models \text{Know}(\Sigma')$. (Note that $\Sigma_{\text{una}}$ is the same for both $\Sigma$ and $\Sigma'$. Therefore, $e_{\Sigma'}$, $w_{\Sigma} \models \Sigma \land \text{Know}(\Sigma')$. By assumption, $e_{\Sigma'}$, $w_{\Sigma} \models \alpha$ and, by Lemma 9, $e, w \models R[\alpha]$.

Conversely, suppose $\Sigma_{\text{una}} \cup \Sigma_0 \land \text{Know}(\Sigma_{\text{una}} \cup \Sigma'_0) \models R[\alpha]$ and let $e, w \models \Sigma \land \text{Know}(\Sigma')$. Then $w \models \Sigma_{\text{una}} \cup \Sigma_0$ and $e \models \text{Know}(\Sigma_{\text{una}} \cup \Sigma'_0)$. Then, by assumption, $e, w \models R[\alpha]$. Then $e_{\Sigma'}$, $w_{\Sigma} \models \alpha$ by Lemma 9. By Lemma 5, $w_{\Sigma} = w$ and, since $e \models \text{Know}(\Sigma')$, $e_{\Sigma'} = e$. Therefore, $e, w \models \alpha$. □

This theorem shows that determining what is true and what is known after any (bounded) number of actions has occurred can always be reduced to reasoning about what is true and known in the initial state.

In order to deal with our robot example, we need to go a little beyond this as we may want to make assumptions about what the robot does not know as in

$$\Sigma \land \text{Know}(\Sigma') \land \forall x. \neg \text{Know}(\text{distance} \neq x) \models [\text{sonar}][\text{backward}] \neg \text{Know}(\text{Close}).$$

The following corollary shows that we can deal with such cases without any problems.
Corollary 1. Let \( \gamma \) be a static sentence which mentions neither Poss nor SF. Then
\[
\Sigma \land \text{Know}(\Sigma') \land \gamma \models \alpha \text{ iff } \Sigma_{\text{una}} \cup \Sigma_0 \land \text{Know}(\Sigma_{\text{una}} \cup \Sigma_0') \land \gamma \models \mathcal{R}[\alpha].
\]

Proof. The proof makes use of the fact that for any static formula \( \gamma \), \( \mathcal{R}[\gamma] = \gamma \cdot \Sigma \land \text{Know}(\Sigma') \models \gamma \rightarrow \alpha \) iff \( \Sigma \land \text{Know}(\Sigma') \models \gamma \rightarrow R[\alpha] \) (by the regression theorem) iff \( \Sigma \land \text{Know}(\Sigma') \models \gamma \supset R[\alpha] \) (because \( R \) leaves \( \gamma \) as is) iff \( \Sigma \land \text{Know}(\Sigma') \models \gamma \supset R[\alpha] \).

5. Mapping to the situation calculus

How do we know that the semantics of \( \mathcal{E}S \) is correct? In this section, we argue that it is indeed correct by showing how formulas \( \alpha \) in \( \mathcal{E}S \) can be translated in a direct way to formulas \( \alpha^* \) in the situation calculus as defined by Reiter. We assume that this language has functional and relational fluents, functions and predicates that are not fluents, the distinguished constant \( S_0 \), function do, binary predicate \( \sqsubseteq \) over situations, predicates Poss and SF, and a binary predicate \( K \) for knowledge.

We take \( \text{Know}(\alpha, \sigma) \) in the situation calculus as an abbreviation for the formula \( \forall s(K(s, \sigma) \supset \alpha^\text{now}_{\sigma}) \), where \( \alpha^\text{now}_{\sigma} \) is the result of replacing by \( s \) in \( \alpha \) every occurrence of now that is not within the scope of a further Knows.14

Perhaps the most desirable and simplest outcome of a translation from \( \mathcal{E}S \) to the situation calculus would be that
\[
\models \alpha \text{ iff } \Sigma \models_{\text{FOL}} \alpha^*.
\]
where \( \models \) is validity in \( \mathcal{E}S \), \( \Sigma \) is the set of foundational axioms of the situation calculus, and \( \models_{\text{FOL}} \) is ordinary logical consequence. Unfortunately, we do not get exactly this correspondence for a variety of reasons we will discuss below. But we do get something close:
\[
\models \alpha \text{ iff } \Sigma \cup \gamma \models_{\text{FOL}} \alpha^*,
\]
where \( \gamma \) is a set of five axioms that we will justify separately.

Somewhat surprisingly, it turns out that the foundational axioms \( \Sigma \) are actually completely irrelevant as far as the fragment of the situation calculus as defined by our translation is concerned. In other words, we also obtain
\[
\models \alpha \text{ iff } \gamma \models_{\text{FOL}} \alpha^*.
\]

Below, we will first prove this result and then show that only small modifications are needed to obtain a proof for the case when foundational axioms are assumed as well. To establish these results it will be necessary to work with ordinary Tarski models of sentences of the situation calculus. As we argued in the beginning, this is difficult and painstaking, and is indeed one of the main reasons to prefer \( \mathcal{E}S \) over the situation calculus. But here there is no alternative. So while the proof of the theorems is quite laborious, we remind the reader that this can be thought of as a final reckoning for a formalism that is unworkable semantically.

5.1. The translation

Before describing \( \gamma \), we present the translation from \( \mathcal{E}S \) into the situation calculus. In the simplest case, the idea is that a formula like \( \text{distance} = 6 \), where distance is a fluent, will be mapped to the situation calculus formula \( \text{distance}(S_0) = 6 \), where we have restored the distinguished situation term \( S_0 \) for the fluent. Similarly, the formula [forward]-(distance = 6) will be mapped to \( \neg\text{distance}(\text{do}([\text{forward}], S_0)) = 6 \), and \( \square(\text{distance} > 0) \) will be mapped to \( \forall s'((S_0 \subseteq s' \supset \text{distance}(s') > 0)) \). For knowledge, \( \text{Know}(\text{distance} > 0) \) will be mapped to \( \text{Know}(\text{distance}(\text{now}) > 0, S_0) \) which is an abbreviation for \( \forall s'((K(s', S_0) \supset \text{distance}(s') > 0)) \). So \( \mathcal{E}S \) formulas can be thought of as “situation-suppressed” (in situation-calculus terminology) and the * mapping we will define restores the situation argument to the fluents, leaving the rigidgs unchanged.

More precisely, we have the following:

Definition 8. Let \( \alpha \) be any term or formula of \( \mathcal{E}S \) without standard names. The expression \( \alpha^* \) is defined as \( \alpha[S_0] \) where, for any situation term \( \sigma \), \( \alpha[S_0] \) 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, predicate, or second-order variable, is \( g(t_1[\sigma], \ldots, t_k[\sigma]) \);
3. \( f(t_1, \ldots, t_k)[\sigma] \), where \( f \) is a fluent function, predicate, or second-order variable is \( f(t_1[\sigma], \ldots, t_k[\sigma], \sigma) \);
4. \( (t_1 = t_2)[\sigma] \) is \( (t_1[\sigma] = t_2[\sigma]) \);
5. \( ([t] \alpha)[\sigma] \) is \( \alpha[\text{do}(t[\sigma], \sigma)] \);
6. \( (\alpha \land \beta)[\sigma] \) is \( (\alpha[\sigma] \land \beta[\sigma]) \);

14 In some versions of the situation calculus, the argument to Knows is a formula \( \alpha \) where situations are suppressed. In that case, \( \alpha^\text{now}_{\sigma} \) should be understood as restoring \( s \) as the situation argument.
Note that the translation of □α 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 \( Y \) are the following:

1. Domain of objects is countably infinite\(^{15}\);
2. Domain of actions is countably infinite (as above);
3. Equality is the identity relation\(^{16}\):
   \[
   \forall x \forall y. (x = y) \equiv \forall Q \left( Q(x) \equiv Q(y) \right);
   \]
4. Less-than over situations: \( \forall s \forall s′. (s \subseteq s′) \equiv \forall P(\cdots \supset P(s, s′)) \),
   where the ellipsis stands for the universal closure of
   \[
   \left[ P(s_1, s_1) \right] \land \left[ P(s_2, s_3) \supset P(s_2, do(a, s_3)) \right];
   \]
5. The \( K \) predicate: \( \forall s \forall s′. K(s′, s) \equiv \forall P(\cdots \supset P(s′, s)) \),
   where the ellipsis stands for the universal closure of
   \[
   \left[ K(s_1, S_0) \supset P(s_1, S_0) \right] \land \left[ P(s_1, s_3) \land P(s_2, s_3) \supset P(s_1, s_2) \right] \land \left[ P(s_1, s_2) \land Poss(a, s_1) \land SF(a, s_1) \equiv SF(a, s_2) \supset P(do(a, s_1), do(a, s_2)) \right].
   \]

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.

Axiom (4) uses second-order logic to define the \( \sqsubseteq \) relation as reachability using \( do \). It does not say anything about \( S_0 \) nor about situations that cannot be reached using \( do \). As it turns out, we do not need to stipulate anything about them. (See the conclusion for more on this.)

Finally axiom (5) 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 [39], and the added machinery to make \( \text{Knows} \) be a weak-S5 operator [14]. 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:

\[
\exists Q. \forall x.Q(x) \land Inf(Q) \land Cnt(Q)
\]

where

\[
\begin{align*}
Cnt(Q) & \overset{\text{def}}{=} \forall Q′. (Q′ \subseteq Q \land Inf(Q′) \supset Q ≤ Q′) \\
Inf(Q) & \overset{\text{def}}{=} \exists Q′. Q′ \subseteq Q \land Q ≤ Q′
\end{align*}
\]

\(^{15}\) See the precise logical rendering of this below.

\(^{16}\) It is not hard to show that this second-order definition entails all the usual properties like reflexivity and substitution of equals for equals.
Let $\alpha$ be any sentence of $\mathcal{ES}$ without standard names. Then

\[ \alpha \text{ is valid} \iff \mathcal{T} \models_{\text{FOL}} \alpha^*. \]

The long and arduous proof of the theorem is left to Appendix A. If nothing else, it provides further evidence that ordinary Tarski models of sentences of the situation calculus are cumbersome to work with.

An interesting aspect of the theorem is that it requires few assumptions about the nature of situations. For example, $\mathcal{T}$ admits models where there are situations other than those reachable from $S_0$ or the situations which are $K$-accessible from $S_0$. In contrast, Reiter’s foundational axioms $\Sigma$, which we will present in a moment, rule out such non-standard models. In order to show that $\mathcal{ES}$ can be fully embedded in Reiter’s situation calculus, which requires the foundational axioms, we still need to show that the above theorem continues to hold if we add $\Sigma$ as additional assumptions on the right-hand side of the theorem. Note that this is not immediately obvious as extra assumptions normally lead to more entailments.

To see why this is not the case here, let us first review the axioms $\Sigma$ for the epistemic situation calculus from [36]:

1. $\text{do}(a_1, s_1) \equiv \text{do}(a_2, s_2) \supset a_1 = a_2 \land s_1 = s_2$;
2. $\forall Q. \forall s. (\text{Ini}(s) \supset P(s)) \supset \forall a. s[Q(s) \supset Q(\text{do}(a, s))] \supset \forall s. Q(s)$, where $\text{Ini}(s) \equiv \forall a \forall s'. s \neq \text{do}(a, s')$;
3. $s \sqsubseteq \text{do}(a, s') \equiv s \sqsubseteq s'$;
4. $\neg s \sqsubseteq S_0$;
5. $K(s', s) \supset (\text{Ini}(s) \equiv \text{Ini}(s'))$.

(1) is a unique names axiom for situations; the second-order axiom (2) defines the set of all situations to be those reachable from an initial situation (Ini(s)) by a sequence of actions; (3) defines $\sqsubseteq$ as reachability (by a sequence of actions) between situations; (4) says that no situation precedes $S_0$; (5) says that an initial situation is $K$-accessible only from another initial situation. Note that Ini($S_0$) is a logical consequence of $\Sigma$.

The intuitive reason why we can add the axioms to Theorem 6 without invalidating it is because they assert properties of situations which we cannot even express in $\mathcal{ES}$, and hence they are not in the image of the translation. For example, since $S_0$ is not part of the language, we simply cannot say that nothing precedes $S_0$. In other words, while we certainly have $\Sigma \cup \mathcal{T} \models_{\text{FOL}} \forall s. \neg s \sqsubseteq S_0$ and $\mathcal{T} \models_{\text{FOL}} \forall s. \neg s \sqsubseteq S_0$, this does not matter for the theorem, as there is no $\alpha \in \mathcal{ES}$ such that $\alpha^*$ is equivalent to $\forall s. \neg s \sqsubseteq S_0$. Hence we obtain:

Theorem 7. Let $\alpha$ be any sentence of $\mathcal{ES}$ without standard names. Then

\[ \alpha \text{ is valid} \iff \Sigma \cup \mathcal{T} \models_{\text{FOL}} \alpha^*. \]

The theorem establishes that $\mathcal{ES}$ is indeed a fragment of Reiter’s situation calculus. As $\mathcal{ES}$ itself seems to be much more workable, this then is perhaps the main significance of the theorem: any property which we obtain for $\mathcal{ES}$ automatically holds for the fragment of the original situation calculus given by our translation. And this fragment covers the main uses of the situation calculus, in particular, Reiter’s basic action theories [36], and those dealing with knowledge [39]. We saw an example in Section 3. In [24] we also showed that $\mathcal{ES}$ is expressive enough to capture the action language Golog [27].

What do we give up? As we have seen, the foundational axioms themselves cannot be expressed in $\mathcal{ES}$. Another sentence which has no straightforward counterpart in is

\[ \exists s \exists s'. S_0 \sqsubset s \land S_0 \sqsubset s' \land (s \neq s') \land F(s) \equiv F(s'), \]

which says that two distinct situations are reachable that agree on the truth value of $F$. It’s the equality between situations that presents a problem here. However, as we showed in [24], sentences like these 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 $\forall s.\sigma \sqsubseteq s \sqsubseteq \alpha$, where $\sigma$ is a situation term. Note, in particular, that the rooted situation calculus without knowledge is equivalent to Reiter’s original version (simply take $\sigma$ to be $S_0$). So, with a little bit of extra effort, $E\Sigma$ has almost the same expressive power as all of the situation calculus.

6. Related work

While the situation calculus has received a lot of attention in the reasoning about action community, there are, of course, a number of alternative formalisms, including close relatives like the fluent calculus [13,40] and more distant cousins described in [16,9,38].

The closest approaches to ours are perhaps those concerned with reasoning about action based on dynamic logic [10]. For example, De Giacomo and Lenzerini [4] and later Demolombe et al. [6], whose work extends [3], show how to import Reiter’s solution to the frame problem into dynamic logic. There are also epistemic extensions of dynamic logic such as [11], which, like $E\Sigma$, is inspired by the desire to capture fragments of the situation calculus in modal logic.

Although they do not consider epistemic notions, the work by [1] is relevant as it reconstructs a version of the situation calculus in Hybrid Logic [1], a variant of modal logic which was inspired by the work on tense logic by Prior [34]. In a sense, though, this work goes only part of the way as an explicit reference to situations within the logic is retained. To us this presents a disadvantage when moving to an epistemic extension. As we said in the beginning, the problem is that the epistemic situation calculus requires us to consider uncountably many situations, which precludes a substitutional interpretation of quantification.

7. Only knowing and actions

In previous work, for example [21,22], we have used the concept of only-knowing as a way of capturing the idea that a sentence is not only believed by an agent, but all that is believed by the agent. This idea has a number of applications, including reconstructing some aspects of nonmonotonic reasoning [25].

One application of only-knowing for our purposes is to provide a convenient way of specifying what is not known: instead of saying that an agent believes some sentences (such as a basic action theory) and also stipulating that she does not believe certain other sentences (as we did in the robot example in Section 3), it will be sufficient to say that the basic action theory is all that the agent believes. It will then follow logically that certain other sentences are not believed. So for example, in the robot example, instead of saying

The following are logical entailments of

$$\Sigma \land \text{Know}(\Sigma') \land \forall x.\neg\text{Know(distance} \not= x),$$

where we had to say explicitly that the agent did not know anything about the distance to the wall, it will be sufficient to say

The following are logical entailments of

$$(\Sigma \land \text{OKnow}(\Sigma')).$$

where $\text{OKnow}(\Sigma')$ will be how we say in the language that $\Sigma'$ is all that is known. Once this operator is properly defined, we will then obtain as a property that

$$\models \text{OKnow}(\Sigma') \supset \forall x.\neg\text{Know(distance} \not= x),$$

which then allows us to carry out the example. In fact, for any two fluent sentences $\phi$ and $\psi$ such that $\models (\phi \supset \psi)$, we will have that $\models (\text{OKnow}(\phi) \supset \neg\text{Know}(\psi)).$

But how should this operator be generalized from our previous work on only-knowing in the static case? In an earlier version of this paper [23], we proposed a definition that had some interesting properties, but some drawbacks as well. We now believe that the definition should be the following:

- $e, w, z, u \models \text{OKnow(}\alpha) \text{ iff for every } w', w' \in e^w z \text{ iff } e^w z'.(,). u \models \alpha.$

where

$\varphi^w_z$ is defined as $\{ w'_z \mid w' \in e \text{ and } w' \approx z \text{ w} \}.$ and $w_z$ is defined by $w_z[H(n), z'] = w[H(n), z \cdot z'].$
This semantic characterization appears to be the correct generalization of only-knowing for a dynamic language like \( ES \). In addition, it appears to have some very nice connections to the concept of the progression of basic action theories [20,28,41]. One complication here is that rigid predicates (predicates whose truth values are unchanging and known) appear to present difficulties for \( ES \). We showed it to have a relatively simple model theoretic semantics based on possible worlds. We showed that this semantics allowed clear and direct proofs of certain properties of interest. We also showed that it coincided with the use of the situation calculus by Reiter.

8. Conclusions

In this paper, we have isolated a fragment of the situation calculus with knowledge (presented using a modal syntax) and showed it to have a relatively simple model theoretic semantics based on possible worlds. We showed that this semantics allowed clear and direct proofs of certain properties of interest. We also showed that it coincided with the use of the situation calculus by Reiter provided we made five assumptions: there are countably infinitely many actions and objects, the = symbol denotes identity, the \( \sqsubseteq \) symbol denotes reachability using do, and knowledge satisfies the properties of a quantified weak S5. Given these assumptions, practitioners are then free to use either the modal syntax we have presented here or the original situation calculus syntax introduced by McCarthy and can expect to reap the same semantic rewards.

The subset of the situation calculus studied here might be thought to owe more to the dialect studied by Ray Reiter and coworkers [36] than to the less constrained dialect first envisioned by John McCarthy [29]. For example, we lean heavily on the use of basic action theories and regression, as popularized by Reiter. However, this is not quite right. A careful reading of the embedding theorem, which shows how a fragment of the classical situation calculus can be realized in \( ES \), demonstrates quite clearly that we get an exact correspondence without requiring all the foundational axioms proposed by Reiter. These axioms (including the second-order induction axiom) were used by Reiter to constrain the space of all possible interpretations on the use of basic action theories and regression, as popularized by Reiter. However, this is not quite right. A careful reading of the embedding theorem, which shows how a fragment of the classical situation calculus can be realized in \( ES \), demonstrates quite clearly that we get an exact correspondence without requiring all the foundational axioms proposed by Reiter.

More notation: For any term \( t \), \( \| t \|_\mu^M \) is the element of \( D \) denoted by \( t \) in the classical sense, and \( M, \mu \models \alpha \) means that \( \alpha \) is true in the classical sense for structure \( M \), when the free variables are interpreted by \( \mu \). We will omit the \( \mu \) when nothing depends on the variable map.

Let \( M \) be a Tarski structure satisfying the axioms in \( \gamma \) over the domain \( D = D_{\text{sit}} \cup D_{\text{act}} \cup D_{\text{obj}} \). Let \( t_0 = S_0^M \) and \( D_{\text{act}} = \{ t_0 \} \cup \{ d \in D_{\text{sit}} \mid M, \mu(s/d) \models K(s, S_0) \} \). \( D_{\text{sit}} \) should be thought of as starting situations consisting of \( S_0 \) and whatever is \( K \)-accessible from \( S_0 \). These play the role of the initial situations in Reiter’s situation calculus except that they may have predecessors because we do not require Reiter’s foundational axioms. Let \( e \) be an epistemic state, and assume that we are given three mappings \( \omega \in D_{\text{act}} \to W \), \( \theta \in \mathcal{N} \to D_{\text{act}} \cup D_{\text{obj}} \), and \( \pi \in [Z \times D_{\text{act}} \to D_{\text{act}}] \). Suppose further that the following properties are satisfied:

1. \( \theta \) is 1-1, onto, and sort preserving;
2. \( \omega \) is onto \( e \) (that is, \( e \subseteq \) the image of \( \omega \));
3. for any rigid function symbol \( g \),
   \[ g^M(\theta(n_1), \ldots, \theta(n_k)) = \theta(\omega(\tau)[g(n_1, \ldots, n_k), (\text{\square})]) \]
4. for any fluent function symbol \( f \),
   \[ f^M(\theta(n_1), \ldots, \theta(n_k), \pi(z, t)) = \theta(\omega(\tau)[f(n_1, \ldots, n_k), z]) \]
5. for any rigid predicate symbol \( G \),
   \[ G^M = \{ \theta(n_1), \ldots, \theta(n_k) \mid \omega(\tau)[G(n_1, \ldots, n_k), (\text{\square})] = 1 \} \]
6. For any fluent predicate symbol $F$ (including Poss and SF),
$$F^M = \{ [\theta(n_1), \ldots, \theta(n_k), \pi(z, t)] \mid \omega(i)[F(n_1, \ldots, n_k), z] = 1 \}$$

7. $do^M(\theta(n), \pi(z, t)) = \pi(z \cdot n, t)$.
8. $=^M$ is identity.
9. $\subseteq^M = \{ (\pi(z, t), \pi(z', t')) \}$.
10. $K^M = \{ (\pi(z, t'), \pi(z, t)) \mid \omega(i') \in e$ and $\omega(i') \simeq \omega(i) \}$.

In what follows, we will consider expressions (terms or formulas) of $ES$ whose free first-order variables appear in the list $x_1, \ldots, x_m$. We will consider substituting these variables by standard names $n_1, \ldots, n_m$ of the right sort. For ease of reading, for any term or formula $\alpha$, we write $\alpha^+$ to mean $\alpha[x_1, \ldots, x_m]$. In the situation calculus, for a variable map $\mu$, we write $\mu^+$ to mean $\mu[x_1/\theta(n_1), \ldots, x_m/\theta(n_m), s/\pi(z, t)]$, where the variable $s$, the $z \in Z$ and the $t \in D_{str}$ will be determined by context. Then we get the following:

**Lemma 10.** Let $t$ be a term of $ES$ without standard names whose free variables are among the $x_1, \ldots, x_m$. Let $t \in D_{str}$, $z \in Z$, and $w = \omega(i)$. Then, given the properties above,
$$\|t[s]\|_{\mu^+} = \theta([t^+]^w_z).$$

**Proof.** By induction on $t$.
If $t$ is the variable $x_i$, then $\|t[s]\|_{\mu^+} = \|x_i\|_{\mu^+} = \theta(n_i) = \theta([t^+]^w_z)$.
If $t$ is of the form $g(t_1, \ldots, t_k)$ then we have the following:
$$\|t[s]\|_{\mu^+} = \theta([t^+]^w_z).$$
If $t$ is of the form $f(t_1, \ldots, t_k)$ then we have the following:
$$\|t[s]\|_{\mu^+} = \theta([t^+]^w_z).$$

**Lemma 11.** Let $t$ be a term of $ES$ without standard names whose free variables are among the $x_1, \ldots, x_m$. Let $i \in D_{str}$, $z \in Z$, and $w = \omega(i)$. Then, given the properties above,
$$d\omega(t[s], s) = \pi(z \cdot t^+|^w_z, t).$$

**Proof.** The proof is immediate from Property 7 and Lemma 10.

**Definition 9.** For a given variable map $\mu$ let $u_\mu$ be any $ES$ variable map which satisfies the following:
1. For any rigid second-order variable $V$ and any $w$ and $z$, let
$$u_\mu[w, V(n_1, \ldots, n_k), z] = 1 \iff [\theta(n_1), \ldots, \theta(n_k)] \in \mu(V);$$
2. for any fluent second-order variable \( V \) and any \( i \) and \( z \), let
\[
\mu_{ij}[\omega(i), V(n_1, \ldots, n_k), z] = 1 \iff (\theta(n_1), \ldots, \theta(n_k), \pi(z, i)) \in \mu(V).
\]

Note that \( \mu_{ij} \) is not necessarily completely specified for fluent variables as there may be worlds in \( W \) which are not in the image of \( \omega \). Let us call a world which is in the image of \( \omega \) an \( \omega \)-world. For example, all worlds in \( e \) are \( \omega \)-worlds because \( \omega \) is assumed to be onto \( e \) (Property 2). Then we have the following:

**Lemma 12.** Let \( w = \omega(i) \) and let \( u_1 \) and \( u_2 \) be variable maps which agree on all values where the first argument is an \( \omega \)-world. Then for all basic sentences \( \alpha \),
\[
e, w, z, u_1 \models \alpha \iff e, w, z, u_2 \models \alpha.
\]

**Proof.** The proof is by induction on the structure of \( \alpha \). All cases except second-order variables are immediate since \( w \) and all worlds in \( e \) are \( \omega \)-worlds.

Let \( e, w, z, u_1 \models \forall V. \alpha \). Then for all \( u' \), \( u' \sim_V u_1 \) then \( e, w, z, u' \models \alpha \). Now consider any \( u'' \sim_V u_2 \). Then there clearly is a \( u' \sim_V u_1 \) which agrees with \( u'' \) on \( V \). Also, by definition, this \( u' \) agrees with \( u_1 \) on all other values and hence agrees with \( u_2 \) and therefore \( u'' \) on all values for \( \omega \)-worlds. Thus, by induction, \( e, w, z, u'' \models \alpha \). Since this holds for all \( u'' \sim_V u_2 \), we obtain \( e, w, z, u_2 \models \forall V. \alpha \). The other direction is completely symmetric. \( \square \)

**Lemma 13.** \( e, w, z, u_\mu \models \forall V. \alpha \iff \text{for all relations \( D \) with arity and sorts given by} \ V \), \( e, w, z, u_\mu[V/Z] \models \alpha \).

**Proof.** We only consider fluent variables \( V \). (Rigids are a simpler special case.) For the only-if direction, suppose \( u_\mu[V/Z] \) is given. Then there is a \( u' \sim_V u_\mu \) such that \( u' \) agrees with \( u_\mu[V/Z] \) on \( V \) and, by definition of \( \sim_V \), also agrees with \( u_\mu[V/Z] \) on all variables other than \( V \) on all \( \omega \)-worlds. Hence \( u_\mu[V/Z] \) agrees with \( u' \) on all \( \omega \)-worlds and thus, by assumption and Lemma 12, \( e, w, z, u_\mu[V/Z] \models \alpha \).

Conversely, let \( u' \sim_V u_\mu \). Consider
\[
Z = \{ (\theta(n_1), \ldots, \theta(n_k), \pi(z, i)) \mid u'(\omega(i), V(n_1, \ldots, n_k), z) = 1 \text{ for all } i \text{ and } z \}.
\]
Then \( u' \) agrees with \( u_\mu[V/Z] \) on all \( \omega \)-worlds and hence \( e, w, z, u' \models \alpha \) by assumption and Lemma 12. Since this holds for any \( u' \sim_V u_\mu \), we obtain \( e, w, z, u_\mu \models \forall V. \alpha \). \( \square \)

Now we can put all the results together and prove the main lemma:

**Lemma 14.** Let \( \alpha \) be a basic formula of \( \mathcal{ES} \) with no standard names and whose free variables are among the \( x_1, \ldots, x_m \). Then, given the properties above, for any variable map \( \mu \), any \( \mathcal{ES} \) variable map \( u_\mu \) satisfying Definition 9, any situation variable \( s \), any \( z \in Z \), any \( i \in \mathcal{D}_{stat} \) and \( w = \omega(i) \),
\[
e, w, z, u_\mu \models \alpha^+ \iff \mathcal{M}, \mu^+ \models \alpha[s].
\]

**Proof.** The proof is by induction on the structure of \( \alpha \). There are 12 cases:

1. For a formula of the form \( F(t_1, \ldots, t_k) \):
\[
e, w, z, u_\mu \models F(t_1, \ldots, t_k)^+ \iff \text{(by definition of satisfaction)}
\]
\[
w[F\{t_1^i | w, \ldots, t_k^i | w\}, z] = 1 \iff \text{(by Property 6)}
\]
\[
\theta(t_1^i | w), \ldots, \theta(t_k^i | w), \pi(z, i) \in F^{\mathcal{M}} \iff \text{(by Lemma 10)}
\]
\[
\langle t_1[s]\rangle_{\mu^+} + \ldots + \langle t_k[s]\rangle_{\mu^+}, \pi(z, i) \in F^{\mathcal{M}} \iff \text{(by definition of satisfaction)}
\]
\[
\mathcal{M}, \mu^+ \models F(t_1[s], \ldots, t_k[s], s).
\]

2. For a formula of the form \( G(t_1, \ldots, t_k) \):
\[
e, w, z, u_\mu \models G(t_1, \ldots, t_k)^+ \iff \text{(by definition of satisfaction)}
\]
\[
w[G\{t_1^i | w, \ldots, t_k^i | w\}, z] = 1 \iff \text{(by the rigidity constraint)}
\]
\[
w[G\{t_1^i | w, \ldots, t_k^i | w\}, \{\} ] = 1 \iff \text{(by Property 5)}
\]
\[
\theta(t_1^i | w), \ldots, \theta(t_k^i | w) \in G^{\mathcal{M}} \iff \text{(by Lemma 10)}
\]
\[
\langle t_1[s]\rangle_{\mu^+} + \ldots + \langle t_k[s]\rangle_{\mu^+} \in G^{\mathcal{M}} \iff \text{(by definition of satisfaction)}
\]
\[
\mathcal{M}, \mu^+ \models G(t_1[s], \ldots, t_k[s], s).
\]

3. For a formula of the form \( P(t_1, \ldots, t_k) \):
\[
e, w, z, u_\mu \models P(t_1, \ldots, t_k)^+ \iff \text{(by definition of satisfaction)}
\]
\[
u_\mu[w, P\{t_1^i | w, \ldots, t_k^i | w\}, z] = 1 \iff \text{(by definition of } u_\mu\text{)}
\]
\[
\theta(t_1^i | w), \ldots, \theta(t_k^i | w), \pi(z, i) \in \mu^+[P] \iff \text{(by Lemma 10)}
\]
12. For a formula of the form \( \langle t_1[s]\rangle_{\mu^+}, \ldots, \langle t_k[s]\rangle_{\mu^+}, \pi(z, i) \in \mu^+[P] \) iff (by definition of satisfaction)
\( \mathcal{M}, \mu^+ \models P(t_1[s], \ldots, t_k[s], s) \).

4. For a formula of the form \( \forall w \cdot M, \mu \models Q(t_1, \ldots, t_k) \) iff (by definition of satisfaction)
\( u_\mu[w, Q(\langle t_1[w]\rangle_{\mu^+}, \ldots, \langle t_k[w]\rangle_{\mu^+}, z) = 1 \) iff (by definition of \( u_\mu \))
\( \theta(\langle t_1[w]\rangle_{\mu^+}, \ldots, \langle t_k[w]\rangle_{\mu^+}, \pi(z, i)) \in \mu^+[Q] \) iff (by Lemma 10)
\( \langle \langle t_1[s]\rangle_{\mu^+}, \ldots, \langle t_k[s]\rangle_{\mu^+}, \pi(z, i) \rangle \in \mu^+[Q] \) iff (by definition of satisfaction)
\( \mathcal{M}, \mu^+ \models Q(t_1[s], \ldots, t_k[s], s) \).

5. For a formula of the form \( \forall \tau \cdot M, \mu \models (t_1 = t_2) \) iff (by definition of satisfaction)
\( \forall \tau \cdot \theta(\langle t_1[w]\rangle_{\mu^+}, \langle t_2[w]\rangle_{\mu^+}, \pi(z, i)) \in \mu^+[Q] \) iff (by Lemma 10)
\( \langle \langle t_1[s]\rangle_{\mu^+}, \langle t_2[s]\rangle_{\mu^+}, \pi(z, i) \rangle \in \mu^+[Q] \) iff (by definition of satisfaction and Property 8)
\( \mathcal{M}, \mu^+ \models (t_1[s] = t_2[s]) \).

6. For a formula of the form \( \exists \alpha \cdot M, \mu \models (t[\alpha])^+ \) iff (by definition of satisfaction)
\( e, w, z; \mu \models (t[\alpha])^+ \) iff (by induction)
\( \mathcal{M}, \mu^+ \models (t[\alpha])^+ \) iff (by definition of satisfaction)
\( \forall \tau \cdot \theta(\langle t_1[w]\rangle_{\mu^+}, \langle t_2[w]\rangle_{\mu^+}, \pi(z, i)) \in \mu^+[Q] \) iff (by Lemma 10)
\( \langle \langle t_1[s]\rangle_{\mu^+}, \langle t_2[s]\rangle_{\mu^+}, \pi(z, i) \rangle \in \mu^+[Q] \) iff (by definition of satisfaction)
\( \mathcal{M}, \mu^+ \models (t[\alpha])^+ \).

7. For a formula of the form \( \alpha \land \beta \cdot M, \mu \models (\alpha \land \beta)^+ \) iff (by induction)
\( \mathcal{M}, \mu^+ \models (\alpha \land \beta)^+ \).

8. For a formula of the form \( \lnot \alpha \cdot M, \mu \models \lnot \alpha \) iff (by induction)
\( \mathcal{M}, \mu^+ \models \lnot \alpha \).

9. For a formula of the form \( \forall V. \alpha \cdot M, \mu \models (\forall V. \alpha)^+ \) iff (by definition of satisfaction)
for all names \( n \) of the right sort, \( e, w, z; \mu \models (\forall V. \alpha)^+ \) iff (by induction)
for all names \( n \) of the right sort, \( \mathcal{M}, \mu^+ [n/\theta(n)] \models (\forall V. \alpha)(s) \) iff (by Property 1)
for all \( d \in D \) of the right sort, \( \mathcal{M}, \mu^+ [V/d] \models (\forall V. \alpha)(s) \) iff (by definition of satisfaction)
\( \mathcal{M}, \mu^+ \models (\forall V. \alpha)(s) \).

10. For a formula of the form \( \forall V. \alpha \cdot M, \mu \models (\forall V. \alpha)^+ \) iff (by Lemma 13)
for all relations \( Z \) over \( D \), \( e, w, z; \mu \models (\forall V. \alpha)(s) \) iff (by induction)
for all \( Z \) over \( D \), \( \mathcal{M}, \mu^+ [V/Z] \models (\forall V. \alpha)(s) \) iff (by definition of satisfaction)
\( \mathcal{M}, \mu^+ \models (\forall V. \alpha)(s) \).

11. For a formula of the form \( \Box \alpha \cdot M, \mu \models (\Box \alpha)^+ \) iff (by definition of satisfaction)
for all \( z \in Z \), \( e, w, z; \mu \models (\Box \alpha)^+ \) iff (by induction)
for all \( z \in Z \), \( \mathcal{M}, \mu^+ [s'/\pi(z, \cdot, i)] \models (\Box \alpha)(s') \) iff (by Property 9)
for all \( \tau \in D_{\text{aut}} \), if there exists \( z \in Z \) such that \( \tau = \pi(z, \cdot, i) \), then \( \mathcal{M}, \mu^+ [s'/\tau] \models (\Box \alpha)(s') \) iff (by definition of satisfaction)
\( \mathcal{M}, \mu^+ \models (\Box \alpha)(s) \).

12. For a formula of the form \( \text{Know}(\alpha) \cdot M, \mu \models (\text{Know}(\alpha))^+ \) iff (by definition of satisfaction)
for all \( w \in e \), if \( w \sim e \cdot w \), then \( e, w, z; \mu \models (\Box \alpha)^+ \) iff (by Property 2)
for all \( \omega(\cdot) \in e \), if \( \omega(\cdot) \sim e \cdot w \), then \( e, \omega(\cdot), z; \mu \models (\Box \alpha)^+ \) iff (by induction)
for all \( \omega(\cdot) \in e \), if \( \omega(\cdot) \sim e \cdot w \), then \( \mathcal{M}, \mu^+ [s'/\pi(z, \cdot, i)] \models (\Box \alpha)(s') \) iff (by Property 10)
for all \( \tau \in D_{\text{aut}} \), if there exists \( \ell \in D_{\text{aut}} \) such that \( \tau = \pi(z, \cdot, i) \) where \( \omega(\cdot) \in e \) and \( \omega(\cdot) \sim e \cdot w \), then \( \mathcal{M}, \mu^+ [s'/\tau] \models (\Box \alpha)(s') \) iff (by definition of satisfaction)
\[
\mathcal{M}, \mu^+ \models \forall s'(K(s', s) \supset \alpha(s')) \iff \\
\mathcal{M}, \mu^+ \models \text{Knows}(\alpha[\text{now}], s) \iff \\
\mathcal{M}, \mu^+ \models \text{Know}(\alpha)[s].
\]
This completes the proof. □

With this lemma in place, we can now prove the correctness of the embedding of ES into the situation calculus:

**Theorem 6.** Let \( \alpha \) be any basic sentence of ES without standard names. Then
\( \alpha \) is valid \( \iff \mathcal{Y} \models_{\text{FOL}} \alpha^* \).

**Proof.** First assume that \( \alpha \) is not valid. Then there is an \( e, w_0 \) such that \( e, w_0 \not\models \alpha \). Define a Tarski structure as follows:

- The domain of \( \mathcal{M} \) is \( D = D_{\text{obj}} \cup D_{\text{act}} \cup D_{\text{sit}} \), where \( D_{\text{obj}} \) (resp. \( D_{\text{act}} \)) is the set of standard names of objects (resp. actions), and \( D_{\text{sit}} = \mathcal{Z} \) times \( W \).
- The fixed vocabulary of \( \mathcal{M} \) is defined by:
  - \( \equiv_\mathcal{M} \) is the identity relation over \( D \),
  - \( S^0_\mathcal{M} = (\{\}, w_0) \),
  - \( d^\mathcal{M}(n, (z, w)) = (z \cdot n, w) \),
  - \( \Box^\mathcal{M} = [(z, w), (z \cdot z', w)] \),
  - \( K^\mathcal{M} = [[(z, w), (z, w_0)] \mid w \in e \text{ and } w \simeq_z w_0] \).
- For every rigid predicate symbol \( g \),
  \[
  G^\mathcal{M} = \{ (n_1, \ldots, n_k) \mid w_0[G(n_1, \ldots, n_k), (\{\}) = 1] \};
  \]
- For every fluent predicate symbol \( F \) (including Poss and SF),
  \[
  F^\mathcal{M} = \{ (n_1, \ldots, n_k, (z, w)) \mid w[F(n_1, \ldots, n_k), z] = 1] \};
  \]
- For every rigid function symbol \( g \),
  \[
  g^\mathcal{M}(n_1, \ldots, n_k) = w_0[g(n_1, \ldots, n_k), (\{}];
  \]
- For every fluent function symbol \( f \),
  \[
  f^\mathcal{M}(n_1, \ldots, n_k, (z, w)) = w[g(n_1, \ldots, n_k), z].
  \]

These definitions ensure that \( \mathcal{M} \) satisfies \( \mathcal{Y} \). Next, define the mappings \( \theta, \pi \), and \( \omega \) by letting \( \theta(n) = n \), and for any initial \( i = (\{\}, w) \), letting \( \pi(z, i) = (z, w) \), and \( \omega(i) = w \). This ensures that the properties needed for Lemma 14 are satisfied, and so \( \mathcal{M} \not\models \alpha^* \). Consequently, \( \mathcal{Y} \not\models_{\text{FOL}} \alpha^* \).

Conversely, assume that \( \mathcal{Y} \not\models_{\text{FOL}} \alpha^* \). Then there is a Tarski structure \( \mathcal{M} \) that satisfies \( \mathcal{Y} \) but such that \( \mathcal{M} \not\models \alpha^* \). The domain \( D \) must be \( D_{\text{sit}} \cup D_{\text{act}} \cup D_{\text{obj}} \), with \( D_{\text{sit}} \subseteq D_{\text{sit}} \) as the set of starting situations, and \( w_0 = S^0_\mathcal{M} \in D_{\text{sit}} \). Since \( \mathcal{M} \models \mathcal{Y} \), both \( D_{\text{obj}} \) and \( D_{\text{act}} \) are countably infinite, say \( D_{\text{obj}} = \{d_1, d_2, \ldots\} \), and \( D_{\text{act}} = \{\lambda_1, \lambda_2, \ldots\} \). We define the mappings \( \theta, \pi \), and \( \omega \) by the following:

- \( \theta \) maps the \( i \)-th standard name for objects to \( \delta_i \), and the \( i \)-th standard name for actions to \( \lambda_i \);
- for any \( z \in \mathcal{Z} \) and \( i \in D_{\text{sit}} \), we define \( \pi(z, i) \) by:
  \[
  \pi(\{\}, i) = i, \\
  \pi(z \cdot n, i) = \alpha^M(\theta(n), \pi(z, i));
  \]
- for any \( i \in D_{\text{sit}} \), we let \( \omega(i) \) be the world \( w \) defined by the following:
  \[
  w[G(n_1, \ldots, n_k), z] = 1 \iff \theta(n_1, \ldots, \theta(n_k)) \in G^\mathcal{M}, \\
  w[F(n_1, \ldots, n_k), z] = 1 \iff \theta(n_1, \ldots, \theta(n_k), \pi(z, i)) \in F^\mathcal{M}, \\
  w[g(n_1, \ldots, n_k), z] = \theta^{-1}(g^\mathcal{M}(\theta(n_1), \ldots, \theta(n_k))), \\
  w[f(n_1, \ldots, n_k), z] = \theta^{-1}(f^\mathcal{M}(\theta(n_1), \ldots, \theta(n_k), \pi(z, i))).
  \]

For any \( i \in D_{\text{sit}} \), \( \omega(i) \) obviously satisfies the rigidity constraint. Finally, we let

- \( e = \{\omega(i) \mid (i, t_0) \in K^\mathcal{M}\} \);
- \( w_0 = \omega(t_0) \).
These definitions ensure that all of the properties needed for Lemma 14 are satisfied, and so \( e, w_0 \not\models \alpha \). Consequently, \( \alpha \) is not valid. \( \square \)

**Theorem 7.** Let \( \alpha \) be any sentence of ES without standard names. Then
\[
\alpha \text{ is valid } \iff \Sigma \cup \Upsilon \models_{\text{FOL}} \alpha^*.
\]

**Proof.** The proof is almost the same as the previous one because we can re-use the same model constructions.

Let us first assume that \( \alpha \) is not valid. Then there is an \( e, w_0 \) such that \( e, w_0 \not\models \alpha \). Now define a Tarski structure \( \mathcal{M} \) exactly as in the first part of the proof of Theorem 6. It is easy to see that \( \Sigma \) (as well as \( \Upsilon \)) is satisfied by \( \mathcal{M} \). In particular, the construction ensures that situations have unique names and that neither \( S_0 \) nor any situation \( K \)-accessible from \( S_0 \) has a predecessor, that is, these are truly initial situations. Furthermore, by construction, the situations reachable from the initial situations are the only situations of the model so that the induction axiom for situations (Axiom 2) is also satisfied. The rest of the argument is exactly as in Theorem 6, from which \( \Sigma \cup \Upsilon \not\models_{\text{FOL}} \alpha^* \) follows.

Conversely, assume that \( \Sigma \cup \Upsilon \not\models_{\text{FOL}} \alpha^* \). Then there is a Tarski structure \( \mathcal{M} \) that satisfies \( \Sigma \cup \Upsilon \) but such that \( \mathcal{M} \not\models \alpha^* \). We define the mappings \( \theta, \pi \) and \( \omega \) as well as \( w_0 \) and \( e \) exactly as in Theorem 6. As before, all the properties of Lemma 14 are satisfied and, hence, \( e, w_0 \not\models \alpha \), which proves that \( \alpha \) is not valid. \( \square \)

**References**


