Infinite Paths in the Situation Calculus (Extended Version)

Shakil M. Khan and Yves Lespérance

Technical Report EECS-2015-05

December 1 2015

Department of Electrical Engineering and Computer Science
4700 Keele Street, Toronto, Ontario M3J 1P3 Canada

Infinite Paths in the Situation Calculus (Extended Version)

Shakil M. Khan and **Yves Lespérance**
Department of Electrical Engineering and Computer Science
York University, Toronto, Ontario, Canada
Email: {skhan, lesperan}@cse.yorku.ca

Abstract

The situation calculus has proved to be a very popular formalism for modeling and reasoning about dynamic systems. This otherwise elegant and refined language however lacks a natural way of dealing with "infinite future histories". To this end, in this paper we introduce a new sort ranging over *infinite paths* in the situation calculus and propose an axiomatization for infinite paths. We thus obtain a convenient way of specifying several kinds of notions that involve infinite futures such as temporal properties of non-terminating executions of agents or programs and mental attitudes such as desires and intentions. We prove the correctness of the axiomatization and show that our formalization has some intuitively desirable properties. We then illustrate the usefulness of our formalization by considering applications involving desires and non-terminating programs.

1. Introduction

One of the most prominent formalisms for modeling dynamic domains is the situation calculus (McCarthy and Hayes 1969). Since its introduction, much work has been done to further refine and enrich this language. Researchers have formalized the prerequisites and effects of actions and proposed reasonable solutions to the frame problem (Reiter 1991; 2001) and the ramification problem (Lin and Reiter 1994). They have modeled various aspects of agents' incomplete knowledge, perceptual actions, and knowledge change (Moore 1990; Scherl and Levesque 2003; Levesque 1996; Lakemeyer and Levesque 1998), agent ability (Lespérance et al. 2000; Lespérance 2001), continuous time (Pinto 1994), stochastic actions (Reiter 2001), agent communication (Shapiro, Lespérance, and Levesque 2002; Khan and Lespérance 2005), belief revision (Shapiro et al. 2011), and dealt with agents' motivational attitudes and rational action/behavior (Shapiro, Lespérance, and Levesque 2005; Sardina and Shapiro 2003; Khan and Lespérance 2010). Finally, work has been done to incorporate complex actions into the formalism that has led to an implemented family of "high-level" programming languages (Levesque et al. 1997; De Giacomo, Lespérance, and Levesque 2000; Boutilier et al. 2000; De Giacomo et al. 2004).

However, this otherwise simple yet elegant language lacks a convenient way of dealing with "infinite future histories". One cannot talk directly about infinite paths in the situation calculus. While some work has been done to capture the notion of paths in the situation calculus, all of these approaches have drawbacks. We discuss some of these here; see Section 8 for a discussion of other related work. While specifying agents' goals and behavior, Shapiro (2005) considers only finite paths. He models a finite path using a pair of situations representing the beginning state and the ending state of the path. Unfortunately, a temporal framework based on such finite paths has limited expressiveness and can't capture arbitrary temporally extended formulae, e.g. the goal to maintain a property ϕ indefinitely far in the future, $\Box\phi$. Also, quantification over these finite paths requires dealing with a pair of situations explicitly which is somewhat clumsy.

Lespérance et al. (2000) on the other hand looked at infinite paths. They introduced the notion of *action selection functions* (also called *ASF* or *strategies*), which are mappings from situations to primitive actions, and showed how ASFs can be used to model infinite paths (see Section 3 for details). Their account however does not have paths as a sort and thus does not allow for first-order quantification over paths.

To deal with these issues, in this paper we introduce a new sort of infinite *paths* (along with path variables that can be quantified over) in the situation calculus and propose an axiomatization for infinite paths. By adding a new sort ranging over infinite paths in the situation tree, we obtain a convenient way of specifying several kinds of notions that involve infinite futures such as temporal properties of non-terminating executions of agents or programs, and future looking mental attitudes such as desires and intentions, as well as beliefs about the future. We discuss how CTL* temporal formulae (Emerson and Halpern 1986) can be interpreted over these infinite paths. We show that our formalization of paths has some intuitively reasonable properties and prove the correctness of the axiomatization. To illustrate the benefits of paths, we then examine two applications of the situation calculus with paths, namely specification of temporal goals of agents and definition of execution semantics for infinitely running programs written in the situation calculus-based ConGolog programming language (De Giacomo, Lespérance, and Levesque 2000).

2. Background

We adopt the version of the situation calculus as formalized in (Reiter 2001). In this framework, a possible state of the domain is represented by a situation. The initial situation is represented by a special constant S_0 which stands for the empty sequence of actions. There is a distinguished binary function symbol *do* where $do(a, s)$ denotes the successor situation to s resulting from performing the action a. Thus the domain of situations can be viewed as a tree, where the root of the tree is the initial situation S_0 and the arcs represent actions (this is enforced by the foundational axioms; see below). Relations (and functions) whose truth values vary from situation to situation, are called relational (functional, resp.) fluents, and are denoted by predicate (function, resp.) symbols taking a situation term as their last argument. There is a special predicate $\text{Poss}(a, s)$ used to state that action a is executable in situation s. $s \sqsubset s'$ means that s' can be reached from s by performing a sequence of actions. $s \sqsubseteq s'$ is an abbreviation for $s \sqsubset s' \vee s = s'$. We will use $s \prec s'$ and $s \preceq s'$ to denote that the sequence of actions performed to reach s' from s were all executable. Finally, a situation is called executable if every action in its history was executable, i.e.:

$$\text{Executable}(s) \doteq \forall a, s'.\ do(a, s') \preceq s \supset \text{Poss}(a, s').$$

A dynamic domain can be represented by a *basic action theory* (Reiter 2001) $\mathcal{D}_{bat}$ that includes: (1) action precondition axioms, one per action a characterizing $\text{Poss}(a, s)$, (2) successor state axioms (SSA), one per fluent, that succinctly encode both effect and frame axioms and specify exactly how and when the fluent changes (Reiter 2001), (3) initial state axioms describing what is true initially (including the mental states of the agents), (4) unique names axioms for actions, and (5) domain-independent foundational axioms Σ describing the structure of situations (Levesque, Pirri, and Reiter 1998). All these axioms are first-order except for Σ, which includes a second order induction axiom for defining the tree of situations.

3. Infinite Paths in the Situation Calculus

Following (Lespérance et al. 2000), we only consider "realistic" paths; paths involving non-executable actions cannot really occur as they are not realistic. Thus a path in our framework is essentially an infinite sequence of situations, where each situation along the path can be reached by performing some *executable* action in the preceding situation. To allow (first-order) quantification over infinite paths, we in addition introduce a new sort called paths in the language with (possibly sub/super-scripted) variables p ranging over paths. We give an axiomatization for infinite paths below.

Thus our formalization of infinite paths is more general than Shapiro's (2005) finite paths. Arbitrary temporally extended formulae such as unbounded maintenance goals can be interpreted using our paths. Moreover, our account is simpler than that of (Lespérance et al. 2000), and unlike them, we allow quantification over paths, which makes our language easier to use.

Before delving into the technical details, let us point out some notational conventions. We will use both state and path formulae denoted by uppercase and lowercase Greek letters, resp., e.g. $\Phi(s)$ and $\phi(p)$. Here s is a free situation variable in which the state formula must be evaluated and p is a free path variable over which the path formula must hold. We sometimes suppress these variables where the intended meaning is clear. Finally, as usual $\Phi[s]$ denotes the reintroduction of situation s in the situation suppressed formula Φ.

Axiomatization: We now give our axiomatization for infinite paths. We have a predicate $\text{OnPath}(p, s)$, meaning that situation s is on path p. Also, the abbreviation $\text{Starts}(p, s)$ means that s is the starting situation of path p. A path p starts with s iff s is the earliest situation on p:

Definition 1.

$$\text{Starts}(p, s) \doteq \text{OnPath}(p, s) \wedge \forall s'.\ \text{OnPath}(p, s') \supset s \preceq s'.$$

As shown in (Lespérance et al. 2000), one can use action selection functions (ASFs) to model infinite paths. Recall that ASFs or strategies are mappings from situations to primitive actions. The idea is that given a situation s, an ASF F prescribes an action that the agent must perform in s if she were to follow the path induced by this strategy. An infinite path can then be formalized as a tuple (s, F), where s is the starting situation of the path, and F is a strategy that defines an infinite sequence of situations by specifying an action for every situation starting from s. Thus, one way of axiomatizing paths is by making them correspond to such pairs (s, F):

Axiom 2.

$$\begin{aligned}&(a).\ \forall p.\ (\exists F, s.\ \text{Executable}(F, s)\\ &\qquad\qquad \wedge \forall s'.\ \text{OnPath}(p, s') \equiv \text{OnPathASF}(F, s, s')),\\ &(b).\ \forall F, s.\ \text{Executable}(F, s) \supset (\exists p.\ \text{Starts}(p, s) \wedge\\ &\qquad\qquad \forall s'.\ \text{OnPathASF}(F, s, s') \equiv \text{OnPath}(p, s')).\end{aligned}$$

This second-order axiom says that for every path p, there is an action selection function F and a situation s such that F starting in s is executable, and that F produces exactly the same sequence of situations on p starting from s. Also, for every executable action selection function F and situation s, there is a path p that starts with s and that corresponds exactly to the sequence of situations produced by F starting from s. Here, $\text{OnPathASF}(F, s, s')$ means that the situation sequence defined by (s, F) includes the situation s':

Definition 3.

$$\begin{aligned}&\text{OnPathASF}(F, s, s') \doteq\\ &\qquad s \preceq s' \wedge \forall a, s^*.\ s \prec do(a, s^*) \preceq s' \supset F(s^*) = a.\end{aligned}$$

Also, the situation sequence encoded by a strategy F and a starting situation s is executable iff s is executable, and for all situations s' on this sequence, the action selected by F in s' is executable in s'.

Definition 4.

$$\begin{aligned}&\text{Executable}(F, s) \doteq \text{Executable}(s) \wedge\\ &\qquad \forall s'.\ \text{OnPathASF}(F, s, s') \supset \text{Poss}(F(s'), s').\end{aligned}$$

Another axiom is needed to state that different situation sequences represent different paths.

Axiom 5.

$$\forall p, p'.\ (\forall s.\ \text{OnPath}(p, s) \equiv \text{OnPath}(p', s)) \equiv p = p'.$$

Note that, for every situation s on a path, there must be an action that is possible in s, i.e. $\forall p, s.\ \text{OnPath}(p, s) \supset \exists a.\ \text{Poss}(a, s)$. We consider that situations where no action is possible are "artificial". One can always introduce a dummy action $noOp$ that has the precondition that True, and consequently is always executable. Taking paths to be sequences of executable situations means that there may be infinite sequences of successor situations that are not paths; even if the situations on a prefix of a sequence are executable, the presence of a non-executable situation in the sequence means that it is not a path. One could easily modify the above axiomatization to include paths with non-executable situations, and identify the subset of such paths that are executable.

Also, while we focus on infinite paths, finite (executable) paths can be viewed as prefixes of paths since a finite path can always be extended to an infinite one, e.g. by extending the prefix with an infinite sequence of $noOp$ actions.

We now define what it means for a path p' to be a *suffix* of another path p w.r.t. a situation s:

Definition 6.

$$\text{Suffix}(p', p, s) \doteq \text{OnPath}(p, s) \wedge \text{Starts}(p', s)$$
$$\wedge \forall s'.\ s \preceq s' \supset (\text{OnPath}(p, s') \equiv \text{OnPath}(p', s')).$$

That is, a path p' is a suffix of another path p w.r.t. a situation s iff s is on p, and p' which starts with s, contains exactly the same situations as p starting from s. Below, we denote by $\mathcal{D}_{path}$ the axioms and definitions for modeling paths in the situation calculus.

4. CTL* in the Situation Calculus

CTL* (Emerson and Halpern 1986) is an expressive branching-time temporal logic that allows arbitrary combinations of path quantifiers and temporal operators with no additional complexity (Schnoebelen 2002) over LTL (Pnueli 1977). In this section, we show how CTL* formulae can be interpreted over the situation calculus with paths. We start by inductively defining a class of state and path formulae (we use Φ and ϕ to denote state and path formulae, resp.):

$$\Phi ::= \varphi \mid E\phi \mid A\phi \mid \Phi \wedge \Phi \mid \neg\Phi \mid \forall x.\ \Phi$$
$$\phi ::= \Phi \mid \phi \wedge \phi \mid \neg\phi \mid \forall x.\ \phi \mid \bigcirc\phi \mid \phi\, \mathcal{U}\, \phi$$

Here, φ is an arbitrary situation suppressed situation calculus formula. $E\phi$, i.e. *over some path* ϕ and $A\phi$, i.e. *over all paths* ϕ are path quantifiers that are used to construct state formulae from given path formulae. Moreover, any state formula is also a path formula. Thus we need to be able to evaluate a state formula Φ over a path. But since state formulae can only be evaluated over a state (in our case, a situation), following (Emerson and Halpern 1986) we evaluate it w.r.t. the starting situation of the path (see below). $\bigcirc\phi$ means that ϕ holds *next* over a path while $\phi\ \mathcal{U}\ \psi$ stands for ϕ *until* ψ. Finally, other logical connectives like $\vee, \supset, \subset, \equiv$, and $\exists$ are handled as the usual abbreviations.

Let us now define a function $[\![\cdot]\!]$ that translates these formulae into formulae of the situation calculus with paths. We write $\Phi[\![s]\!]$ (and $\phi[\![p]\!]$) to mean that state formula Φ (and path formula ϕ) holds in situation s (and over path p, resp.). $[\![\cdot]\!]$ is defined inductively as follows:

Definition 7.

A. $\varphi[\![s]\!] \doteq \varphi[s]$, where φ is a sit. suppressed sit. calc. formula,
B. $E\phi[\![s]\!] \doteq \exists p.\ \text{Starts}(p, s) \wedge \phi[\![p]\!]$,
C. $A\phi[\![s]\!] \doteq \forall p.\ \text{Starts}(p, s) \supset \phi[\![p]\!]$,
D. $(\Phi \wedge \Psi)[\![s]\!] \doteq \Phi[\![s]\!] \wedge \Psi[\![s]\!]$,
E. $\neg\Phi[\![s]\!] \doteq \neg(\Phi[\![s]\!])$,
F. $(\forall x.\ \Phi)[\![s]\!] \doteq \forall x.(\Phi[\![s]\!])$,

G. $\Phi[\![p]\!] \doteq \exists s.\ \text{Starts}(p, s) \wedge \Phi[\![s]\!]$,
H. $(\phi \wedge \psi)[\![p]\!] \doteq \phi[\![p]\!] \wedge \psi[\![p]\!]$,
I. $\neg\phi[\![p]\!] \doteq \neg(\phi[\![p]\!])$,
J. $(\forall x.\ \phi)[\![p]\!] \doteq \forall x.(\phi[\![p]\!])$,
K. $\bigcirc\phi[\![p]\!] \doteq$
$\exists s, a, p'.\ \text{Starts}(p, s) \wedge \text{Suffix}(p', p, do(a, s)) \wedge \phi[\![p']\!]$,
L. $(\phi\, \mathcal{U}\, \psi)[\![p]\!] \doteq \exists s, s', p'.\ \text{Starts}(p, s) \wedge \text{Suffix}(p', p, s') \wedge \psi[\![p']\!]$
$\wedge (\forall s^*, p^*.\ s \preceq s^* \prec s' \wedge \text{Suffix}(p^*, p, s^*) \supset \phi[\![p^*]\!])$.

Thus, the situation suppressed situation calculus formula φ holds in situation s if $\varphi[s]$ is true, i.e. the formula obtained by reintroducing situation s in φ holds in s. $E\phi[\![s]\!]$ holds if ϕ holds over at least one path in the future of s. On the other hand, $A\phi[\![s]\!]$ holds if ϕ holds over all the paths in the future of s. ϕ holds next over a path p (i.e. $\bigcirc\phi[\![p]\!]$) if ϕ holds over the suffix of p that starts with the successor to the starting situation of p. ϕ until ψ holds over a path p (i.e. $(\phi\ \mathcal{U}\ \psi)[\![p]\!]$) if there is a suffix p' of p that starts with s', ψ holds over p', and for all suffixes p^* of p that start with an earlier situation s^* than s' (i.e. $s^* \prec s'$), ϕ holds over p^*. The rest are self-explanatory. It should be clear that our semantics of CTL* is essentially the same as the one given by Emerson and Halpern (1986).

Other CTL* operators can be defined as usual, e.g. *eventually* ϕ (denoted by $\Diamond\phi$), *always* ϕ (denoted by $\Box\phi$), ϕ *unless* ψ (denoted by $\phi\ \mathcal{W}\ \psi$), ϕ *before* ψ (denoted by $\phi\ \mathcal{B}\ \psi$), etc.

Definition 8.

A. $\Diamond\phi[\![p]\!] \doteq (\text{True}\, \mathcal{U}\ \phi)[\![p]\!]$,
B. $\Box\phi[\![p]\!] \doteq \neg\Diamond\neg\phi[\![p]\!]$,
C. $(\phi\, \mathcal{W}\, \psi)[\![p]\!] \doteq ((\phi\, \mathcal{U}\, \psi) \vee \Box(\phi \wedge \neg\psi))[\![p]\!]$,
D. $(\phi\, \mathcal{B}\, \psi)[\![p]\!] \doteq \neg(\neg\phi\, \mathcal{U}\, \psi)[\![p]\!]$.

Let $\mathcal{D}_{CTL^*}$ be our definitions of CTL* above.

Example: Consider an agent in a mine-sweeper-like domain. Assume that there are only a finite number of cells (one can also handle a version where the grid of cells is infinite). The agent has two actions $visit(x, y)$ and $flag(x, y)$, for visiting cell (x, y) and flagging/marking that cell (x, y) possibly has a mine, respectively. There are three fluents, $\text{Dead}(s)$, $\text{Visited}(x, y, s)$, and $\text{Flagged}(x, y, s)$. Initially the agent is alive, but she can die if she accidentally steps on a cell with a mine. Initially all the cells are neither visited nor flagged. The objective here is to visit as many cells as possible without stepping on a mine and dying. If the agent suspects that a cell has a mine in it, she can block the possibility of visiting it by flagging it provided that she is alive. Also, she can visit a cell if it is not flagged, again provided that she is alive.[1] The action theory $\mathcal{D}_{swip}$ includes the foundational axioms Σ, $\mathcal{D}_{path}$, $\mathcal{D}_{CTL^*}$, unique names axioms for $visit()$ and $flag()$ actions, and the following initial situation description, and precondition and successor-state axioms:

$$\exists n.\ \text{Cell}(x, y) \equiv (x = 1 \land y = 1) \lor \cdots \lor (x = n \land y = n),$$
$$\text{Mine}(x, y) \supset \text{Cell}(x, y),$$

$$\neg\text{Dead}(S_0) \land \neg\text{Visited}(x, y, S_0) \land \neg\text{Flagged}(x, y, S_0),$$

$$\text{Poss}(visit(x, y), s) \equiv \text{Cell}(x, y) \land \neg\text{Dead}(s) \land \neg\text{Flagged}(x, y, s),$$
$$\text{Poss}(flag(x, y), s) \equiv \text{Cell}(x, y) \land \neg\text{Dead}(s),$$

$$\text{Dead}(do(a, s)) \equiv \text{Dead}(s) \lor (a = visit(x, y) \land \text{Mine}(x, y)),$$
$$\text{Visited}(x, y, do(a, s)) \equiv \text{Visited}(x, y, s) \lor (a = visit(x, y) \land \text{Cell}(x, y)),$$
$$\text{Flagged}(x, y, do(a, s)) \equiv \text{Flagged}(x, y, s) \lor (a = flag(x, y) \land \text{Cell}(x, y)).$$

Now, define Solved(s) as follows:

$$\text{Solved}(s) \doteq \forall x, y.\ \text{Mine}(x, y) \supset \text{Flagged}(x, y, s) \land \text{Cell}(x, y) \land \neg\text{Mine}(x, y) \supset \text{Visited}(x, y, s).$$

Then from $\mathcal{D}_{swip}$, we have that there is a path where sooner or later the problem will be solved. Using CTL* introduced above, we have that:

$$\mathcal{D}_{swip} \models E(\Diamond\text{Solved}()).$$

Also, from $\mathcal{D}_{swip}$ and the additional constraint that there is at least one cell with a mine in it, we have that there is a path where eventually the agent dies and that for all paths she will be always alive as long as she never visits any cells:

$$\mathcal{D}_{swip} \cup \{\exists x, y.\ \text{Mine}(x, y)\} \models E(\Diamond\text{Dead}()),$$
$$\mathcal{D}_{swip} \cup \{\exists x, y.\ \text{Mine}(x, y)\} \models A(\forall x, y.\ \Box\neg\text{Visited}(x, y) \supset \Box\neg\text{Dead}()).$$

On the other hand, if we assume that there are no mines at all, then we have that for all futures she is always alive:

$$\mathcal{D}_{swip} \cup \{\forall x, y.\ \neg\text{Mine}(x, y)\} \models A(\Box\neg\text{Dead}()).$$

5. Properties of Paths

We now show some properties of our axiomatization of paths. Proofs of the properties appear in the appendix.

Let Σ be the set of foundational axioms, and $\mathcal{D}_{path}$ consist of the axiomatization for paths, i.e. Axioms 2 and 5 and the associated definitions. Our first property captures the conditions under which a situation can be extended to a path: $\Sigma \cup \mathcal{D}_{path}$ entails that for any executable situation, there is a path that starts with that situation, provided that for any situation there exists an executable action.

[1]Since the agent cannot perform an action when she is dead, we assume the inclusion of other executable actions, e.g. a $noOp$ action that is executable when she is dead so that she does not run out of executable actions.

Proposition 9.

$$\Sigma \cup \mathcal{D}_{path} \models (\forall s'.\ \exists a.\ \text{Poss}(a, s')) \supset (\forall s.\ \text{Executable}(s) \supset \exists p.\ \text{Starts}(p, s)).$$

This can be proven by constructing an executable ASF and using Axiom 2(b). Again, we maintain that situations with no executable actions are "artificial".

Next, we prove some properties of the starting situation of a path. In particular, we can show that $\Sigma \cup \mathcal{D}_{path}$ entails that (a) any path starts with some situation, (b) the starting situation of any path is unique, and (c) the starting situation of any path is executable.

Proposition 10.

(a). $\Sigma \cup \mathcal{D}_{path} \models \forall p.\ \exists s.\ \text{Starts}(p, s)$,

(b). $\Sigma \cup \mathcal{D}_{path} \models \forall p, s, s'.\ \text{Starts}(p, s) \land \text{Starts}(p, s') \supset s = s'$,

(c). $\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ \text{Starts}(p, s) \supset \text{Executable}(s)$.

These properties can be proven using Axiom 2(a) and Definitions 1, 3, and 4.

The next two properties deal with the successor situation of a situation on a path that is also on the path. The first states that $\Sigma \cup \mathcal{D}_{path}$ entails that for any situation s on a path p, there is a successor situation $s' = do(a, s)$ on p, and s' can be reached from s by performing an executable action a.

Proposition 11.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ \text{OnPath}(p, s) \supset \exists s', a.\ \text{OnPath}(p, s') \land s' = do(a, s) \land \text{Poss}(a, s).$$

This can be proven by Axiom 2(a) and Definitions 3 and 4.

Moreover, $\Sigma \cup \mathcal{D}_{path}$ entails that the successor situation of a situation on a path is unique.

Proposition 12.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ [\text{OnPath}(p, s) \land \text{OnPath}(p, do(a, s)) \land \text{OnPath}(p, do(b, s))] \supset a = b.$$

We can prove this using Axiom 2(a) and Definition 3.

The next property deals with the uniqueness of paths: $\Sigma \cup \mathcal{D}_{path}$ entails that if $p \neq p'$, then there is a situation that is on path p but not on path p'.

Proposition 13.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, p'.\ p \neq p' \supset \exists s.\ (\text{OnPath}(p, s) \land \neg\text{OnPath}(p', s)).$$

This follows from Axiom 5.

We can also show that $\Sigma \cup \mathcal{D}_{path}$ entails that all situations on a path are executable.

Corollary 14.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ \text{OnPath}(p, s) \supset \text{Executable}(s).$$

This can be proven by induction on s using Propositions 10(b, c), 11, and 12.

We say that two situations are co-linear if they are the same or if one of them strictly precedes the other. Our next set of properties deal with the structure of situations on paths and shows that paths are essentially linear sequences of situations. First, we have $\Sigma \cup \mathcal{D}_{path}$ entails that any pair of situations on the same path are co-linear:

Proposition 15.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s, s'. \text{ OnPath}(p, s) \wedge \text{OnPath}(p, s') \supset s = s' \vee s \prec s' \vee s' \prec s.$$

We can prove this using Axiom 2(a), Definition 3, and other properties of structure of situations w.r.t. $\prec$.

Secondly, we have $\Sigma \cup \mathcal{D}_{path}$ entails that if situations s and s' are on a given path p, then all situations in the interval defined by these two situations are also on p.

Proposition 16.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s, s', s^*. \text{ (OnPath}(p, s) \wedge \text{OnPath}(p, s') \wedge s \preceq s^* \preceq s') \supset \text{OnPath}(p, s^*).$$

We can prove this using Axiom 2(a) and Definition 3.

Finally, we can show that $\Sigma \cup \mathcal{D}_{path}$ entails that two paths can share only one common prefix. Once they branch at some situation, they never merge after that.

Proposition 17.

$$\begin{aligned}\Sigma \cup \mathcal{D}_{path} \models \forall p_1, p_2, s, a, b, s_1, s_2. \text{ [OnPath}(p_1, do(a, s)) \\ \wedge \text{OnPath}(p_2, do(b, s)) \wedge a \neq b \wedge s \prec s_1 \wedge s \prec s_2 \\ \wedge \text{OnPath}(p_1, s_1) \wedge \text{OnPath}(p_2, s_2)] \supset s_1 \neq s_2.\end{aligned}$$

We can prove this using Proposition 15 and other properties of structure of situations w.r.t. actions and $\prec$.

The next few properties deal with suffixes and prefixes of a given path. The first of these states that $\Sigma \cup \mathcal{D}_{path}$ entails that for any situation s on a path p, there is a suffix of p that starts with s.

Proposition 18.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s. \text{ OnPath}(p, s) \supset \exists p'. \text{ Suffix}(p', p, s).$$

This can be proven using Axioms 2(a, b), and Definitions 3, 4, and 6.

Secondly, we can show that given a path p with starting situation $do(a, s)$, $\Sigma \cup \mathcal{D}_{path}$ entails that there is a path p' s.t. p' starts with s, and p is a suffix of p' starting from $do(a, s)$.

Proposition 19.

$$\Sigma \cup \mathcal{D}_{path} \models \text{Starts}(p, do(a, s)) \supset \exists p'. \text{ Starts}(p', s) \wedge \text{Suffix}(p, p', do(a, s)).$$

This can be proven using Axioms 2(a, b), and Definitions 1, 3, 4, and 6, and Proposition 10(b).

Finally, $\Sigma \cup \mathcal{D}_{path}$ entails that any path that starts with a non-initial situation can be extended in the past; formally, for all situations s_1 and s_2, if s_1 strictly precedes s_2 and there is a path p_2 that starts with s_2, then there must also exist a path p_1 such that p_1 starts with s_1 and p_2 is a suffix of p_1 starting from s_2.

Proposition 20.

$$\Sigma \cup \mathcal{D}_{path} \models \forall s_1, s_2, p_2. \ s_1 \prec s_2 \wedge \text{Starts}(p_2, s_2) \supset \exists p_1. \text{ Starts}(p_1, s_1) \wedge \text{Suffix}(p_2, p_1, s_2).$$

We can prove this by induction on situation s_2 using Proposition 19 and Definition 6.

We now prove some second-order induction principles for paths and for situations in a path. First we have $\Sigma \cup \mathcal{D}_{path}$ entails that if some property Q holds for all paths that start with an initial situation, and if whenever Q holds for all paths that start with situation s, then it holds for all paths that start with any successor situation to s, then the property Q holds for all paths.

Theorem 21 (Induction on Paths)**.**

$$\begin{aligned}\Sigma \cup \mathcal{D}_{path} \models \forall Q. \ [\{\forall s, p. \text{ Init}(s) \wedge \text{Starts}(p, s) \supset Q(p)\} \wedge \\ \{\forall a, s. \ (\forall p. \text{ Starts}(p, s) \supset Q(p)) \\ \supset (\forall p'. \text{ Starts}(p', do(a, s)) \supset Q(p'))\}] \\ \supset \forall p. \ Q(p).\end{aligned}$$

Moreover, $\Sigma \cup \mathcal{D}_{path}$ entails that if some property Q holds for the starting situation of a given path p, and if whenever Q holds for a situation s on path p, then it holds for the successor situation to s on p, then the property Q holds for all situations on path p.

Theorem 22 (Induction on Situations in a Path)**.**

$$\begin{aligned}\Sigma \cup \mathcal{D}_{path} \models \forall p, Q. \ [\{\forall s. \text{ Starts}(p, s) \supset Q(s)\} \wedge \\ \{\forall a, s. \ (\text{OnPath}(p, s) \wedge Q(s) \wedge \text{OnPath}(p, do(a, s))) \\ \supset Q(do(a, s))\}] \\ \supset \forall s. \text{ OnPath}(p, s) \supset Q(s).\end{aligned}$$

See the appendix for proofs of these.

Next, we prove the correctness of our axiomatization. A natural way of capturing the notion of infinite path is by specifying it as a mapping from the set of natural numbers to situations on a path. To this end, we use a function σ of the following sort (here $\mathcal{S}$ denotes the set of all situations): $\sigma : \mathbb{N} \to \mathcal{S}$. We say that such a function σ models a *path sequence* if σ maps the number 0 to an executable situation (representing the starting situation of the path), and for each number n, there is an action a that is executable in the situation s_n produced by $\sigma(n)$ such that σ maps the immediate successor of n (i.e. $n + 1$) to the situation $do(a, s_n)$.

Definition 23.

$$\begin{aligned}\text{PathSeq}(\sigma) \doteq \text{Executable}(\sigma(0)) \wedge \\ \forall n. \ \exists a. \text{ Poss}(a, \sigma(n)) \wedge \sigma(n + 1) = do(a, \sigma(n)).\end{aligned}$$

We say that a path p matches a path sequence σ if σ is indeed a path sequence, $\sigma(0)$ is the starting situation of p, and for all n, s and a, if $\sigma(n)$ is a situation s on path p, then $\sigma(n + 1)$ is the successor situation $do(a, s)$ of s on p:

Definition 24.

$$\begin{aligned}\text{Matches}(p, \sigma) \doteq \text{PathSeq}(\sigma) \wedge (\sigma(0) = s \equiv \text{Starts}(p, s)) \\ \wedge \forall n, s. \ [\sigma(n) = s \wedge \text{OnPath}(p, s) \supset \\ \forall a. \ (\sigma(n + 1) = do(a, s) \equiv \text{OnPath}(p, do(a, s)))].\end{aligned}$$

Given this formalization, the task of proving correctness of our axiomatization for infinite paths can be reduced to showing that path sequences are isomorphic to paths defined by $\Sigma \cup \mathcal{D}_{path}$, i.e. that there is an one-to-one mapping between these two. To this end, we first show that for any path p, there is a path sequence σ that matches p.[2]

[2]Here $\Sigma_{\mathbb{N}}$ is an axiomatization of the natural numbers, i.e., standard second-order Peano arithmetic, for the natural number sort.

Theorem 25 (Soundness)**.**

$$\Sigma_{\mathbb{N}} \cup \Sigma \cup \mathcal{D}_{path} \models \forall p.\ (\exists \sigma.\ \text{PathSeq}(\sigma) \wedge \text{Matches}(p, \sigma)).$$

See the appendix for a proof. Note that our soundness result implies that for any path p, there is a countably infinite number of distinct situations on p.

Conversely, for any path sequence σ, there is a path p that matches σ.

Theorem 26 (Completeness)**.**

$$\Sigma_{\mathbb{N}} \cup \Sigma \cup \mathcal{D}_{path} \models \forall \sigma.\ \text{PathSeq}(\sigma) \supset \exists p.\ \text{Matches}(p, \sigma).$$

A proof for this appears in the appendix.

6. Application: Temporally Extended Goals

In this section, we illustrate the utility of introducing infinite paths in the situation calculus by showing how these paths can be used to formalize motivational attitudes. Our formalization is a simplified version of that in (Khan and Lespérance 2010) and is only meant to demonstrate the use of infinite paths in formalizing *simple temporally extended goals* in the situation calculus.

Here, we use a variant of basic action theories described in (Levesque, Pirri, and Reiter 1998). In particular, to deal with multiple possible worlds, rather than having just one initial situation, we assume that there is a set of possible initial situations, i.e. situations in which no actions have yet occurred. Init(s) means that s is an initial situation. S_0 is taken to denote the actual initial situation.

We specify an agent's goals using a goal accessibility relation/fluent G. A path p is G-accessible in situation s, denoted by $G(p, s)$, iff the goals of the agent in s are satisfied over this path and if it starts with a situation that has the same action history as s. The latter requirement follows from our assumption that the agent is aware of the actions that have already occurred and that her goals should be compatible with this knowledge.

We say that an agent has the goal that ϕ in situation s iff ϕ holds over all paths that are G-accessible in s:

$$\text{Goal}(\phi, s) \doteq \forall p.\ G(p, s) \supset \phi[\![p]\!].$$

A domain theory for our framework includes the axioms of a theory $\mathcal{D}_{bat}$, which is as in Section 2, but with a modified set of foundational axioms Σ that captures multiple initial situations (Shapiro 2005), the axiomatization of paths and CTL*, i.e. $\mathcal{D}_{path}$ and $\mathcal{D}_{CTL^*}$, domain dependent initial goal axioms and the domain independent axioms 27-28 (see below). Typically the modeler provides some specification of what paths are G accessible initially. We call these axioms *initial goal axioms*. This can be done by providing an axiom as in the example below. Nevertheless, the agent's goals can be incompletely specified, i.e. the theory might not specify fully what the goals are initially.

Returning to our example of Section 4, assume that our mine-sweeper agent initially has the goal ϕ_1 to eventually visit cell (1,1), i.e. $\phi_1 = \Diamond\text{Visited}(1, 1)$; then her initial goals can be specified using the following initial goal axiom:

$$\text{Init}(s) \supset \forall p.\ (G(p, s) \equiv \text{Starts}(p, s') \wedge \text{Init}(s') \wedge \phi_1[\![p]\!]).$$

This specifies the goals ϕ_1 of the agent in the initial situations by making $G(p, s)$ true for every path p that starts with an initial situation and over which ϕ_1 holds.

Goal Dynamics: An agent's goals change as a result of the occurrence of an action (including exogenous events), or when she adopts or drops a goal. Here, we only discuss how goals change as a result of a regular action as well as that of an $adopt(\phi)$ action, i.e. the agent adopting ϕ as a goal.[3] We assume that agents are aware of all actions that occur. The action precondition axiom for $adopt()$ is as follows:

Axiom 27.

$$\text{Poss}(adopt(\phi), s) \equiv \neg\text{Goal}(\phi, s).$$

That is, an agent can adopt the goal that ϕ in situation s if she does not already have ϕ as her goal in s.

In the following, we specify the dynamics of goals by giving the successor-state axiom (SSA) for G:

Axiom 28 (SSA for G)**.**

$$\begin{aligned} &G(p, do(a, s)) \equiv \\ &\quad \exists p', s'.\ G(p', s) \wedge \text{Starts}(p', s') \wedge \text{Suffix}(p, p', do(a, s')) \\ &\quad\ \wedge a = adopt(\phi) \supset \phi[\![p]\!]. \end{aligned}$$

The overall idea of the SSA for G is as follows. To handle the occurrence of a non-adopt (i.e. regular) action a, we progress all G-accessible paths to reflect the fact that this action has just happened; this is done by replacing each G-accessible path p' in s with starting situation s' by its suffix p, provided that it starts with $do(a, s')$. Any path over which the next action performed is not a is eliminated from the G-accessibility relation.

Moreover, to handle the adoption of a goal ϕ, we progress the set of G-accessible paths as above and then we eliminate from this set all paths over which ϕ does not hold. Thus the agent acquires the goal that ϕ after she performs the $adopt(\phi)$ action.

In our example, it can be shown that the agent will have the goal that $\Diamond\text{Solved}()$ after she adopts the goal to eventually solve the problem in S_0; let $\mathcal{D}^{goal}_{swip}$ includes $\mathcal{D}_{swip}$ as in Section 4 (with modified Σ), Axioms 27 and 28, and the above initial goal axiom; then we have:[4]

$$\mathcal{D}^{goal}_{swip} \models \text{Goal}(\Diamond\text{Solved}(), do(adopt(\Diamond\text{Solved}()), S_0)).$$

This is because when the agent adopts the goal that $\Diamond\text{Solved}()$, all her G-accessible paths in S_0 will be progressed, i.e. these paths will be replaced by their "immediate" suffixes and any such path over which the next action performed is not $adopt(\Diamond\text{Solved}())$ will be dropped from the new G-relation. This is to reflect that this adopt action has happened. Moreover, any path over which the

[3]Since the action $adopt(\phi)$ takes a formula as argument, we must encode formulae as terms as in (De Giacomo, Lespérance, and Levesque 2000). For notational simplicity, we suppress this encoding and use formulas as terms directly.

[4]Note that, this holds even if there is a mine in cell (1,1) since in that case the agent's set of G-accessible paths in $do(adopt(\Diamond\text{Solved}()), S_0)$ will become empty and thus the result trivially follows; see (Khan and Lespérance 2010) to see how one can properly deal with such inconsistent goals.

agent does not eventually solve the problem will be eliminated from the G-relation in $do(adopt(\Diamond\text{Solved}()), S_0)$. Thus the agent acquires the goal that $\Diamond$Solved(). Note that from the given theory and the assumption that cell (1,1) is free of mines, it can be shown that there are no paths in $do(adopt(\Diamond\text{Solved}()), S_0)$ over which both $\Diamond$Solved() and $\Diamond$Dead() hold, and thus the agent will also acquire the goal that $\Box\neg$Dead() (as well as all other consequences of the adopted goal) after she adopts the goal that $\Diamond$Solved():

$$\mathcal{D}^{goal}_{swip} \cup \{\neg\text{Mine}(1,1)\} \models \\ \text{Goal}(\Box\neg\text{Dead}(), do(adopt(\Diamond\text{Solved}()), S_0)).$$

In this section, we used a simplified framework to show that infinite paths in the situation calculus enable a very straightforward and intuitive model of future-looking attitudes like goals and desires. Again, see (Khan and Lespérance 2009; 2010) for a much enhanced account of goals that handles the dropping of a goal ϕ as well as the adoption of a subgoal ψ w.r.t a parent goal ϕ and deals with (possibly inconsistent) prioritized goals and various grades of commitments towards goals, e.g. desires, intentions, etc. It also discusses agents' knowledge and its relationship to their goals, in particular how agents' knowledge can be used to filter out their "realistic goals" from their set of desires, what kind of constraints are required to manage the rational balance needed among these mental attitudes, etc.

7. Application: Non-terminating Programs

As a second application, in this section we show how paths in the situation calculus can be used to represent the executions of non-terminating high-level programs and then check that these satisfy given temporal properties. We consider programs in a subset of the situation calculus-based programming language ConGolog (De Giacomo, Lespérance, and Levesque 2000).

We start by outlining the ConGolog programming language, which is developed on top of situation calculus action theories to express and reason about complex processes that exhibit semantically rich agent behaviors. Here we focus on a subset of ConGolog that allows programs to be composed according to the following grammar:[5]

$$\delta ::= \alpha \mid \varphi? \mid (\delta_1;\delta_2) \mid (\delta_1 \mid \delta_2) \mid (\pi x.\ \delta) \mid (\delta_1 \| \delta_2) \mid \delta^*$$

Here α denotes a situation calculus primitive action, φ is a situation-suppressed situation calculus formula, and δ, δ_1, and δ_2 stand for complex actions. Thus programs can be constructed of primitive actions α, tests for conditions $\varphi?$, sequences $\delta_1;\delta_2$, nondeterministic branch $\delta_1 \mid \delta_2$, nondeterministic choices of arguments $\pi x.\ \delta$, concurrent executions of programs (understood as interleavings) $\delta_1 \| \delta_2$, and nondeterministic iteration δ^*, which performs δ 0 or more times. $\pi x.\delta$ nondeterministically picks a legal binding for the variable x and performs the program δ for this binding of x. Other convenient control structures such as conditionals and while loops can be introduced as abbreviations, e.g. **while** φ **do** δ **endWhile** can be defined as $(\varphi?;\delta)^*;\neg\varphi?$. As an example, consider the simple program to clear a table in a blocks world: $\pi b.\ (\text{OnTable}(b)?; putAway(b))^*; \neg\exists b.\ \text{OnTable}(b)?$; this repeatedly picks a block that is on the table and puts it away until the table is clear.

The semantics of ConGolog programs is defined in terms of single-step transitions using two special predicates $Final$ and $Trans$, where $Final(\delta, s)$ means that program δ may legally terminate in situation s, and where $Trans(\delta, s, \delta', s')$ means that program δ in situation s may legally execute one step, ending in situation s' with program δ' remaining. $Trans$ and $Final$ are inductively characterized by giving equivalence axioms for each construct, such as: $Trans((\delta_1 \mid \delta_2), s, \delta', s') \equiv Trans(\delta_1, s, \delta', s') \vee Trans(\delta_2, s, \delta', s')$. Complete executions of a *terminating* ConGolog program are specified by the Do predicate, which is defined as follows:

$$Do(\delta, s, s') \doteq \exists \delta'.\ (Trans^*(\delta, s, \delta', s') \wedge Final(\delta', s')),$$

where $Trans^*$ is the reflexive transitive closure of $Trans$. Thus $Do(\delta, s, s')$ holds if and only if s' can be reached by performing a sequence of transitions starting with the program δ in s, and the remaining program δ' may legally terminate in s'.

We can use paths to give a characterization of complete executions of non-terminating programs. We use $Do(\delta, s, p)$ to mean that the path p can be produced by a complete execution of the non-terminating ConGolog program δ starting from situation s:

$$Do(\delta, s, p) \doteq \\ \text{Starts}(p, s) \wedge \forall s'.\ \text{OnPath}(p, s') \supset \exists \delta'.\ Trans^*(\delta, s, \delta', s').$$

Thus $Do(\delta, s, p)$ holds if path p starts with s and any situation s' on p can be reached from s by executing δ starting in s with some program δ' remaining to be executed. Note that this implies that for any situation s' on p, there is a sequence of transitions (involving a sequence of intermediate programs) from program δ in s to a program δ' in s'. Thus all situations in p are produced by an execution of δ starting in s. Also, any terminating program can easily be transformed into a non-terminating one that loops over a "completed" state. Thus our Do generalizes the original one.

Given this characterization of the complete executions of non-terminating programs, we can use our temporal logic encoding to express/verify properties over executions of such programs. All executions of a non-terminating program δ starting in situation s satisfy an LTL property ϕ if and only if we have:

$$\forall p.\ Do(\delta, s, p) \supset \phi[\![p]\!].$$

LTL properties are path formulae that do not contain any path quantifiers. Note that we cannot verify general CTL* properties in this way, as the path quantifiers in our CTL* encoding quantify over the executable paths in the situation tree, not the execution paths of the program.[6]

[5] Other ConGolog constructs include concurrency with different priorities, interrupt calls, concurrent iterations, procedure calls, etc.

[6] For so called situation-determined ConGolog programs, there is a unique remaining program for each situation that is reachable in an execution of the program. As shown in (De Giacomo et al. 2016), it is possible to compile such a program into the action the-

Also given the above definition of Do, one can easily model what it means for an agent to have a *procedural goal*, i.e. a goal to execute a program. An agent having the goal in situation s to execute a (non-terminating) program δ starting in s, Goal$(Do(\delta), s)$, can be defined as: $\forall p.\ G(p, s) \supset Do(\delta, s, p)$, i.e. we require G-accessible paths to be produced by an execution of δ.

8. Discussion

We have already discussed the work most closely related to ours in Section 1. Beyond this, there is some work that deals with the temporal aspects of situations, i.e. the starting time of situations and action durations (Pinto 1994; Reiter 1996), but not temporally extended paths. Another set of approaches introduces some notion of paths while addressing some application of paths and shows how various temporal logic formulae can be interpreted over such paths. Gabaldon (2004) was the first to introduce statements of temporal logic (LTL) into the situation calculus. He used these to express search control knowledge for forward-chaining planning. However, he only considers finite paths defined by pairs of situations. Fritz and McIlraith (2006) show how an extended version of LTL interpreted over a finite horizon can be compiled into ConGolog. (Claßen and Lakemeyer 2008) develops a second-order modal logic inspired by CTL* and dynamic logic to express properties about (possibly) non-terminating ConGolog programs. The authors define infinite "traces" using program configurations. A configuration is a pair (δ, z), where δ is a ConGolog program that remains to be executed and z is a sequence of actions that have been already performed. Given z and world w, they define infinite execution traces of δ as infinite sequences of configurations, s.t. the ending configuration of any finite prefix of the sequence can be reached from the initial configuration (δ, z) and w. Note that, a key difference between this work and our formalization is that while we define program semantics and paths axiomatically in the situation calculus, they define a modal logic on top of the situation calculus that allows temporal properties over execution of programs to be expressed and the semantics of programs is part of the model theoretic semantics of the logic. For the CTL-like fragment of the language, the authors also propose a verification method based on fixpoint approximation and "characteristic graphs", which can finitely represent a ConGolog program's configuration graph; the method is sound but incomplete. (De Giacomo, Ternovskaia, and Reiter 1997) uses a first-order version of the μ-calculus (Emerson 1996) to specify properties of non-terminating Golog (Levesque et al. 1997) programs. The μ-calculus is a very expressive temporal logic that provides least and greatest fixpoints. It subsumes CTL*, but translating a CTL* formula often results in a much less readable μ-calculus formula.

ory so that actions are only possible if they can be performed by the program, and then one can use the resulting theory to verify CTL* properties of the program. In general however, CTL* properties of a program must be evaluated over the program's transition system, where nodes are program configurations consisting of a remaining program and a situation. One cannot use paths in the situation tree to do this.

To summarize, our main contribution in this paper is twofold: first, we introduced infinite paths in the situation calculus by providing a sound and complete axiomatization for infinite paths; and second, we proved some desirable properties and showed that infinite paths in the situation calculus indeed correspond to an intuitive notion of paths. To the best of our knowledge, ours is the only work that introduces infinite paths as a sort in the language of the situation calculus and proves interesting properties. To complement this main result, we showed how CTL* formulae can be interpreted in the situation calculus with paths. Also, we discussed two applications involving temporally extended goals and non-terminating programs that illustrate the usefulness of infinite paths.

In the future, we would like to utilize paths in the situation calculus to develop/further refine various applications involving infinite histories, such as verification of temporal properties of non-terminating programs and modeling of agents' beliefs about the future, etc. Also, it would be interesting to incorporate program execution paths explicitly in the language of situation calculus. Introducing a notion of program execution path, i.e. sequence of program configurations (situation and remaining program) would allow state and path formulae to be evaluated w.r.t. the tree of configurations induced by a given program and starting situation, and this would in turn allow the verification of CTL* properties over arbitrary programs within the situation calculus.

Acknowledgments

We would like to thank Mikhail Soutchanski for helpful comments on an earlier version of this paper. This work was supported in part by the National Science and Engineering Research Council of Canada under a Discovery grant for the project Specification, Verification, and Synthesis of Autonomous Adaptive Agents.

Appendix: Proofs

The following Lemmata are required for some of the proofs. Σ entails that doing an action yields a different situation:

Lemma 29.

$$\Sigma \models \forall a, s.\ s \neq do(a, s).$$

Proof. See Proposition 2.4.1 in (Shapiro 2005). □

Σ entails that a situation s strictly precedes the situation that results from doing an action in s:

Lemma 30.

$$\Sigma \models \forall a, s.\ s \prec do(a, s).$$

Proof. See Proposition 2.4.2 in (Shapiro 2005). □

Σ entails that if a situation strictly precedes another situation, then they are different:

Lemma 31.

$$\Sigma \models \forall s, s'.\ s \prec s' \supset s \neq s'.$$

Proof. (By contradiction) Fix situations S_1 and S_2 and assume that $S_1 \prec S_2$ and $S_1 = S_2$. If we substitute S_1 for S_2 in the former, we have $S_1 \prec S_1$, but this is contradictory to the irreflexivity of $\prec$. $\square$

Σ entails that the result of doing a in s does not precede s:

Lemma 32.

$$\Sigma \models \forall a, s.\ do(a, s) \not\prec s.$$

Proof. See Proposition 2.4.9 in (Shapiro 2005). $\square$

Σ entails that a situation s precedes doing an action in s:

Lemma 33.

$$\Sigma \models \forall a, s.\ s \preceq do(a, s).$$

Proof. See Corollary 2.4.3 in (Shapiro 2005). $\square$

Σ entails that if $do(a, s)$ is executable, then it is possible to execute a in s, and s is executable:

Lemma 34.

$$\Sigma \models \forall a, s.\ \text{Executable}(do(a, s)) \supset \text{Poss}(a, s) \wedge \text{Executable}(s).$$

Proof. See Proposition 2.4.16 in (Shapiro 2005). $\square$

Σ entails that if two situations $do(a, s)$ and $do(b, s)$ obtained by performing two actions a and b in the same situation s each precedes a third situation s', then a and b represent the same action.

Lemma 35.

$$\Sigma \models \forall a, b, s, s'.\ (do(a, s) \preceq s' \wedge do(b, s) \preceq s') \supset a = b.$$

Proof. (By induction on s') For the base case, fix A_1, B_1, S_1, and S_1', and assume that:

$$do(A_1, S_1) = S_1',$$
$$do(B_1, S_1) = S_1'.$$

From this, we have:

$$do(A_1, S_1) = do(B_1, S_1).$$

From this and the injectivity of $do(\cdot, \cdot)$, we have $A_1 = B_1$.

For the inductive hypothesis, fix S_1'' and assume that:

$$(do(A_1, S_1) \preceq S_1'' \wedge do(B_1, S_1) \preceq S_1'') \supset A_1 = B_1. \quad (1)$$

Fix action C_1. We have to show that:

$$(do(A_1, S_1) \preceq do(C_1, S_1'') \wedge do(B_1, S_1) \preceq do(C_1, S_1'')) \supset A_1 = B_1.$$

Assume that $(do(A_1, S_1) \preceq do(C_1, S_1'') \wedge do(B_1, S_1) \preceq do(C_1, S_1''))$. Then we have 4 cases to consider.

Case 1. $do(A_1, S_1) = do(C_1, S_1'')$ and $do(B_1, S_1) = do(C_1, S_1'')$. In this case, by the injectivity of $do(\cdot, \cdot)$, we have $A_1 = C_1$ and $B_1 = C_1$, and thus $A_1 = B_1$.

Case 2. $do(A_1, S_1) = do(C_1, S_1'')$ and $do(B_1, S_1) \prec do(C_1, S_1'')$. From the former and the injectivity of $do(\cdot, \cdot)$, we have $S_1 = S_1''$. If we substitute S_1 for S_1'' in the latter, we have $do(B_1, S_1) \prec do(C_1, S_1)$. From this, the definition of $\prec$, and the fact that do is a function, we have $do(B_1, S_1) \preceq S_1$; but by the definition of $\preceq$ and Lemmata 29 and 32, this is impossible.

Case 3. $do(A_1, S_1) \prec do(C_1, S_1'')$ and $do(B_1, S_1) = do(C_1, S_1'')$. As in Case 2, this case is also not possible.

Case 4. $do(A_1, S_1) \prec do(C_1, S_1'')$ and $do(B_1, S_1) \prec do(C_1, S_1'')$. From these, the definition of $\prec$, and the fact that do is a function, we have:

$$do(A_1, S_1) \preceq S_1'' \wedge do(B_1, S_1) \preceq S_1''.$$

Thus we can apply the inductive hypothesis (1), which gives us $A_1 = B_1$. $\square$

Assume that s_1 and s_2 are successors of two different situations, obtained by performing two different actions a and b, respectively, in the same situation s. Σ entails that s_1 and s_2 are not co-linear.

Lemma 36.

$$\Sigma \models \forall s, s_1, s_2, a, b.\ a \neq b \wedge do(a, s) \preceq s_1 \wedge do(b, s) \preceq s_2 \supset s_1 \neq s_2 \wedge s_1 \not\prec s_2 \wedge s_2 \not\prec s_1.$$

Proof. (By contradiction) Fix A_1, B_1, S_1, S_1^1, and S_1^2 and assume that:

$$A_1 \neq B_1, \quad (2)$$
$$do(A_1, S_1) \preceq S_1^1 \wedge do(B_1, S_1) \preceq S_1^2, \text{ and} \quad (3)$$
$$S_1^1 = S_1^2 \vee S_1^1 \prec S_1^2 \vee S_1^2 \prec S_1^1. \quad (4)$$

Now, (3) above gives us the following four cases:

Case 1. Assume that:

$$S_1^1 = do(A_1, S_1), \text{ and} \quad (5)$$

$$S_1^2 = do(B_1, S_1). \quad (6)$$

From these, (2), and injectivity of $do(\cdot, \cdot)$, we have:

$$S_1^1 \neq S_1^2. \quad (7)$$

Thus from (4) and (7), we have: $S_1^1 \prec S_1^2 \vee S_1^2 \prec S_1^1$. Assume that $S_1^1 \prec S_1^2$. Then from this, (5), and (6), we have: $do(A_1, S_1) \prec do(B_1, S_1)$. From this, the definition of $\prec$, and the fact that do is a function, we have: $do(A_1, S_1) \preceq S_1$. From this and the definition of $\preceq$, it follows that either $do(A_1, S_1) = S_1$ or $do(A_1, S_1) \prec S_1$. But by Lemma 29, the former is impossible. By Lemma 32, the latter is also impossible. It thus follows that:

$$S_1^1 \not\prec S_1^2. \quad (8)$$

Similarly, it can be shown that:

$$S_1^2 \not\prec S_1^1. \quad (9)$$

But (7), (8), and (9) is contradictory to (4).

Case 2a. Assume that:

$$S_1^1 = do(A_1, S_1), \text{ and} \quad (10)$$

$$do(B_1, S_1) \prec S_1^2. \quad (11)$$

I will show that $\neg(S_1^1 = S_1^2 \vee S_1^1 \prec S_1^2 \vee S_1^2 \prec S_1^1)$ by going over each case, one at a time. First, suppose that $S_1^1 =$

S_1^2. From this and (10), we have $S_1^2 = do(A_1, S_1)$. From this and (11), we have $do(B_1, S_1) \prec do(A_1, S_1)$. From this, the definition of $\prec$, and the fact that do is a function, we have $do(B_1, S_1) \preceq S_1$. But again, this is impossible by the definition of $\preceq$ and Lemmata 29 and 32. Thus we have:

$$S_1^1 \neq S_1^2. \tag{12}$$

Now suppose that $S_1^1 \prec S_1^2$. Then from this and (10), we have:

$$do(A_1, S_1) \prec S_1^2.$$

From this, (11), and the definition of $\preceq$, we have:

$$do(A_1, S_1) \preceq S_1^2 \wedge do(B_1, S_1) \preceq S_1^2.$$

From this and Lemma 35, we have $A_1 = B_1$, which is contradictory to (2). Thus we have:

$$S_1^1 \not\prec S_1^2. \tag{13}$$

Finally, suppose that $S_1^2 \prec S_1^1$. Then from this and (10), we have $S_1^2 \prec do(A_1, S_1)$. From (11), this, and transitivity of $\prec$, we have $do(B_1, S_1) \prec do(A_1, S_1)$. But as shown above, this is impossible. Thus:

$$S_1^2 \not\prec S_1^1. \tag{14}$$

But (12), (13), and (14) is contradictory to (4).
Case 2b. Assume that:

$$do(A_1, S_1) \prec S_1^1, \text{ and} \tag{15}$$
$$S_1^2 = do(B_1, S_1). \tag{16}$$

The proof for this case is similar to that of Case 2a.
Case 3. Assume that:

$$do(A_1, S_1) \prec S_1^1, \text{ and} \tag{17}$$
$$do(B_1, S_1) \prec S_1^2. \tag{18}$$

Again, we will show that $\neg(S_1^1 = S_1^2 \vee S_1^1 \prec S_1^2 \vee S_1^2 \prec S_1^1)$ by going over each case separately. First, assume that $S_1^1 = S_1^2$. From this and (18), we have $do(B_1, S_1) \prec S_1^1$. From this and (17), we have:

$$do(A_1, S_1) \prec S_1^1 \wedge do(B_1, S_1) \prec S_1^1.$$

From this and the definition of $\preceq$, we have:

$$do(A_1, S_1) \preceq S_1^1 \wedge do(B_1, S_1) \preceq S_1^1.$$

From this and Lemma 35, we have $A_1 = B_1$, which is contradictory to (2). Thus we have:

$$S_1^2 \neq S_1^1. \tag{19}$$

Next, assume that $S_1^1 \prec S_1^2$. Then by this, (17), and transitivity of $\prec$, we have $do(A_1, S_1) \prec S_1^2$. From this, (18), and the definition of $\preceq$, we have:

$$do(A_1, S_1) \preceq S_1^2 \wedge do(B_1, S_1) \preceq S_1^2.$$

From this and Lemma 35, we have $A_1 = B_1$, which is contradictory to (2). Thus we have:

$$S_1^1 \not\prec S_1^2. \tag{20}$$

Finally, assume that $S_1^2 \prec S_1^1$. The proof for this case is similar to the above. Hence we have:

$$S_1^2 \not\prec S_1^1. \tag{21}$$

But (19), (20), and (21) is contradictory to (4). □

If two situations are both preceded by a third situation and they are not co-linear, then there must be two different situations that precede them, and these situations can be obtained by performing two different actions in the same situation.

Lemma 37.

$$\begin{aligned}\Sigma \models \forall s, s_1, s_2.\ (&s \preceq s_1 \wedge s \preceq s_2 \\ &\wedge \neg(s_1 = s_2 \vee s_1 \prec s_2 \vee s_2 \prec s_1)) \supset \\ &(\exists s', a_1, a_2.\ s \preceq s' \wedge do(a_1, s') \preceq s_1 \\ &\wedge do(a_2, s') \preceq s_2 \wedge a_1 \neq a_2).\end{aligned}$$

Proof Sketch. Fix S_1, S_1^1, and S_1^2, and assume that:

$$S_1 \preceq S_1^1, \tag{22}$$
$$S_1 \preceq S_1^2, \text{ and} \tag{23}$$
$$\neg(S_1^1 = S_1^2 \vee S_1^1 \prec S_1^2 \vee S_1^2 \prec S_1^1). \tag{24}$$

Now (22) and (23) above give us 4 cases.
Case 1. Assume that $S_1 = S_1^1$ and $S_1 = S_1^2$. Then we have $S_1^1 = S_1^2$; but then this case is ruled out by (24).
Case 2. Assume that $S_1 = S_1^1$ and $S_1 \prec S_1^2$. Then we have $S_1^1 \prec S_1^2$; but then this case too is ruled out by (24).
Case 3. Assume that $S_1 \prec S_1^1$ and $S_1 = S_1^2$. Then we have $S_1^2 \prec S_1^1$; again by (24), this is also impossible.
Case 4. Assume that:

$$S_1 \prec S_1^1, \text{ and} \tag{25}$$
$$S_1 \prec S_1^2. \tag{26}$$

Consider the path from S_1 to S_1^1: there must be a situation s' such that $S_1 \prec s' \preceq S_1^1$ and $\neg(s' \preceq S_1^2)$, otherwise S_1^1 and S_1^2 are colinear, contradicting (24). Let S' be the unique situation such that:

$$S_1 \prec S' \preceq S_1^1, \tag{27}$$
$$\neg(S' \preceq S_1^2), \text{ and} \tag{28}$$
$$\forall s^*.\ S_1 \preceq s^* \prec S' \supset s^* \prec S_1^2. \tag{29}$$

From (27), we have $S_1 \prec S'$. From this and Definition of $\prec$, it follows that there is an action A_1 and situation S'' such that:

$$S' = do(A_1, S''), \text{ and} \tag{30}$$
$$S_1 \preceq S''. \tag{31}$$

From (30) and (27), we have:

$$do(A_1, S'') \preceq S_1^1. \tag{32}$$

From (31), (30), and Definition of $\prec$, we have: $S_1 \preceq S'' \prec S'$. By this and (29), we have: $S'' \prec S_1^2$. From this, (30), and (28), it follows that there exists an action A_2 such that:

$$A_1 \neq A_2, \text{ and} \tag{33}$$
$$do(A_2, S'') \preceq S_1^2. \tag{34}$$

The consequent thus follows from (31), (32), (34), and (33). □

Proposition 9.

$$\begin{aligned}\Sigma \cup \mathcal{D}_{path} \models\ &(\forall s'.\ \exists a.\ \text{Poss}(a, s')) \supset \\ &(\forall s.\ \text{Executable}(s) \supset \exists p.\ \text{Starts}(p, s)).\end{aligned}$$

Proof. Fix situation S_1 and assume that $\forall s.\ \exists a.\ \mathrm{Poss}(a,s)$ and Executable(S_1). Construct an action selection function F_1 as follows: $F_1(s) = a,$ for any situation s, where a is an arbitrary action that is executable in s, i.e. Poss(a,s); by the antecedent, such an action is always available. Then by the antecedent, Definitions 4 and 3, and by construction of F_1, we have: Executable(F_1, S_1). The consequent follows from this and Axiom 2(b). □

Proposition 10.

(a). $\Sigma \cup \mathcal{D}_{path} \models \forall p.\ \exists s.\ \mathrm{Starts}(p,s),$
(b). $\Sigma \cup \mathcal{D}_{path} \models \forall p, s, s'.\ \mathrm{Starts}(p,s) \wedge \mathrm{Starts}(p,s') \supset s = s',$
(c). $\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ \mathrm{Starts}(p,s) \supset \mathrm{Executable}(s).$

Proof. **(a).** Fix path P_1. By Axiom $2(a)$, there is a corresponding function F_1 and situation S_1 such that:

$$\forall s.\ \mathrm{OnPath}(P_1,s) \equiv \mathrm{OnPathASF}(F_1,S_1,s).$$

From this and Definition 3, it follows that:.

$$\forall s.\ \mathrm{OnPath}(P_1,s) \equiv S_1 \preceq s \wedge \forall a, s^*.\ S_1 \prec do(a,s^*) \preceq s \supset F_1(s^*) = a.$$

From this and Definition of $\preceq$, we have: OnPath(P_1,S_1) and $\forall s.\ \mathrm{OnPath}(P_1,s) \supset S_1 \preceq s$. From these and Definition 1, it follows that Starts(P_1,S_1).
(b). Fix path P_1 and starting situations S_1 and S_1'. By the antecedent, we have Starts(P_1,S_1). From this and Definition 1, we have:

$$\mathrm{OnPath}(P_1,S_1), \quad (35)$$

$$\forall s.\ \mathrm{OnPath}(P_1,s) \supset S_1 \preceq s. \quad (36)$$

Again, from the antecedent, we have Starts(P_1,S_1'). From this and Definition 1, we have:

$$\mathrm{OnPath}(P_1,S_1'), \quad (37)$$

$$\forall s.\ \mathrm{OnPath}(P_1,s) \supset S_1' \preceq s. \quad (38)$$

From (36) and (37), we have: $S_1 \preceq S_1'$, and from (35) and (38), we have: $S_1' \preceq S_1$. The consequent follows from these and antisymmety of $\preceq$.
(c). Fix path P_1. By Axiom $2(a)$, there is a corresponding function F_1 and situation S_1 such that: Executable(F_1,S_1) and $\forall s.\ \mathrm{OnPath}(P_1,s) \equiv \mathrm{OnPathASF}(F_1,S_1,s)$. As in the proof of Proposition 10(a), from the latter and Definitions 1 and 3, it follows that Starts(P_1,S_1). Finally, from Executable(F_1,S_1) and Definition 4, it follows that Executable(S_1). □

Proposition 11.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ \mathrm{OnPath}(p,s) \supset \exists s', a.\ \mathrm{OnPath}(p,s') \wedge s' = do(a,s) \wedge \mathrm{Poss}(a,s).$$

Proof. Fix path P_1. By Axiom $2(a)$, there is a corresponding function F_1 and situation S_1 such that:

$$\mathrm{Executable}(F_1,S_1), \quad (39)$$

$$\forall s.\ \mathrm{OnPath}(P_1,s) \equiv \mathrm{OnPathASF}(F_1,S_1,s). \quad (40)$$

Consider any situation S_n on P_1, i.e., OnPath(P_1,S_n). By (40), S_n must also be on the sequence defined by (S_1,F_1):

$$\mathrm{OnPathASF}(F_1,S_1,S_n). \quad (41)$$

From (41) and Definition 3, we have:

$$S_1 \preceq S_n, \text{ and} \quad (42)$$

$$\forall a, s.\ S_1 \prec do(a,s) \preceq S_n, \supset F_1(s) = a. \quad (43)$$

Assume $F_1(S_n) = A_n$. Then from (42), transitivity of $\preceq$, and Lemma 33, we have:

$$S_1 \preceq do(A_n,S_n). \quad (44)$$

From (44), (43), the assumption that $F_1(S_n) = A_n$, and Definition 3, it follows that the situation $do(A_n,S_n)$ must be on the sequence defined by (S_1,F_1):

$$\mathrm{OnPathASF}(F_1,S_1,do(A_n,S_n)). \quad (45)$$

Also, by (45) and (40), $do(A_n,S_n)$ must be on path P_1:

$$\mathrm{OnPath}(P_1,do(A_n,S_n)). \quad (46)$$

Finally, by (39), (45), and Definition 4, action A_n must have been executable in S_n:

$$\mathrm{Poss}(A_n,S_n). \quad (47)$$

The proposition follows from (46) and (47). □

Proposition 12.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ [\mathrm{OnPath}(p,s) \wedge \mathrm{OnPath}(p,do(a,s)) \wedge \mathrm{OnPath}(p,do(b,s))] \supset a = b.$$

Proof. (By contradiction) Fix path P_1. By Axiom $2(a)$, there is a corresponding function F_1 and situation S_1 such that:

$$\forall s.\ \mathrm{OnPath}(P_1,s) \equiv \mathrm{OnPathASF}(F_1,S_1,s). \quad (48)$$

Fix S_1^1, A_1, and A_2 and assume that:

$$\mathrm{OnPath}(P_1,do(A_1,S_1^1)), \quad (49)$$

$$\mathrm{OnPath}(P_1,do(A_2,S_1^1)), \text{ and} \quad (50)$$

$$A_1 \neq A_2. \quad (51)$$

From (48), (49), and Definition 3, it follows that:

$$\forall a, s^*.\ S_1 \prec do(a,s^*) \preceq do(A_1,S_1^1) \supset F_1(s^*) = a. \quad (52)$$

Similarly, from (48), (50), and Definition 3, it follows that:

$$\forall a, s^*.\ S_1 \prec do(a,s^*) \preceq do(A_2,S_1^1) \supset F_1(s^*) = a. \quad (53)$$

But from (52), (53), and (51), we have that:

$$F_1(S_1^1) = A_1 \wedge F_1(S_1^1) = A_2 \wedge A_1 \neq A_2,$$

which is contradictory to the fact that F_1 is a function. □

Proposition 13.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, p'.\ p \neq p' \supset \exists s.\ (\mathrm{OnPath}(p,s) \wedge \neg\mathrm{OnPath}(p',s)).$$

Proof. Follows from Axiom 5. □

Corollary 14.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ \text{OnPath}(p, s) \supset \text{Executable}(s).$$

Proof. (By induction on s) Fix path P_1. The base case follows from Propositions 10(b) and 10(c). For the inductive hypothesis, fix situation S_1 and assume that:

$$\text{OnPath}(P_1, S_1), \text{ and} \tag{54}$$

$$\text{Executable}(S_1). \tag{55}$$

From this and Propositions 11 and 12, it follows that there is a unique successor situation S_2 and action A_1 such that:

$$\text{OnPath}(P_1, S_2) \wedge S_2 = do(A_1, S_1), \text{ and} \tag{56}$$

$$\text{Poss}(A_1, S_1). \tag{57}$$

From (55), (57), and definition of Executable$(\cdot)$, it follows that Executable$(do(A_1, S_1))$. □

Proposition 15.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s, s'.\ \text{OnPath}(p, s) \wedge \text{OnPath}(p, s') \supset s = s' \vee s \prec s' \vee s' \prec s.$$

Proof. (By contradiction) Fix path P_1. By Axiom $2(a)$, there is a corresponding function F_1 and situation S_1 such that:

$$\forall s.\ \text{OnPath}(P_1, s) \equiv \text{OnPathASF}(F_1, S_1, s). \tag{58}$$

Fix S_m and S_n and assume that:

$$\text{OnPath}(P_1, S_m), \tag{59}$$

$$\text{OnPath}(P_1, S_n), \tag{60}$$

$$\neg(S_m = S_n \vee S_m \prec S_n \vee S_n \prec S_m). \tag{61}$$

From (58), (59), and Definition 3,it follows that:

$$S_1 \preceq S_m, \text{ and} \tag{62}$$

$$\forall a, s^*.\ S_1 \prec do(a, s^*) \preceq S_m \supset F_1(s^*) = a. \tag{63}$$

Similarly, from (58), (60), and Definition 3, it follows that:

$$S_1 \preceq S_n, \text{ and} \tag{64}$$

$$\forall a, s^*.\ S_1 \prec do(a, s^*) \preceq S_n \supset F_1(s^*) = a. \tag{65}$$

From (61), (62), (64), and Lemma 37, it follows that there is a situation S_2 and actions A_1 and A_2 such that:

$$S_1 \preceq S_2 \wedge do(A_1, S_2) \preceq S_m \wedge do(A_2, S_2) \preceq S_n \wedge A_1 \neq A_2. \tag{66}$$

But from (63), (65), and (66), we have that:

$$F_1(S_2) = A_1 \wedge F_1(S_2) = A_2 \wedge A_1 \neq A_2,$$

which is contradictory to the fact that F_1 is a function. □

Proposition 16.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s, s', s^*.\ (\text{OnPath}(p, s) \wedge \text{OnPath}(p, s') \wedge s \preceq s^* \preceq s') \supset \text{OnPath}(p, s^*).$$

Proof. Fix path P_1. By Axiom $2(a)$, there is a corresponding function F_1 and situation S_1 such that:

$$\forall s.\ \text{OnPath}(P_1, s) \equiv \text{OnPathASF}(F_1, S_1, s). \tag{67}$$

Fix S_m, S_n, S_p and assume that:

$$\text{OnPath}(P_1, S_m), \tag{68}$$

$$\text{OnPath}(P_1, S_p), \tag{69}$$

$$S_m \preceq S_n \preceq S_p. \tag{70}$$

From (67) and (68), it follows that:

$$\text{OnPathASF}(F_1, S_1, S_m). \tag{71}$$

Similarly, from (67) and (69), it follows that:

$$\text{OnPathASF}(F_1, S_1, S_p). \tag{72}$$

From (71) and Definition 3, it follows that:

$$S_1 \preceq S_m. \tag{73}$$

From this, (70), and transitivity of $\preceq$, it follows that:

$$S_1 \preceq S_n. \tag{74}$$

From (72) and Definition 3, it follows that:

$$\forall a, s^*.\ S_1 \prec do(a, s^*) \preceq S_p \supset F_1(s^*) = a. \tag{75}$$

From this, (70), and (74), it follows that:

$$\forall a, s^*.\ S_1 \prec do(a, s^*) \preceq S_n \supset F_1(s^*) = a. \tag{76}$$

From (74), (76), and Definition 3, it follows that OnPathASF(F_1, S_1, S_n). The proposition follows from this and (67). □

Proposition 17.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p_1, p_2, s, a, b, s_1, s_2.\ [\text{OnPath}(p_1, do(a, s)) \wedge \text{OnPath}(p_2, do(b, s)) \wedge a \neq b \wedge s \prec s_1 \wedge s \prec s_2 \wedge \text{OnPath}(p_1, s_1) \wedge \text{OnPath}(p_2, s_2)] \supset s_1 \neq s_2.$$

Proof. (By contradiction) Fix P_1, P_2, S_1, A_1, B_1, S_{11}, and S_{12}, and assume that:

$$\text{OnPath}(P_1, do(A_1, S_1)), \tag{77}$$

$$\text{OnPath}(P_2, do(B_1, S_1)), \tag{78}$$

$$A_1 \neq B_1, \tag{79}$$

$$S_1 \prec S_{11}, \tag{80}$$

$$S_1 \prec S_{12}, \tag{81}$$

$$\text{OnPath}(P_1, S_{11}), \text{ and} \tag{82}$$

$$\text{OnPath}(P_2, S_{12}). \tag{83}$$

Also, assume that the consequent is false:

$$S_{11} = S_{12}. \tag{84}$$

From (77), (82), Proposition 15, and definition of $\preceq$, it follows that:

$$do(A_1, S_1) \preceq S_{11} \vee S_{11} \preceq do(A_1, S_1). \tag{85}$$

Now suppose that:

$$S_{11} \prec do(A_1, S_1).$$

Then by Definition of $\prec$, we have:

$$\exists b, s.\ (do(A_1, S_1) = do(b, s) \land S_{11} \preceq s).$$

Then from this and injectivity of $do(\cdot,\cdot)$, it follows that $S_{11} \preceq S_1$. By (80) and Lemma 31, it follows that $S_{11} \neq S_1$. Thus by Definition of $\preceq$, we have:

$$S_{11} \prec S_1.$$

Since $\prec$ is asymmetric, it follows from this that $\neg(S_1 \prec S_{11})$. But this contradicts (80). Thus it follows that:

$$\neg(S_{11} \prec do(A_1, S_1)). \tag{86}$$

From (85) and (86), it follows that:

$$do(A_1, S_1) \preceq S_{11}. \tag{87}$$

Similarly, it can be shown that:

$$do(B_1, S_1) \preceq S_{12}. \tag{88}$$

Now from (84), we have $S_{11} = S_{12}$. But from (87), (88), (79), and Lemma 36, this is impossible. □

Proposition 18.

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, s.\ \text{OnPath}(p, s) \supset \exists p'.\ \text{Suffix}(p', p, s).$$

Proof. Fix path P_1. By Axiom $2(a)$, there is a function F_1 and situation S_1 such that:

$$\text{Executable}(F_1, S_1), \tag{89}$$

$$\forall s.\ \text{OnPath}(P_1, s) \equiv \text{OnPathASF}(F_1, S_1, s). \tag{90}$$

Fix situation S_n such that:

$$\text{OnPath}(P_1, S_n). \tag{91}$$

We will show that there is a path P_n s.t. P_n, that starts with S_n, is a suffix of P_1.

Consider the pair (S_n, F_1). From (90) and (91), we have that:

$$\text{OnPathASF}(F_1, S_1, S_n). \tag{92}$$

From this and Definition 3, it follows that:

$$S_1 \preceq S_n. \tag{93}$$

From (92), (93), (89), and the definitions of Executable$(\cdot,\cdot)$ and Executable$(\cdot)$, we have:

$$\text{Executable}(F_1, S_n). \tag{94}$$

By (94) and Axiom $2(b)$, it follows that there is a path P_n s.t.:

$$\text{Starts}(P_n, S_n), \text{ and} \tag{95}$$

$$\forall s.\ \text{OnPathASF}(F_1, S_n, s) \equiv \text{OnPath}(P_n, s). \tag{96}$$

Now, we need show that Suffix(P_n, P_1, S_n). From (90) and Definition 3, we have:

$$\begin{aligned}\forall s.\ \text{OnPath}(P_1, s) \equiv (&S_1 \preceq s \\ &\land \forall a, s^*.\ S_1 \prec do(a, s^*) \preceq s \supset F_1(s^*) = a).\end{aligned} \tag{97}$$

Similarly, from (96) and Definition 3, we have:

$$\begin{aligned}\forall s.\ \text{OnPath}(P_n, s) \equiv (&S_n \preceq s \\ &\land \forall a, s^*.\ S_n \prec do(a, s^*) \preceq s \supset F_1(s^*) = a).\end{aligned} \tag{98}$$

From (97), (98), (93), and (91), it follows that:

$$\forall s.\ S_n \preceq s \supset \text{OnPath}(P_1, s) \equiv \text{OnPath}(P_n, s). \tag{99}$$

Then Suffix(P_n, P_1, S_n) follows from (91), (95), (99), and Definition 6. □

Proposition 19.

$$\begin{aligned}\Sigma \cup \mathcal{D}_{path} \models\ &\text{Starts}(p, do(a, s)) \supset \\ &\exists p'.\ \text{Starts}(p', s) \land \text{Suffix}(p, p', do(a, s)).\end{aligned}$$

Proof. Fix P_1, A_1, and S_1 and assume that:

$$\text{Starts}(P_1, do(A_1, S_1)). \tag{100}$$

By Axiom $2(a)$, there is a function F_1 and situation S_2 such that:

$$\text{Executable}(F_1, S_2), \text{ and} \tag{101}$$

$$\forall s.\ \text{OnPath}(P_1, s) \equiv \text{OnPathASF}(F_1, S_2, s). \tag{102}$$

From (102), reflexivity of $\preceq$, and Definition 3, we have:

$$\text{OnPath}(P_1, S_2). \tag{103}$$

Again, from (102) and Definition 3, we have:

$$\forall s.\ \text{OnPath}(P_1, s) \supset S_2 \preceq s. \tag{104}$$

From (103), (104), and Definition 1, we have that:

$$\text{Starts}(P_1, S_2). \tag{105}$$

From (100), (105), and Proposition 10(b), it follows that:

$$S_2 = do(A_1, S_1).$$

From this and (101) and (102), it follows that:

$$\text{Executable}(F_1, do(A_1, S_1)), \text{ and} \tag{106}$$

$$\forall s.\ \text{OnPath}(P_1, s) \equiv \text{OnPathASF}(F_1, do(A_1, S_1), s). \tag{107}$$

Now, consider the pair (S_1, F_1^1), where F_1^1 is defined as follows:

$$\begin{aligned}F_1^1(s) &= A_1, \text{ if } s = S_1 \\ &= F_1(s), \text{ otherwise }.\end{aligned}$$

From (106) and Definition 4, it follows that:

$$\text{Executable}(do(A_1, S_1)).$$

From this and Lemma 34, it follows that:

$$\text{Poss}(A_1, S_1), \text{ and} \tag{108}$$

$$\text{Executable}(S_1). \tag{109}$$

From (106), (108), Definition 4, and by definition of F_1^1, it follows that:

$$\forall s.\ \text{OnPathASF}(F_1^1, S_1, s) \supset \text{Poss}(F_1^1(s), s). \tag{110}$$

From (109), (110), and Definition 4, we have that:

$$\text{Executable}(F_1^1, S_1). \tag{111}$$

Now, by (111) and Axiom 2(b), there is a path P_1^1 such that:

$$\text{Starts}(P_1^1, S_1),\ \text{and} \tag{112}$$

$$\forall s.\ \text{OnPathASF}(F_1^1, S_1, s) \equiv \text{OnPath}(P_1^1, s). \tag{113}$$

We need to show that Suffix$(P_1, P_1^1, do(A_1, S_1))$. From Lemma 33, we have:

$$S_1 \preceq do(A_1, S_1). \tag{114}$$

Also, by definition of F_1^1, it follows that:

$$\forall a, s.\ S_1 \prec do(a, s) \preceq do(A_1, S_1) \supset F_1^1(s) = a. \tag{115}$$

From (114), (115), and Definition 3, it follows that:

$$\text{OnPathASF}(F_1^1, S_1, do(A_1, S_1)).$$

From this and (113), it follows that:

$$\text{OnPath}(P_1^1, do(A_1, S_1)). \tag{116}$$

From (107) and Definition 3, we have:

$$\begin{aligned}&\forall s.\ \text{OnPath}(P_1, s) \equiv do(A_1, S_1) \preceq s\\ &\wedge \forall a, s^*.\ do(A_1, S_1) \prec do(a, s^*) \preceq s \supset F_1(s^*) = a.\end{aligned} \tag{117}$$

Similarly, from (113) and Definition 3, we have:

$$\begin{aligned}&\forall s.\ \text{OnPath}(P_1^1, s) \equiv S_1 \preceq s\\ &\wedge \forall a, s^*.\ S_1 \prec do(a, s^*) \preceq s \supset F_1^1(s^*) = a.\end{aligned} \tag{118}$$

Note that, by Lemmata 30 and 31 and transitivity of $\prec$, it follows that:

$$\forall s.\ do(A_1, S_1) \preceq s \supset s \neq S_1.$$

From this, (118), and definition of F_1^1, we have:

$$\begin{aligned}&\forall s.\ do(A_1, S_1) \preceq s \supset\\ &\quad \text{OnPath}(P_1^1, s) \equiv S_1 \preceq s\\ &\quad\quad \wedge \forall a, s^*.\ S_1 \prec do(a, s^*) \preceq s \supset F_1(s^*) = a.\end{aligned} \tag{119}$$

From (117) and (119), it follows that:

$$\forall s.\ do(A_1, S_1) \preceq s \supset \text{OnPath}(P_1, s) \equiv \text{OnPath}(P_1^1, s). \tag{120}$$

From (116), (100), (120), and Definition 6, it follows that:

$$\text{Suffix}(P_1, P_1^1, do(A_1, S_1)). \tag{121}$$

The consequent follows from (112) and (121). □

Proposition 20.

$$\begin{aligned}\Sigma \cup \mathcal{D}_{path} \models\ &\forall s_1, s_2, p_2.\ s_1 \prec s_2 \wedge \text{Starts}(p_2, s_2) \supset\\ &\quad \exists p_1.\ \text{Starts}(p_1, s_1) \wedge \text{Suffix}(p_2, p_1, s_2).\end{aligned}$$

Proof. (By induction on situation s_2) For the base case, fix S_2^b such that Init(S_2^b). Then by this and the definitions of Init$(\cdot)$ and $\prec$, we have: $\neg\exists s.\ s < S_2^b$, and thus the antecedent is false and the thesis follows trivially.

For the inductive hypothesis, fix situation S_2 and assume that:

$$\begin{aligned}&\forall s_1, p_2.\ s_1 \prec S_2 \wedge \text{Starts}(p_2, S_2) \supset\\ &\quad \exists p_1.\ \text{Starts}(p_1, s_1) \wedge \text{Suffix}(p_2, p_1, S_2).\end{aligned} \tag{122}$$

Fix A_2. We have to show that:

$$\begin{aligned}&\forall s_1, p_2.\ s_1 \prec do(A_2, S_2) \wedge \text{Starts}(p_2, do(A_2, S_2)) \supset\\ &\quad \exists p_1.\ \text{Starts}(p_1, s_1) \wedge \text{Suffix}(p_2, p_1, do(A_2, S_2)).\end{aligned}$$

Fix S_1 and P_2 and assume that:

$$S_1 \prec do(A_2, S_2),\ \text{and} \tag{123}$$

$$\text{Starts}(P_2, do(A_2, S_2)). \tag{124}$$

By (124) and Proposition 19, it follows that there is a path P_3 s.t.:

$$\text{Starts}(P_3, S_2) \wedge \text{Suffix}(P_2, P_3, do(A_2, S_2)). \tag{125}$$

Also by (123) and the definition of $\prec$, we have:

$$\exists s, a.\ do(A_2, S_2) = do(a, s) \wedge S_1 \preceq s.$$

By this and the injectivity of $do(\cdot, \cdot)$, we have:

$$S_1 \preceq S_2.$$

By this and the definition of $\preceq$, it follows that:

$$S_1 = S_2 \vee S_1 \prec S_2.$$

Case 1. Assume that $S_1 = S_2$. Then by (124) and Proposition 19, it follows that:

$$\exists p.\ \text{Starts}(p, S_1) \wedge \text{Suffix}(P_2, p, do(A_2, S_2)),$$

and we are done.

Case 2. Assume that $S_1 \prec S_2$. Then by this, (125), and (122), it follows that there is a path P_4 s.t.:

$$\text{Starts}(P_4, S_1) \wedge \text{Suffix}(P_3, P_4, S_2). \tag{126}$$

We will show that Suffix$(P_2, P_4, do(A_2, S_2))$. By (125) and Definition 6, we have:

$$\text{OnPath}(P_3, do(A_2, S_2)),\ \text{and} \tag{127}$$

$$\begin{aligned}&\forall s'.\ do(A_2, S_2) \preceq s' \supset\\ &\quad (\text{OnPath}(P_3, s') \equiv \text{OnPath}(P_2, s')).\end{aligned} \tag{128}$$

By (126) and Definition 6, we have:

$$\forall s'.\ S_2 \preceq s' \supset (\text{OnPath}(P_3, s') \equiv \text{OnPath}(P_4, s')). \tag{129}$$

Since by Lemma 33, $S_2 \preceq do(A_2, S_2)$, it follows from (127) and (129) that:

$$\text{OnPath}(P_4, do(A_2, S_2)). \tag{130}$$

Also from (128) and (129), it follows that:

$$\begin{aligned}&\forall s'.\ do(A_2, S_2) \preceq s' \supset\\ &\quad (\text{OnPath}(P_2, s') \equiv \text{OnPath}(P_4, s')).\end{aligned} \tag{131}$$

Finally, from (124), (130), (131), and Definition 6, we have: Suffix$(P_2, P_4, do(A_2, S_2))$. □

Theorem 21 (Induction on Paths)**.**

$$\Sigma \cup \mathcal{D}_{path} \models \forall Q.\ [\{\forall s, p.\ \text{Init}(s) \wedge \text{Starts}(p, s) \supset Q(p)\} \wedge \{\forall a, s.\ (\forall p.\ \text{Starts}(p, s) \supset Q(p)) \supset (\forall p'.\ \text{Starts}(p', do(a, s)) \supset Q(p'))\}] \supset \forall p.\ Q(p).$$

Proof. (By contradiction) Fix property Q_1 and assume:

$$\forall s, p.\ \text{Init}(s) \wedge \text{Starts}(p, s) \supset Q_1(p), \tag{132}$$

$$\forall a, s.\ (\forall p.\ \text{Starts}(p, s) \supset Q_1(p)) \supset (\forall p'.\ \text{Starts}(p', do(a, s)) \supset Q_1(p')). \tag{133}$$

Also assume that there is a path P_1 over which Q_1 is false:

$$\neg Q_1(P_1). \tag{134}$$

By Proposition 10(a,b), P_1 must start with some unique situation, call it S_1:

$$\text{Starts}(P_1, S_1). \tag{135}$$

We now prove by induction on s that:

$$\forall s, p.\ \text{Starts}(p, s) \supset Q_1(p).$$

For the base case when s is an initial situation, the thesis follows from (132).

For the inductive step, fix S_2 and assume that:

$$\forall p.\ \text{Starts}(p, S_2) \supset Q_1(p). \tag{136}$$

Take some arbitrary action A_1. It follows from (133) and (136) that:

$$\forall p.\ \text{Starts}(p, do(A_1, S_2)) \supset Q_1(p). \tag{137}$$

Thus by induction on s, we have:

$$\forall s, p.\ \text{Starts}(s, p) \supset Q_1(p). \tag{138}$$

By (138) and (135), it follows that $Q_1(p)$; but this is contradictory to (134). □

Theorem 22 (Induction on Situations in a Path)**.**

$$\Sigma \cup \mathcal{D}_{path} \models \forall p, Q.\ [\{\forall s.\ \text{Starts}(p, s) \supset Q(s)\} \wedge \{\forall a, s.\ (\text{OnPath}(p, s) \wedge Q(s) \wedge \text{OnPath}(p, do(a, s))) \supset Q(do(a, s))\}] \supset \forall s.\ \text{OnPath}(p, s) \supset Q(s).$$

Proof. (By contradiction) Fix path P_1 and property Q_1 and assume:

$$\forall s.\ \text{Starts}(P_1, s) \supset Q_1(s), \tag{139}$$

$$\forall a, s.\ \text{OnPath}(P_1, s) \wedge Q_1(s) \wedge \text{OnPath}(P_1, do(a, s)) \supset Q_1(do(a, s)). \tag{140}$$

Also assume that there is a situation S_{p_1} on path P_1 over which Q_1 is false:

$$\text{OnPath}(P_1, S_{P_1}) \wedge \neg Q_1(S_{P_1}). \tag{141}$$

By Proposition 10(a,b), P_1 must start with some unique situation, call it S_1:

$$\text{Starts}(P_1, S_1). \tag{142}$$

From this and Definition 1, we have:

$$\text{OnPath}(P_1, S_1). \tag{143}$$

We now prove by induction on s that:

$$\forall s.\ \text{OnPath}(P_1, s) \supset Q_1(s).$$

For the base case when s is the starting situation of P_1, i.e. S_1, the thesis follows from (139), (142), and (143).

For the inductive step, fix S_2 and assume that:

$$\text{OnPath}(P_1, S_2) \wedge Q_1(S_2). \tag{144}$$

Take some arbitrary action A_1 such that $\text{OnPath}(P_1, do(A_1, S_2))$. Then by this, (144), and (140), it follows that:

$$Q_1(do(A_1, S_2)). \tag{145}$$

Thus by induction on s, we have:

$$\forall s.\ \text{OnPath}(P_1, s) \supset Q_1(s). \tag{146}$$

But this is contradictory to (141). □

Theorem 25 (Soundness)**.**

$$\Sigma_{\mathbb{N}} \cup \Sigma \cup \mathcal{D}_{path} \models \forall p.\ (\exists \sigma.\ \text{PathSeq}(\sigma) \wedge \text{Matches}(p, \sigma)).$$

Proof. Fix path P_1. By Propositions 10(a), and 10(c), there is an executable situation S_1 such that P_1 starts with S_1:

$$\text{Starts}(P_1, S_1), \text{ and} \tag{147}$$

$$\text{Executable}(S_1). \tag{148}$$

By Axiom 2(a), there is an action selection function F_1 and situation S_2 such that: $\text{Executable}(F_1, S_2) \wedge \forall s'.\ \text{OnPathASF}(F_1, S_2, s') \equiv \text{OnPath}(P_1, s')$. From this and Definition 3, it follows that S_2 is the earliest situation of path P_1. Moreover, from Definition 1 and (147), it follows that S_1 is the earliest situation that is also on path P_1. Thus it follows that $S_1 = S_2$ and hence we have:

$$\text{Executable}(F_1, S_1), \text{ and} \tag{149}$$

$$\forall s'.\ \text{OnPathASF}(F_1, S_1, s') \equiv \text{OnPath}(P_1, s'). \tag{150}$$

Let σ_1 be defined as follows:

$$\sigma_1(0) = S_1, \tag{151}$$

$$\sigma_1(n+1) = do(F_1(\sigma_1(n)), \sigma_1(n)), \quad \text{for } n \geq 0. \tag{152}$$

We have to prove that $\text{PathSeq}(\sigma_1) \wedge \text{Matches}(P_1, \sigma_1)$.

First, let us show that $\text{PathSeq}(\sigma_1)$. By Definition 23, to show this we have to prove that:

(a). $\text{Executable}(\sigma_1(0))$, and

(b). $\forall n.\ \exists a.\ \text{Poss}(a, \sigma_1(n)) \wedge \sigma_1(n+1) = do(a, \sigma_1(n))$.

(a) follows from (148) and (151). By (152), for each n there is indeed an action $a = F_1(\sigma_1(n))$ s.t. $\sigma_1(n+1) = do(a, \sigma_1(n))$. Thus to show (b), we have to prove that $\forall n.\ \text{Poss}(F_1(\sigma_1(n)), \sigma_1(n))$. Now, from (149) and Definition 4, it follows that:

$$\forall s'.\ \text{OnPathASF}(F_1, S_1, s') \supset \text{Poss}(F_1(s'), s').$$

Thus, to prove that $\forall n.\ \text{Poss}(F_1(\sigma_1(n)), \sigma_1(n))$, we just need to show that:

$$\forall n.\ \text{OnPathASF}(F_1, S_1, \sigma_1(n)).$$

I will show this by induction on n. For the base case, i.e. when $n = 0$, it follows from (151) that $\sigma_1(n) = S_1$. Thus we have to show that OnPathASF(F_1, S_1, S_1). This follows trivially from the definition of $\preceq$, Definition 3, and transitivity and irreflexivity of $\prec$ (which imply that there are no situations $do(a, s^*)$ such that $S_1 \prec do(a, s^*) \preceq S_1$). For the inductive case, fix N_1 and assume that:

$$\text{OnPathASF}(F_1, S_1, \sigma_1(N_1)). \tag{153}$$

We have to show that OnPathASF$(F_1, S_1, \sigma_1(N_1 + 1))$. From (153) and Definition 3, we have:

$$S_1 \preceq \sigma_1(N_1), \text{ and} \tag{154}$$
$$\forall a, s^*.\ S_1 \prec do(a, s^*) \preceq \sigma_1(N_1) \supset F_1(s^*) = a. \tag{155}$$

From (155) and Definition 3, it follows that OnPathASF$(F_1, S_1, \sigma_1(N_1 + 1))$ holds if the following hold:

(b1). $S_1 \preceq \sigma_1(N_1 + 1)$, and
(b2). $\forall a, s^*.\ \sigma_1(N_1) \prec do(a, s^*) \preceq \sigma_1(N_1 + 1)$
$\supset F_1(s^*) = a$.

Now, from (152), we have:

$$\sigma_1(N_1 + 1) = do(F_1(\sigma_1(N_1)), \sigma_1(N_1)). \tag{156}$$

From this and Lemma 33, we have:

$$\sigma_1(N_1) \preceq \sigma_1(N_1 + 1).$$

(b1) follows from this, (154), and the transitivity of $\preceq$. Moreover, (b2) follows from (156) and transitivity and irreflexivity of $\prec$ (which imply that there are no situations between $\sigma_1(N_1)$ and $\sigma_1(N_1 + 1)$). Thus, we have PathSeq(σ_1).

Next, let us show that Matches(P_1, σ_1). We already proved that σ_1 is a path sequence. Thus by Definition 24, we need to show that:

(c). $\sigma_1(0) = s \equiv \text{Starts}(P_1, s)$ and
(d). $\forall n, s.\ [\sigma_1(n) = s \wedge \text{OnPath}(P_1, s) \supset$
$\forall a.\ (\sigma_1(n + 1) = do(a, s) \equiv \text{OnPath}(P_1, do(a, s)))]$.

(c) follows from (147), (151) and the uniqueness of starting situations of paths, i.e. Proposition 10(b). For (d), fix N_1 and $\widehat{S_1}$ and assume that:

$$\sigma_1(N_1) = \widehat{S_1}, \text{ and} \tag{157}$$
$$\text{OnPath}(P_1, \widehat{S_1}). \tag{158}$$

For the $(\supset)$ direction, fix A_1 and assume that:

$$\sigma_1(N_1 + 1) = do(A_1, \widehat{S_1}).$$

Then by this and (152), we have:

$$do(A_1, \widehat{S_1}) = do(F_1(\sigma_1(N_1)), \sigma_1(N_1)).$$

From this and (157), we have:

$$do(A_1, \widehat{S_1}) = do(F_1(\widehat{S_1}), \widehat{S_1}). \tag{159}$$

From (150) and (158), we have:

$$\text{OnPathASF}(F_1, S_1, \widehat{S_1}).$$

From this and Definition 3, it follows that:

$$S_1 \preceq \widehat{S_1}, \text{ and} \tag{160}$$
$$\forall a, s^*.\ S_1 \prec do(a, s^*) \preceq \widehat{S_1} \supset F_1(s^*) = a. \tag{161}$$

Now, consider the situation $do(F_1(\widehat{S_1}), \widehat{S_1})$. From Lemma 33, we have:

$$\widehat{S_1} \preceq do(F_1(\widehat{S_1}), \widehat{S_1}). \tag{162}$$

From this, (160), and the transitivity of $\preceq$, it follows that:

$$S_1 \preceq do(F_1(\widehat{S_1}), \widehat{S_1}). \tag{163}$$

Moreover, from (161), (162), and transitivity and irreflexivity of $\prec$, it follows that:

$$\forall a, s^*.\ S_1 \prec do(a, s^*) \preceq do(F_1(\widehat{S_1}), \widehat{S_1}) \supset F_1(s^*) = a. \tag{164}$$

From (163), (164), and Definition 3, it follows that:

$$\text{OnPathASF}(F_1, S_1, do(F_1(\widehat{S_1}), \widehat{S_1})).$$

From this and (150), it follows that OnPath$(P_1, do(F_1(\widehat{S_1}), \widehat{S_1}))$, i.e. by (159) that OnPath$(P_1, do(A_1, \widehat{S_1}))$.

For the $(\subset)$ direction, fix A_1 and assume that:

$$\text{OnPath}(P_1, do(A_1, \widehat{S_1})).$$

Then from this and (150), it follows that:

$$\text{OnPathASF}(F_1, S_1, do(A_1, \widehat{S_1})).$$

From this and Definition 3, it follows that:

$$A_1 = F_1(\widehat{S_1}). \tag{165}$$

Now, since by (157), $\sigma_1(N_1) = \widehat{S_1}$, it follows by (152) that:

$$\sigma_1(N_1 + 1) = do(F_1(\widehat{S_1}), \widehat{S_1}).$$

From this and (165), we have:

$$\sigma(N_1 + 1) = do(A_1, \widehat{S_1}).$$

Thus P_1 matches σ_1. $\square$

We will use the following to prove the completeness theorem. Given Definition 23, it can be shown that if σ is a path sequence and $i < j$, then the situation given by $\sigma(i)$ precedes the one given by $\sigma(j)$:

Lemma 38.

$$\Sigma_{\mathbb{N}} \cup \Sigma \cup \mathcal{D}_{path} \models \forall \sigma, i, j.\ \text{PathSeq}(\sigma) \wedge i < j \supset \sigma(i) \prec \sigma(j).$$

Proof. (By induction on n, where $n = j - i$) Fix σ_1 and assume:

$$\text{PathSeq}(\sigma_1). \tag{166}$$

For the base case, fix I_1 and J_1 and assume that $J_1 - I_1 = 1$. Then it follows from (166) and Definition 23 that there is an action A_1 s.t.:

$$\sigma_1(J_1) = \sigma_1(I_1 + 1) = do(A_1, \sigma_1(I_1)).$$

From this and Lemma 30, it follows that $\sigma_1(I_1) \prec do(A_1, \sigma_1(I_1))$, i.e. $\sigma_1(I_1) \prec \sigma_1(I_1+1)$, and thus $\sigma_1(I_1) \prec \sigma_1(J_1)$.

For the inductive case, fix I_N, J_N, and N_1 and assume that:

$$J_N - I_N = N_1, \text{ and} \tag{167}$$
$$\text{PathSeq}(\sigma_1) \wedge I_N < J_N \supset \sigma_1(I_N) \prec \sigma_1(J_N). \tag{168}$$

We have to show that:

$$\text{PathSeq}(\sigma_1) \wedge I_N < J_{N+1} \supset \sigma_1(I_N) \prec \sigma_1(J_{N+1}),$$

where $J_{N+1} - I_N = N_1 + 1$, i.e. by (167), $J_{N+1} = J_N + 1$. Now, from (166) and (167), it follows that PathSeq$(\sigma_1) \wedge I_N < J_N$. From this and the inductive hypothesis (i.e. (168)), we have:

$$\sigma_1(I_N) \prec \sigma_1(J_N). \tag{169}$$

Moreover, from (166) and Definition 23 it follows that there is an action A_N s.t.:

$$\sigma_1(J_N + 1) = do(A_N, \sigma_1(J_N)).$$

From this and Lemma 30, it follows that $\sigma_1(J_N) \prec do(A_N, \sigma_1(J_N))$, i.e. $\sigma_1(J_N) \prec \sigma_1(J_N + 1)$, and thus:

$$\sigma_1(J_N) \prec \sigma_1(J_{N+1}). \tag{170}$$

Finally, from (169), (170), and the transitivity of $\prec$, we have: $\sigma_1(I_N) \prec \sigma_1(J_{N+1})$. □

To show that for every path sequence there is a corresponding path, it is useful to first introduce a corresponding ASF. Given path sequence σ, let F_σ be the ASF defined as follows:

$$F_\sigma(s) = a_n, \quad \text{if } \exists n.\, \sigma(n) = s \wedge \sigma(n+1) = do(a_n, s),$$
$$F_\sigma(s) = b \qquad \text{otherwise},$$

where b is some fixed but arbitrary action.

We can show that that given a path sequence σ, any situation s that is on the path defined by the corresponding ASF F_σ and the initial situation of the path sequence $\sigma(0)$ is in fact on the path sequence σ at some position n:

Lemma 39.

$$\Sigma_{\mathbb{N}} \cup \Sigma \cup \mathcal{D}_{path} \models$$
$$\forall s, \sigma.\ \text{PathSeq}(\sigma) \wedge \sigma(0) \prec s \wedge \text{OnPathASF}(F_\sigma, \sigma(0), s)$$
$$\supset \exists n.\ \sigma(n) = s.$$

Proof. (By induction on s) Fix σ_1. Construct a function from situations to actions F_{σ_1} such that F_{σ_1} is the corresponding ASF to σ_1. Also, assume that:

$$\text{PathSeq}(\sigma_1). \tag{171}$$

In the base case where s is an initial situation, $\neg\exists s'.\ s' \prec s$ by the definition of Init$(\cdot)$ and $\prec$, so the antecedent is false and the thesis trivially holds.

For the inductive step, fix S_N and assume that:

$$\sigma_1(0) \prec S_N \wedge \text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), S_N)$$
$$\supset \exists n.\ \sigma_1(n) = S_N. \tag{172}$$

Also, fix action A_N and assume that:

$$\sigma_1(0) \prec do(A_N, S_N), \text{ and} \tag{173}$$
$$\text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), do(A_N, S_N)). \tag{174}$$

From (174) and Definition 3, it follows that:

$$\forall a, s.\ \sigma_1(0) \prec do(a, s) \preceq do(A_N, S_N) \supset F_{\sigma_1}(s) = a. \tag{175}$$

From (173) and the definition of $\prec$, it follows that:

$$\sigma_1(0) \preceq S_N. \tag{176}$$

From Lemma 33, we have $S_N \preceq do(A_N, S_N)$. From this and (175), we have:

$$\forall a, s.\ \sigma_1(0) \prec do(a, s) \preceq S_N \supset F_{\sigma_1}(s) = a. \tag{177}$$

From (176), (177), and Definition 3, we have:

$$\text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), S_N). \tag{178}$$

Now, (176) and the definition of $\preceq$ give us two cases. In the case where $\sigma_1(0) = S_N$, it trivially follows that $\exists n.\ \sigma_1(n) = S_N$. In the case where $\sigma_1(0) \prec S_N$, from this, (178), and the induction hypothesis, i.e. (172), it follows that $\exists n.\ \sigma_1(n) = S_N$. Thus, in both these cases, there is a N_1 such that:

$$\sigma_1(N_1) = S_N. \tag{179}$$

From (171), (179), and Definition 23, it follows that there is an action, let us call it A_N^*, s.t.:

$$\sigma_1(N_1 + 1) = do(A_N^*, \sigma_1(N_1)). \tag{180}$$

We just need to show that $A_N^* = A_N$. From the definition of F_{σ_1}, it follows that:

$$\forall a.\ \sigma_1(N_1 + 1) = do(a, \sigma_1(N_1)) \supset F_{\sigma_1}(\sigma_1(N_1)) = a. \tag{181}$$

From (179), (180), and (181), it follows that:

$$F_{\sigma_1}(S_N) = A_N^*. \tag{182}$$

From (175), we have $F_{\sigma_1}(S_N) = A_N$. Finally from this and (182), we have $A_N = A_N^*$, and thus from this, (180), and (179), it follows that $\sigma_1(N_1 + 1) = do(A_N, S_N)$, i.e. $\exists n.\ \sigma_1(n) = do(A_N, S_N)$. □

Theorem 26 (Completeness)**.**

$$\Sigma_{\mathbb{N}} \cup \Sigma \cup \mathcal{D}_{path} \models \forall \sigma.\ \text{PathSeq}(\sigma) \supset \exists p.\ \text{Matches}(p, \sigma).$$

Proof. Fix function σ_1 and assume that:

$$\text{PathSeq}(\sigma_1). \tag{183}$$

From this and Definition 23, it follows that:

$$\text{Executable}(\sigma_1(0)), \text{ and} \tag{184}$$
$$\forall n.\ \exists a.\ \text{Poss}(a, \sigma_1(n)) \wedge \sigma_1(n+1) = do(a, \sigma_1(n)). \tag{185}$$

Construct a tuple $(\sigma_1(0), F_{\sigma_1})$ such that F_{σ_1}, which is a function from situations to actions, is the corresponding ASF to σ_1. We will now show that Executable$(F_{\sigma_1}, \sigma_1(0))$. Assume otherwise. Then from Definition 4 and (184), it follows that there is a situation S_N such that:

$$\text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), S_N), \text{ and} \tag{186}$$
$$\neg\text{Poss}(F_{\sigma_1}(S_N), S_N). \tag{187}$$

From (186) and Definition 3, it follows that $\sigma_1(0) \preceq S_N$. This and the definition of $\preceq$ give us two cases. In the case where $\sigma_1(0) = S_N$, it trivially follows that $\exists n.\ \sigma_1(n) = S_N$. In the case where $\sigma_1(0) \prec S_N$, from (183), the assumption for this case that $\sigma_1(0)$ strictly precedes S_N, (186), and Lemma 39, it follows that $\exists n.\ \sigma_1(n) = S_N$. Thus, for both these cases, we have that there is a n, say N_1, s.t.:

$$\sigma_1(N_1) = S_N. \tag{188}$$

Then from this and (185), it follows that there is an action A_N s.t.:

$$\sigma_1(N_1 + 1) = do(A_N, S_N), \text{ and} \tag{189}$$
$$\text{Poss}(A_N, S_N). \tag{190}$$

From (188), (189), and the definition of F_{σ_1}, it follows that:

$$F_{\sigma_1}(S_N) = A_N.$$

Finally, from this and (190), we have Poss$(F_{\sigma_1}(S_N), S_N)$; but this is contradictory to (187). Thus, we have:

$$\text{Executable}(F_{\sigma_1}, \sigma_1(0)). \tag{191}$$

From this and Axiom 2(b), it follows that there is a path P_1 such that:

$$\text{Starts}(P_1, \sigma_1(0)), \text{ and} \tag{192}$$
$$\forall s.\ \text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), s) \equiv \text{OnPath}(P_1, s). \tag{193}$$

Now, we need to show that Matches(P_1, σ_1). By Definition 24, this amounts to showing that:

(a). PathSeq(σ_1), and
(b). $\sigma_1(0) = s \equiv$ Starts(P_1, s) and
(c). $\forall n, s.\ [\sigma_1(n) = s \wedge \text{OnPath}(P_1, s) \supset$
$\quad \forall a.\ (\sigma_1(n+1) = do(a, s) \equiv \text{OnPath}(P_1, do(a, s)))]$.

(a) follows from the antecedent, i.e. (183). (b) follows from (192) and the uniqueness of starting situations of paths, i.e. Proposition 10(b). For (c), fix $\widehat{N_1}$ and $\widehat{S_1}$ and assume that:

$$\sigma_1(\widehat{N_1}) = \widehat{S_1}, \text{ and} \tag{194}$$
$$\text{OnPath}(P_1, \widehat{S_1}). \tag{195}$$

For the ($\supset$) direction, fix $\widehat{A_1}$ and assume that:

$$\sigma_1(\widehat{N_1} + 1) = do(\widehat{A_1}, \widehat{S_1}). \tag{196}$$

From (195) and (193), it follows that:

$$\text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), \widehat{S_1}).$$

From this and Definition 3, we have:

$$\forall a, s.\ \sigma_1(0) \prec do(a, s) \preceq \widehat{S_1} \supset F_{\sigma_1}(s) = a. \tag{197}$$

From (194), (196), and the definition of F_{σ_1}, we have:

$$F_{\sigma_1}(\widehat{S_1}) = \widehat{A_1}. \tag{198}$$

Now, suppose $\neg$OnPath$(P_1, do(\widehat{A_1}, \widehat{S_1}))$. Then by (193), we have:

$$\neg\text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), do(\widehat{A_1}, \widehat{S_1})). \tag{199}$$

From Lemma 38, (183), and the fact that $0 < \widehat{N_1} + 1$, we have $\sigma_1(0) \prec \sigma_1(\widehat{N_1} + 1)$. From this and (196), we have:

$$\sigma_1(0) \prec do(\widehat{A_1}, \widehat{S_1}).$$

From this, (199), and Definition 3, we have:

$$\exists a, s.\ \sigma_1(0) \prec do(a, s) \preceq do(\widehat{A_1}, \widehat{S_1}) \wedge \neg F_{\sigma_1}(s) = a.$$

From this and (197), it follows that $\neg(F_{\sigma_1}(\widehat{S_1}) = \widehat{A_1})$; but this is contradictory to (198).

For the ($\subset$) direction, fix $\widehat{A_2}$ and assume that:

$$\text{OnPath}(P_1, do(\widehat{A_2}, \widehat{S_1})). \tag{200}$$

From (183) and Definition 23, it follows that there is an action, say $\widehat{A_3}$, s.t.:

$$\sigma_1(\widehat{N_1} + 1) = do(\widehat{A_3}, \sigma_1(\widehat{N_1})). \tag{201}$$

I will show that $\widehat{A_2} = \widehat{A_3}$. From (201) and (194), it follows that:

$$\sigma_1(\widehat{N_1} + 1) = do(\widehat{A_3}, \widehat{S_1}). \tag{202}$$

From (200) and (193), we have:

$$\text{OnPathASF}(F_{\sigma_1}, \sigma_1(0), do(\widehat{A_2}, \widehat{S_1})).$$

From this and Definition 3, we have $F_{\sigma_1}(\widehat{S_1}) = \widehat{A_2}$. Finally from this, (194), (202), and the definition of F_{σ_1}, we have $\widehat{A_2} = \widehat{A_3}$. Thus from this and (202), it follows that $\sigma_1(\widehat{N_1} + 1) = do(\widehat{A_2}, \widehat{S_1})$. □

References

Boutilier, C.; Reiter, R.; Soutchanski, M.; and Thrun, S. 2000. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 355–362.

Claßen, J., and Lakemeyer, G. 2008. A logic for non-terminating Golog programs. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference (KR-08)*, 589–599.

De Giacomo, G.; Lespérance, Y.; Levesque, H. J.; and Sardina, S. 2004. On the semantics of deliberation in IndiGolog – from theory to implementation. *Annals of Mathematics and Artificial Intelligence* 41(2–4):259–299.

De Giacomo, G.; Lespérance, Y.; Patrizi, F.; and Sardina, S. 2016. Verifying ConGolog programs on bounded situation calculus theories. In *AAAI Conference on Artificial Intelligence (to appear)*.

De Giacomo, G.; Lespérance, Y.; and Levesque, H. J. 2000. ConGolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence* 121:109–169.

De Giacomo, G.; Ternovskaia, E.; and Reiter, R. 1997. Non-terminating processes in the situation calculus. In *Working Notes of Robots, Softbots, Immobots: Theories of Action, Planning and Control. AAAI-97 Workshop*.

Emerson, E. A., and Halpern, J. Y. 1986. "sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. ACM* 33(1):151–178.

Emerson, E. A. 1996. Model checking and the mu-calculus. In *Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop*, 185–214.

Fritz, C., and McIlraith, S. A. 2006. Decision-theoretic golog with qualitative preferences. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Tenth International Conference (KR-06)*, 153–163.

Gabaldon, A. 2004. Precondition control and the progression algorithm. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR-04)*, 634–643.

Khan, S. M., and Lespérance, Y. 2005. ECASL: A Model of Rational Agency for Communicating Agents. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-05)*, 762–769.

Khan, S. M., and Lespérance, Y. 2009. Prioritized goals and subgoals in a logical account of goal change - A preliminary report. In *Declarative Agent Languages and Technologies VII, 7th International Workshop, DALT 2009, Budapest, Hungary, May 11, 2009. Revised Selected and Invited Papers*, 119–136.

Khan, S. M., and Lespérance, Y. 2010. A logical framework for prioritized goal change. In *9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3*, 283–290.

Lakemeyer, G., and Levesque, H. J. 1998. AOL: A logic of acting, sensing, knowing, and only-knowing. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR-98)*, 316–327.

Lespérance, Y.; Levesque, H. J.; Lin, F.; and Scherl, R. 2000. Ability and knowing how in the situation calculus. *Studia Logica* 66(1):165–186.

Lespérance, Y. 2001. On the epistemic feasibility of plans in multiagent systems specifications. In Meyer, J.-J. C., and Tambe, M., eds., *Intelligent Agents VIII, Agent Theories, Architectures, and Languages, 8th International Workshop (ATAL-2001)*.

Levesque, H. J.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. B. 1997. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming* 31:59–84.

Levesque, H. J.; Pirri, F.; and Reiter, R. 1998. Foundations for a calculus of situations. *Electronic Transactions of AI (ETAI)* 2(3–4):159–178.

Levesque, H. J. 1996. What is planning in the presence of sensing? In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 1139–1146.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678.

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

Moore, R. C. 1990. A formal theory of knowledge and action. In Allen, J. F.; Hendler, J.; and Tate, A., eds., *Readings in Planning*. San Mateo, CA: Morgan Kaufmann Publishers. 480–519.

Pinto, J. A. 1994. *Temporal Reasoning in the Situation Calculus*. Ph.D. Dissertation, University of Toronto, Toronto, ON, Canada.

Pnueli, A. 1977. The temporal logic of programs. In *Proceedings of the Eighteenth Annual Symposium on Foundations of Computer Science (FOCS-77)*, 46–57.

Reiter, R. 1991. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Lifschitz, V., ed., *Artificial Intelligence and Mathematical Theory of Computation: Papers in the Honor of John McCarthy*. San Diego, CA: Academic Press. 359–380.

Reiter, R. 1996. Natural actions, concurrency, and continuous time in the situation calculus. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR-96)*, 2–13.

Reiter, R. 2001. *Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Sardina, S., and Shapiro, S. 2003. Rational action in agent programs with prioritized goals. In *Second International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '03)*, 417–424.

Scherl, R. B., and Levesque, H. J. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144:1–39.

Schnoebelen, P. 2002. The complexity of temporal logic model checking. In *Advances in Modal Logic 4, papers from the fourth conference on "Advances in Modal logic"*, 393–436.

Shapiro, S.; Pagnucco, M.; Lespérance, Y.; and Levesque, H. J. 2011. Iterated belief change in the situation calculus. *Artificial Intelligence* 175(1):165–192.

Shapiro, S.; Lespérance, Y.; and Levesque, H. 2002. The cognitive agents specification language and verification environment for multiagent systems. In Castelfranchi, C., and Johnson, W. L., eds., *Proc. of the 1st Int. Joint Conference on Autonomous Agents and Multiagent Systems*, 19–26. Bologna, Italy: ACM Press.

Shapiro, S.; Lespérance, Y.; and Levesque, H. J. 2005. Goal change. In *International Joint Conference on Artificial Intelligence (IJCAI 2005)*.

Shapiro, S. 2005. *Specifying and Verifying Multiagent Systems using the Cognitive Agents Specification Language (CASL)*. Ph.D. Dissertation, University of Toronto, Toronto, ON, Canada.