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

by

Vahid Vaezian

M.Sc., Sharif University of Technology, Tehran, Iran, 2011
B.Sc., Iran University of Science and Technology, Iran, 2008

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2019

Approval

Name:	**Vahid Vaezian**
Degree:	**Master of Science (Computer Science)**
Title:	**On the Role of Possibility in Action Execution and Knowledge in the Situation Calculus**
Examining Committee:	**Chair:** Oliver Schulte Professor
	James P. Delgrande Senior Supervisor Professor
	David Mitchell Supervisor Associate Professor
	Aaron Hunter External Examiner Research Faculty School of Computing and Academic Studies British Columbia Institute of Technology
Date Defended:	**January 11, 2019**

Abstract

Formalization of knowledge is an important aspect of reasoning about change. We review how knowledge is formalized in the Situation Calculus (a logical formalism for reasoning about action and change) and discuss the problems that occur when unexecutable actions (those actions whose preconditions are not met at the time of execution) are involved. We then provide a generalized framework that addresses these problems by tracing back source of the problem to the answer provided to the Frame Problem in the Situation Calculus. We develop a more generalized form for Successor Sate Axioms based on the new account of the solution to the Frame Problem and show how this solves the problems related to involvement of unexecutable actions.

Keywords: Knowledge Representation and Reasoning, Theories of Action, Situation Calculus, Event Calculus, Frame Problem

Acknowledgements

We wish to thank Jim Delgrande for his academic and financial support, and Dave Mitchell and Aaron Hunter for their helpful discussions.

Table of Contents

List of Figures

Chapter 1

Introduction

A *theory of action* is a set of axioms that captures the dynamics of the world, and make it possible to reason about and represent actions. Examples of theories of actions are -but not limited to- Situation Calculus[15], Event Calculus[12], Action Languages[8] and Fluent Calculus[9].

The Situation Calculus is among the first and more popular of these formalisms, and we use it as the main formalism throughout this thesis. We use the Event Calculus in some sections for comparison.

Formalizing knowledge is an important aspect of reasoning about change. The standard approach to this problem in the Situation Calculus is based on works of Moore ([19], [20]) which provides a possible-world account of knowledge by using situations to represent possible worlds. There have been several variants of formalization of knowledge in the Situation Calculus based on this approach. We will argue that these approaches does not work when unexecutable actions (i.e. those actions which their preconditions are not met at the time of execution) are involved. In addition to formalization of knowledge, presence of unexecutable actions causes problems in some other areas such as the Projection Problem (given an initial state and a sequence of actions, determining whether a formula holds in the resulting state) as well. We trace back the source of problem to the answer provided to the frame problem in the Situation Calculus. We provide a more general solution to the frame problem in the Situation Calculus and develop a more general form for successor state axioms.

We will show that this new form solves the aforementioned problems. Another result of the new framework is a different reading of the term $do(a, s)$. While in the traditional framework it is defined as "the successor situation to s, resulting from *performing* action a", in our framework the reading would be "the successor situation to s, resulting from *attempting* action a." In other words, it is not assumed that actions always succeed.

In Chapter 2 we start with review of the required background material by giving a summary of the Situation and Event calculus along with their epistemic versions. Chapter 3 includes discussion of problems and limitations that exist in the current framework in

presence of unexecutable situations. In Chapter 4 we introduce our framework and discuss how it solves the aforementioned problems in the traditional framework of the Situation Calculus. Finally in Chapter 5 we conclude and present directions for future work.

Chapter 2

Background

In this section we review two of the most well-known formalism for representing and reasoning about actions and change, namely the Situation Calculus and the Event Calculus. We then briefly review syntax and semantics of modal logics and discuss the epistemic versions of the Situation and the Event Calculus.

2.1 Situation Calculus

The Situation Calculus is a first-order formalism (with some second-order elements) designed for representing and reasoning about actions. It was first proposed by J. McCarthy [15], [16] but it was not widely investigated until Reiter reintroduced it in the early Nineties. Since then it has been an active field of research. The version of the Situation Calculus we discuss in this thesis is mainly based on [24]. Definitions and axioms in this section are from [24] unless stated otherwise.

The language of the Situation Calculus $\mathcal{L}_{sitcalc}$ is a many-sorted second-order language. There are three sorts: *action* for actions, *situation* for situations, and *object* for all other domain objects.

Informally, a *basic action theory* consists of a set of distinguished symbols, a set of domain-independent axioms describing how the theory formalizes the dynamics of the world and some domain-specific axioms which describe the dynamics of the specific domain under consideration.

Distinguished Symbols

There are three distinguished symbols: A constant symbol S_0 called the *initial situation* which represents the initial state of the world where no action has been executed, a function symbol *do* which maps an action-situation pair to a new situation where the action has been executed and a predicate symbol $Poss$ where $Poss(a, s)$ denotes "it is possible to perform action a is situation s".

A *situation* is a finite sequence of actions. A binary predicate symbol $\sqsubset$ defines an ordering relation on situations.[1] A *fluent* is a predicate or function whose value may change in different situations as a result of performing actions. Syntactically a fluent is a predicate or a function (depending on whether it is a relational or a functional fluent) that takes a situation as the final argument.

Domain-Specific Axioms

Domain-specific axioms consist of an *action precondition* axiom for each action symbol, a *successor state axiom* for each fluent symbol, plus *unique name axioms.*

An *action precondition axiom* of $\mathcal{L}_{sitcalc}$ is a formula of the form:[2]

$$Poss(A(\vec{x}), s) \equiv \psi_A(\vec{x}, s)$$

where A is an action, $\equiv$ is biconditional, and $\psi_A(\vec{x}, s)$ is a formula that is uniform[3] in s and whose free variables are among $\vec{x}, s$.

Successor state axioms for relational fluents F and functional fluents f are sentences of $\mathcal{L}_{sitcalc}$ of the form:

$$\begin{aligned} F(\vec{x}, do(a, s)) &\equiv \phi_F(\vec{x}, a, s) \\ f(\vec{x}, do(a, s)) = y &\equiv \phi_f(\vec{x}, y, a, s) \end{aligned} \tag{2.1}$$

where $\phi_F(\vec{x}, a, s)$ and $\phi_f(\vec{x}, y, a, s)$ are formulas uniform in s. The uniformity requirement in s is to make sure that the definition of $Poss$, the truth value of F in the successor situation and value of f in the successor situation are determined only by the current situation.

In practice, the following forms of relational and functional successor state axioms are used in the literature:

$$\begin{aligned} F(\vec{x}, do(a, s)) &\equiv \gamma_F^+(\vec{x}, a, s) \vee [(F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s))] \\ f(\vec{x}, do(a, s)) = y &\equiv \gamma_f(\vec{x}, y, a, s) \vee [(f(\vec{x}, s) = y \wedge \neg(\exists y')\gamma_f(\vec{x}, y', a, s))] \end{aligned}$$

where $\gamma_F^+(\vec{x}, a, s)$ and $\gamma_F^-(\vec{x}, a, s)$ represent the conditions under which action a affects the value of fluent F positively or negatively, and $\gamma_f(\vec{x}, y, a, s)$ states "the conditions under which action a can cause the functional fluent f to take on value y in the successor situation"[24].

[1] $s \sqsubseteq s'$ is an abbreviation for $s \sqsubset s' \vee s = s'$.

[2] The free variables are implicitly universally quantified.

[3] A formula of $\mathcal{L}_{sitcalc}$ is *uniform* in σ iff "it does not mention the predicates $Poss$ or $\sqsubset$, it does not quantify over variables of sort situation, it does not mention equality on situations, and whenever it mentions a term of sort *situation* in the situation argument position of a fluent, then that term is σ."[24]

Note that $\gamma_F^+(\vec{x}, a, s)$ and $\gamma_F^-(\vec{x}, a, s)$ are first-order formulas with free variables among $\vec{x}, a, s$, and $\gamma_f(\vec{x}, y, a, s)$ is a first-order formula with free variables among $\vec{x}, y, a, s$. Our focus will be on relational fluents.

Let A and B be two different actions. *Unique names axioms* for actions are as follows.

$$A(\vec{x}) \neq B(\vec{y})$$
$$A(x_1, \ldots, x_n) = A(y_1, ..., y_n) \supset x_1 = y_1 \wedge \ldots \wedge x_n = y_n.$$

Domain-Independent Axioms

The following are the domain-independent (aka foundational axioms) of the Situation Calculus:

$$do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2 \tag{2.2}$$
$$(\forall P).P(S_0) \wedge (\forall a, s)[P(s) \supset P(do(a, s))] \supset (\forall s)P(s) \tag{2.3}$$
$$\neg s \sqsubset S_0 \tag{2.4}$$
$$s \sqsubset do(a, s') \equiv s \sqsubseteq s' \tag{2.5}$$

Axiom (2.3), called the induction axiom, is the second-order element of Situation Calculus. It "has the effect of limiting the sort situation to the smallest set containing S_0, and is closed under the application of the function do"[24]. Axiom (2.2) can be seen as a unique names axiom for situations.

A *basic action theory* is a collection of axioms $\mathcal{D} = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ where $\Sigma, \mathcal{D}_{ap}, \mathcal{D}_{ss}, \mathcal{D}_{una}$ are the set of foundational, action precondition, successor state and unique name axioms respectively, as defined above, and $\mathcal{D}_{S_0}$ is the set of formulas uniform in S_0 describing the initial situation.

An important concept in the Situation Calculus is the concept of an *executable situation* (i.e. those sequences of actions in which all the actions are possible as determined by $Poss$). It is formally defined as

$$executable(s) \stackrel{def}{=} (\forall a, s').do(a, s') \sqsubseteq s \supset Poss(a, s')$$

If action a is not executable in situation s, the resulting situation $do(a, s)$ is what Reiter calls a *ghost situation* which represents an imaginary state of the world where a was actually executed (although it was not possible to be executed). Only executable situations are regarded as "useful" and ghost situations are excluded from consideration.

Automated reasoning in the Situation Calculus is done using *regression*. Intuitively, to prove that the axiomatization entails a formula ϕ, instead of a direct proof we first replace

fluents and *Poss* atoms in ϕ with equivalent formulas given by successor state and action precondition axioms. We repeat this until we have a formula which is uniform in S_0. We then determine whether this transformed formula is entailed or not. Since we are substituting equivalent formulas, the original formula is entailed by the axiomatization if and only if the transformed formula is.

2.2 Event Calculus

The Event calculus was introduced by Kowalski and Sergot [12] as a logic-based approach for reasoning about events with special focus on incorporating the element of time in the formalism. A variety of different versions and extensions of the original Event Calculus have been developed. Here we review a simple version. In what follows, definitions are taken from [18] unless stated otherwise.

The language of the Event Calculus is a many-sorted first-order language. There are four sorts: *action* for actions, *fluent* for fluents, *time* for time points and *object* for all other domain objects.

Distinguished Symbols

The language contains five distinguished predicate symbols as described in the following table along with their intended meanings:

$Happens(a, t)$	Action a occurs at time t
$Initiates(a, f, t)$	An occurrence of a at time t has an initiative influence on f
$Terminates(a, f, t)$	An occurrence of a at time t has a terminating influence on f
$HoldsAt(f, t)$	Fluent f holds at time t
$t_1 < t_2$	standard linear order relation between time points

It is convenient to define the following predicates:

$$Clipped(t_1, f, t_2) \doteq (\exists a, t).Happens(a, t) \wedge \ t_1 \leq t < t_2 \ \wedge Terminates(a, f, t)$$
$$Declipped(t_1, f, t_2) \doteq (\exists a, t).Happens(a, t) \wedge \ t_1 \leq t < t_2 \ \wedge Initiates(a, f, t)$$

$clipped(t_1, f, t_2)$ means fluent f is terminated between t_1 and t_2 (because of the occurrence of some action which has a terminating effect on f, i.e. after that action the fluent f doesn't hold). $Declipped(t_1, f, t_2)$ is defined similarly.

Domain-Independent Axioms

There are four domain-independent axioms as follows:[4]

$$HoldsAt(f, t_2) \leftarrow Happens(a, t_1) \wedge Initiates(a, f, t_1) \wedge\ t_1 < t_2\ \wedge \neg Clipped(t_1, f, t_2) \tag{EC1}$$

$$\neg HoldsAt(f, t_2) \leftarrow Happens(a, t_1) \wedge Terminates(a, f, t_1) \wedge\ t_1 < t_2\ \wedge \neg Declipped(t_1, f, t_2) \tag{EC2}$$

$$HoldsAt(f, t_2) \leftarrow HoldsAt(f, t_1) \wedge\ t_1 < t_2\ \wedge \neg Clipped(t_1, f, t_2) \tag{EC3}$$

$$\neg HoldsAt(f, t_2) \leftarrow \neg HoldsAt(f, t_1) \wedge\ t_1 < t_2\ \wedge \neg Declipped(t_1, f, t_2) \tag{EC4}$$

These axioms state the effects of actions and persistence of truth values for fluents. Note that axioms (EC3) and (EC4) are needed only for time points when the fluents have not yet changed value. If an initiating or terminating action happens then the persistence of truth values and effects of actions will be taken care of by (EC1) and (EC2). As a result, for domains with non-negative time we can replace (EC3) and (EC4) with the following:

$$HoldsAt(f, t) \leftarrow Initially_P(f) \wedge \neg Clipped(0, f, t) \tag{EC3$'$}$$

$$\neg HoldsAt(f, t) \leftarrow Initially_N(f) \wedge \neg Declipped(0, f, t) \tag{EC4$'$}$$

$$Initially_P(f) \vee Initially_N(f) \tag{EC5}$$

where $Initially_P(f)$ and $Initially_N(f)$ formalize the initial state of the world (whether f is true at time 0 or not).

Domain-Specific Axioms

Domain-specific axioms define the predicates $Initiates, Terminates$ and $Happens$.

After writing domain-dependent axioms, we need to do *minimization* on these axioms. This basically boils down to making the one-sided conditionals into bi-conditionals. This will be explained in the following example. The minimization of $Initiates$ and $Terminates$ captures the assumption that "actions have no unexpected effects", and the minimization of $Happens$ captures the assumption that "there are no unexpected event occurrences"[18].

Consider the following example which is a shorter version of an example from [17]. There is a robot that can lock or unlock a door by inserting a key card. We have fluents $hasKey$ and *locked*, and actions *pickup* and *insert*. We assume we have unique name axioms stating

[4]Traditionally, in Situation Calculus conditionals are represented using $\supset$ symbol and in the Event calculus using backward arrows ($\leftarrow$) where consequents are written first (in the style of logic programming).

these constant symbols refer to different objects. We have the following axioms for $Initiates$ and $Terminates$:

$$Initiates(pickup, hasKey, t) \tag{2.6}$$

$$Initiates(insert, locked, t) \leftarrow \neg HoldsAt(locked, t) \wedge HoldsAt(hasKey, t) \tag{2.7}$$

$$Terminates(insert, locked, t) \leftarrow HoldsAt(locked, t) \wedge HoldsAt(hasKey, t) \tag{2.8}$$

and the following axioms:

$$HoldsAt(locked, 0), Happens(pickup, 2), Happens(insert, 4)$$

Applying minimization to $Initiates, Terminates$ and $Happens$ gives

$$\begin{aligned} Initiates(a, f, t) \equiv\ & (a = pickup \wedge f = hasKey) \\ & \vee (a = insert \wedge f = locked \wedge \neg HoldsAt(locked, t) \wedge HoldsAt(hasKey, t)) \\ Terminates(a, f, t) \equiv\ & (a = insert \wedge f = locked \wedge HoldsAt(locked, t) \wedge HoldsAt(hasKey, t)) \\ Happens(a, t) \equiv\ & (a = pickup \wedge t = 2) \vee (a = insert \wedge t = 4) \end{aligned}$$

From the above axioms it follows that for example $\neg HoldsAt(locked, 5)$ holds.

It is worth noting that while in the Situation Calculus we need to axiomatize in what situations and under what conditions fluents are true (using successor state axioms), in the Event Calculus in contrast we axiomatize how and when fluents *change value* (using $Initiates$ and $Terminates$ predicates). It is interesting that such a seemingly minor difference makes the two formalisms develop in different paths.

2.3 Modal Logic

Modal logic is a formalism that extends propositional and first order logic by adding operators to the language to express modalities (e.g. epistemic, doxastic, temporal, ...). Here we briefly review propositional modal logic based on [3] and [4].

2.3.1 Syntax

Given a set of modality symbols MOD, the language of propositional modal logic $\mathcal{L}_M$ is inductively defined using the following BNF where $m \in$ MOD:

$$\varphi ::= p \mid \top \mid \bot \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \Box_m \varphi$$

The denotation of $\Box_m\varphi$ depends on the context. For example in the context of reasoning about beliefs, $\Box_m\varphi$ denotes "agent m believes that φ". The symbol $\Diamond$ is defined as $\Diamond\varphi \equiv \neg\Box\neg\varphi$.

2.3.2 Semantics

The semantics of modal logic is based on the concept of possible worlds. These worlds comprise of the actual world and alternative worlds representing different ways the world could have been. The idea of possible worlds dates back to the German philosopher Gottfried Leibniz but its usage in modal logic and development of possible world semantics (described below) is attributed to Saul Kripke.

Given a set W of possible worlds and a set $\{R^m\}_{m\in\text{MOD}}$ of the accessibility relations between worlds, a *frame* is a tuple $\langle W, \{R^m\}_{m\in\text{MOD}}\rangle$. A *(Kripke) model* M is a triple $\langle W, \{R^m\}_{m\in\text{MOD}}, V\rangle$ where $\langle W, \{R^m\}_{m\in\text{MOD}}\rangle$ is a frame and V is a valuation function, mapping each propositional variable to a subset W of worlds in which p is true. If $|\text{MOD}| = 1$, we write R instead of $\{R^m\}_{m\in\text{MOD}}$ and $\Box$ instead of $\Box_m$.

Given a model $M = \langle W, \{R^m\}_{m\in\text{MOD}}, V\rangle$ and $w \in W$, definition of φ being *true* (or *satisfied*) in M at w is as follows:

$$
\begin{array}{lll}
M, w \vDash p & \text{iff} & w \in V_p \\
M, w \vDash \neg p & \text{iff} & M, w \nvDash p \\
M, w \vDash \varphi \wedge \psi & \text{iff} & M, w \vDash \varphi \text{ and } M, w \vDash \psi \\
M, w \vDash \Box_m\varphi & \text{iff} & \text{for all } v \in W \text{ such that } wR^m v \text{ we have } M, v \vDash \varphi
\end{array}
$$

A formula φ is *true* in M, written $M \vDash \varphi$, if $M, w \vDash \varphi$ for all $w \in W$. A formula is *valid*, written $\vDash \varphi$, if it is true at every model.

2.4 Epistemic Situation Calculus

2.4.1 The Standard Model

Formalizing knowledge is an important aspect of reasoning about change. The standard approach to this problem in the situation calculus, based on works of Moore ([19], [20]), provides a possible-world account of knowledge by using situations to represent possible worlds. There have been several variants of formalization of knowledge in the Situation Calculus based on this approach ([27], [2], [13], [24], [25]). These approaches differ in how they formalize the accessibility relation. The most recent of these approaches (i.e. [25])) has been used as the standard representation in the literature. In the sequel we review epistemic Situation Calculus based on this approach.

The accessibility relation is represented by a relational fluent K. $K(s', s)$ denotes "situation s' is accessible from situation s". Knowledge is then defined naturally using the K fluent:

$$\mathbf{Knows}(\phi, s) \stackrel{def}{=} (\forall s').K(s', s) \supset \phi[s']$$

where ϕ is an expression derived from a formula φ where all situation arguments are suppressed and $\phi[s]$ denotes φ.

We have the following definition:

$$\mathbf{KWhether}(\phi, s) \stackrel{def}{=} \mathbf{Knows}(\phi, s) \vee \mathbf{Knows}(\neg\phi, s)$$

There are two kinds of actions:

- ordinary/physical actions that make changes in the world,
- knowledge-producing actions or sensing actions that make changes in the state of mind of agents. These actions inform the agent about the truth value of a relational fluent or value of a functional fluent.

A key part in formalizing knowledge is to define how the accessibility relation changes as a result of an action. This is done using the following successor state axiom for fluent K:

$$\begin{aligned} K(s'', do(a, s)) \equiv (\exists s').s'' = do(a, s') \wedge K(s', s) \\ \wedge Poss(a, s') \wedge \textsc{sr}(a, s) = \textsc{sr}(a, s') \end{aligned} \tag{2.9}$$

where SR is called the *sensing result function* and is defined as:

$$\begin{aligned} \textsc{sr}(sense_Q, s) = r \equiv (r = \text{“Yes”} \wedge Q(s)) \vee (r = \text{“No”} \wedge \neg Q(s)) \\ \textsc{sr}(read_t, s) = r \equiv r = t(s) \end{aligned}$$

For example if $holding(x, s)$ is a relational fluent and $info(x, s)$ is a functional fluent, then we have

$$\begin{aligned} \textsc{sr}(sense_{holding(x)}, s) = r \equiv (r = \text{“Yes”} \wedge holding(x, s)) \vee (r = \text{“No”} \wedge \neg holding(x, s)) \\ \textsc{sr}(read_{info(x)}, s) = r \equiv r = info(x, s) \end{aligned}$$

For physical actions, the result of SR is defined to be always the same. So for an ordinary action a, it can be defined as for example $\textsc{sr}(a, s) = r \equiv r = \text{“ok”}$.

The authors then make the following *axiomatization policy*: Actions are to be axiomatized as affecting only either the K fluent or other fluents. In other words Knowledge-producing actions only affect the K fluent, and physical actions only affect other fluents.

They argue that this assumption does not limit the capabilities of the formalism as any action that affects both the K fluent and other fluents can be replaced by two actions where one affects only the K fluent and the other affects only the world. They give the example of the action *openBag* which opens the bag and makes the contents of the bag known to the agent. They suggest this action can be replaced by two actions *open* (which only opens the bag) and a sense action (which makes the contents known to the agent).

Foundational Axioms for Epistemic Situations Calculus

Since in the epistemic Situation Calculus we consider more than one initial situation, the sort $Init$ is introduced which contains S_0 and the situations which are K-accessible from an $Init$ situation.[5] Nothing else is in $Init$. $Init(s)$ is defined as follows

$$Init(s) \stackrel{def}{=} \neg(\exists a, s')s = do(a, s').$$

The following five axioms form the new set of foundational axioms.

$$do(a_1, s_1) = do(a_2, s_2) \supset a_1 = a_2 \wedge s_1 = s_2,$$
$$s \sqsubset do(a, s') \equiv s \sqsubset s'.$$
$$Init(s') \supset \neg s \sqsubset s',$$
$$(\forall P).(\forall s)[Init(s) \supset P(s)] \wedge (\forall a, s)[P(s) \supset P(do(a, s))] \supset (\forall s)P(s).$$
$$K(s, s') \supset [Init(s) \equiv Init(s')]$$

The first two are the same as before. The second two are the generalized form of their counterparts for more than one initial situation. The last one states that only initial situations can be K-related to an initial situation.

We have also the following theorem stating properties of the accessibility relation are preserved over executable situations:

Theorem 2.4.1. *([25]) If the K relation on the set of initial situations is restricted to conform to the reflexive condition along with some subset of the symmetric, transitive and Euclidean properties, then the K relation at every situation, resulting from the execution of a sequence of possible actions (as defined by Poss), will satisfy the same set of properties.*

2.4.2 Extensions

The standard model discussed above is limited to a single agent, which has perfect sensors, and whose actions are sequential and deterministic. There has been a significant number of papers in the literature relaxing these assumptions. For example, in [26] the standard

[5]Introducing this sort is not necessary for defining knowledge, but it is useful for reasoning about properties of the formalism.

formalization has been extended with time and *concurrent actions*. [2] generalizes an earlier version of the standard framework [27] to accommodate *noisy sensors*. [1] extends the above formalism to accommodate *noisy effectors* (i.e. noise in the effect of physical actions), which formalizes non-deterministic actions. While [1] can be regarded as a quantitative characterization of non-deterministic actions, [6] gives a qualitative account of non-determinism (but with perfect sensors i.e. no noise). [29] extends the standard framework to account for multiple agents. Considering the multi-agent case brings up other questions such as whether the environment is fully observable (i.e. each agent is aware of all the actions performed by all the other agents) and whether the domain is synchronized (i.e. agents always know when an action has occurred). [10] and [11], among others, discuss these issues.

2.5 Epistemic Event Calculus

The possible-world approach to formalization of knowledge is an old idea which dates back to introduction of Kripke models in the Sixties, and can still be considered among the more intuitive approaches (though not computationally very efficient).

Since Situation Calculus has a branching model of time, for a possible-world account of knowledge, situations are good candidates to be regarded as possible worlds. But te Event Calculus has a linear model of time, and therefore does not have any 'natural' candidates for representing possible worlds for a possible-world account of knowledge. As a result treatment of knowledge in the Event Calculus has been different from the Situation Calculus. We have to formalize knowledge in a more 'direct' way, something like:

$$Initiates(sense_f, Knows(f), t) \leftarrow HoldsAt(f, t)$$

where knowledge production is formalized directly without using possible-worlds. The problem is that formulas like the one above are not well-formed; if we consider *Knows* as a predicate then it cannot appear as an argument of *Initiates*, and if we consider it as a function then it is not clear how we can interpret it (the denotation needs to be an object in the domain).

In general for adding belief or knowledge to first-order logic we have a few options. Consider for example the sentence "John knows that block A is on the table". One approach is to extend the language of FOL with a modal operator, and use the possible world semantics on top of Tarskian semantics of FOL [5]. In this approach we can formalize the above sentence as $\Box(on(A, Table))$. We can avoid using modal operators by formalizing possible worlds in the object language (first introduced by [19]) as done in the Situation Calculus, i.e. formalizing the accessibility relation in the language using a fluent (K) and using situations

as possible worlds. Knowledge is then an abbreviation for a formula in the language:

$$Knows(on(A, Table), s) \doteq (\forall s').\ K(s', s) \supset on(A, Table, s')$$

There is another approach which avoids possible worlds and is adopted in formalization of knowledge in the Event Calculus. This approach is based on the formalism developed by Morgenstern in [21]. She developed a sorted FOL L which has the sort *str* (among other sorts) for string objects. For any expression ϕ, its string counterpart 'ϕ' is an object of sort *str*. For example 'FatherOf(John, Bill)' represents the formula *FatherOf(John,Bill)*. This makes it possible to practically use formulas as arguments in predicates, while remaining in FOL. This solves the problem mentioned above.

The general term for this approach is *syntactic theory* which dates back further in the past. The problem with this approach is that it is too strong. Strings make it possible to talk "about expressions of L *within* L" [21]. As a result we can formalize self-referential statements as well (e.x. liar's paradox), and this leads to paradoxes. There have been several solutions proposed to resolve this issue, but we don't go further into details.

In [7] the authors provide a "meta-language representation" of knowledge in the Event Calculus, utilizing the third approach describe above. They introduce the following axioms (note that the first argument in the following axioms is actually the string representation of the formula, omitted for simplicity)

$$HoldsAt([k(f_1 \rightarrow f_2) \rightarrow (k(f_1) \rightarrow k(f_2))], t) \tag{K1}$$
$$HoldsAt(k(f) \rightarrow f, t) \tag{K2}$$
$$HoldsAt(k(f) \rightarrow k(k(f)), t) \tag{K3}$$
$$HoldsAt([\forall x\ k(f) \rightarrow k(\forall x\ f)], t) \tag{K4}$$
$$\text{If } f \text{ is a classical tautology then } HoldsAt(k(f), t) \tag{K5}$$
$$HoldsAt(kw(f) \equiv k(f) \vee k(\neg f), t) \tag{K6}$$

The first three are counterparts of axioms K, T and 4 in modal logic. (K5) is a weaker version of the necessitation rule (not all axioms are known; only tautologies). There are also the following axioms

$$HoldsAt(f_1 \rightarrow f_2, t) \rightarrow (HoldsAt(f_1, t) \rightarrow HoldsAt(f_2, t))$$
$$\forall x\ HoldsAt(f_1, t) \rightarrow HoldsAt(\forall x\ f_1, t)$$
$$HoldsAt(\neg f, t) \equiv \neg HoldsAt(f, t)$$

Sensing actions produce knowledge:

$$Initiates(sense(f), kw(f), t)$$

In other words, after sensing a fluent, the agent knows whether the fluent is true or not.

There has been other approaches in formalization of knowledge in the Event Calculus (e.g. [14], [22]), but we don't go further into details.

Chapter 3

The Problem with the Standard Formalization

If we look at different proposals for the successor state axiom of fluent K in the Situation Calculus; we see that there has been an uncertainty on how to take into account possibility of performing actions (I have applied the terminology of [25] to all proposals for ease of comparison):

(Scherl and Levesque, 1993) [27]

$$\boldsymbol{Poss(a,s)} \supset K(s'', do(a,s)) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \wedge \text{SR}(a,s) = \text{SR}(a,s') \quad (3.1)$$

(Bacchus et al., 1995) [2]

$$\begin{aligned} \boldsymbol{Poss(a,s)} \supset K(s'', do(a,s)) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \\ \wedge \boldsymbol{Poss(a,s')} \wedge \text{SR}(a,s) = \text{SR}(a,s') \end{aligned} \quad (3.2)$$

(Levesque et al., 1998) [13]

$$\begin{aligned} K(s'', do(a,s)) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \\ \wedge \boldsymbol{Poss(a,s) \equiv Poss(a,s')} \wedge \text{SR}(a,s) = \text{SR}(a,s') \end{aligned} \quad (3.3)$$

(Scherl and Levesque, 2003) [25]

$$\begin{aligned} K(s'', do(a,s)) \equiv (\exists s').s'' = do(a,s') \wedge K(s',s) \wedge \boldsymbol{Poss(a,s')} \\ \wedge \text{SR}(a,s) = \text{SR}(a,s') \end{aligned} \quad (3.4)$$

The following toy example shows that none of these proposals work properly when unexecutable actions (i.e. those actions whose preconditions are not met in a situation) are involved.

Example 1. Suppose we have a robot that has one arm and can pick up objects. Suppose block A is on the floor and the agent doesn't know whether it is glued to the floor or not.

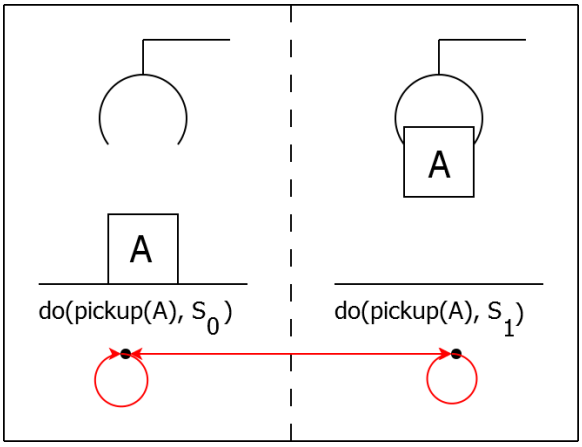

Figure 3.1: Desired accessibility relations and possible worlds after action $pickup(A)$

So we have two initial situations which the agent cannot distinguish between; one where A is glued to the floor (S_0), and one where it isn't (S_1). Suppose further that A actually is glued to the floor. Now we want to reason about what the agent believes after the action $pickup(A)$. We expect from a 'correct' formalization of the problem that after the action, the agent be uncertain between two possible worlds: One where A is still glued to the floor, and one where the agent is holding the block A.

But applying the above successor state axioms we see that after the action $pickup(A)$ either the accessibility relation is silent about truth value of $K(s'', do(pickup(A, S_0)))$ (3.1, 3.2), or the agent believes that it is holding block A (3.3, 3.4). And none of these are the desired results. Formally

Action Precondition Axiom:

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

Successor State Axioms:

$$holding(x, do(a, s)) \equiv a = pickup(x) \vee holding(x, s)$$

$$gluedToFloor(x, do(a, s)) \equiv gluedToFloor(x, s)$$

Initial Situations:

$$(\forall x) \neg holding(x, S_0), gluedToFloor(Obj, S_0),$$

$$(\forall x) \neg holding(x, S_1), \neg gluedToFloor(Obj, S_1)$$

Initial Accessibility Relations:

$$K(S_0, S_0), K(S_0, S_1), K(S_1, S_0), K(S_1, S_1)$$

Depending on which successor state axiom we opt for, the above formalization yields the following results on beliefs of agent in $do(pickup(A), S_0)$:

(3.1) and (3.2) are silent on whether $Knows(holding(A), do(pickup(A), S_0))$ holds (because of $\neg Poss(pickup(A, S_0))$).

From (3.3) and (3.4) we can deduce that both of the following formulas hold

$$Knows(holding(A), do(pickup(A), S_0))$$

$$Knows(holding(A), do(pickup(A), S_1))$$

In other words the agent believes that it is holding A after the action $pickup(A)$ in S_0, which should not be the case.

The reason that none of the above proposals works in general, is that using ghost situations to represent possible worlds causes the agent to obtain incorrect beliefs about the domain. Recall from the background section the form of successor state axioms:

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

We can see that in this form, truth or falsity of a fluent after an action is independent of whether the action is executable or not. For example for the fluent $holding(x, s)$ we have

$$holding(x, do(a, s)) \equiv a = pickup(x) \vee [holding(x) \wedge a \neq drop(x)]$$

Note that $holding(x, do(pickup(x), s))$ is always true for all x and s, no matter whether executing the action $pickup(x)$ is possible in s or not.

As a result, the fluents that hold in a situation are independent of whether the actions comprising the situation are executable or not. If the actions comprising a situation are all executable then the resulting situation is an executable situation. But if any of such actions is not executable then the resulting situation is a ghost situation. While ghost situations are perfectly fine situations, they are not physically realizable. Therefore using them as possible worlds causes the agent to have incorrect beliefs about the domain.

In fact the problem with unexecutable actions is not limited to formalization of knowledge. There are also issues in dealing with the *projection problem*:

> Given an initial situation, a sequence of actions $a_1, \ldots, a_n$, and a formula G, determine whether G is true in the resulting situation.

We can solve this problem in the traditional framework only if the resulting situation is executable (i.e. all the actions in the action sequence are possible to be executed). In this case the projection problem is equivalent to whether $Axioms \models G(do([a_1, \ldots, a_n], S_0)$[1], and we can use Regression to answer it.

[1] $do([a_1, \ldots, a_n], s)$ abbreviates $do(a_n, do(a_{n-1}, \ldots do(a_1, s) \ldots))$

To see why this doesn't work when unexecutable actions are involved, consider the following example.

Example 2. Suppose we have a robot that has one arm and can pick up objects and move. Agent is ordered to pickup an object from room 1 and move to the next room. But the pickup is not possible, say as a result of the object being glued to the floor. In the current framework, following the pickup-and-move sequence, in the resulting situation the agent is holding the object and is in the next room (which is impossible), as formalized below:

We have fluents $holding(x, s)$ denoting "the robot is holding object x, in situation s", $gluedToFloor(x, s)$ with the obvious intended meaning and $loc(x, y, s)$ denoting "location of x is y, in situation s".

We have two actions $pickup(x)$ and $moveTo(x)$ denoting the actions of picking up object x and going to location x, respectively. $R1$ and $R2$ denote room 1 and room 2.

Action Precondition Axioms

$$Poss(pickup(x), s) \equiv \neg gluedToFloor(x) \wedge (\forall y)\neg holding(y, s)$$
$$Poss(moveTo(x), s) \equiv True$$

Successor state axioms:

$$holding(x, do(a, s)) \equiv a = pickup(x) \vee holding(x, s)$$
$$gluedToFloor(x, do(a, s)) \equiv gluedToFloor(x, s)$$
$$loc(x, y, do(a, s)) \equiv [a = moveTo(y) \wedge (holding(x, s) \vee x = Robot)]$$
$$\vee [loc(x, y, s) \wedge ((\neg\exists z \neq y).a = moveTo(z) \vee \neg holding(x, s))$$

The initial situation:

$$(\forall x)\neg holding(x, S_0), gluedToFloor(Obj, S_0), loc(Obj, R1, S_0), loc(Robot, R1, S_0).$$

The following formulas are entailed by the formalization:
$holding(Obj, do([pickup(Obj), moveTo(R2)], S_0)$
$loc(Obj, R2, do([pickup(Obj), moveTo(R2)], S_0)$
$loc(Robot, R2, do([pickup(Obj), moveTo(R2)], S_0)$

This is clearly impossible and the (separate) limitation of situations to executable ones rules out such a (ghost) situation. As a result we cannot correctly formalize these scenarios in the traditional framework. Clearly a more desirable outcome of this pickup-and-move sequence is that the agent is in the next room and the object's location is unchanged. This will be the result in our proposed framework.

3.1 Why this Form for Successor State Axioms?

In the previous section we observed that the current form of successor state axioms does not give the desired result when unexecutable actions are involved, as a result of being insensitive to possibility of actions. In this section we explore the roots of this form and review how it was developed as a solution to the Frame Problem. This will be helpful when we provide our framework and show how it solves the aforementioned problems.

First introduced by McCarthy and Hayes in the late Sixties [16], the *frame problem* is the problem of formalizing effects of an action without explicitly formalizing a large number of fluents that are not affected by it. For example for the action $pickup(x)$ we have

$$color(x, s) = c \supset color(y, do(pickup(x), s))$$

meaning picking up an object does not have an effect on its color. These kinds of axioms are called *frame axioms*. Naively writing frame axioms requires one axiom for each pair of action and fluent. So the total number of axioms needed would be $O(\mathcal{F} \times \mathcal{A})$, where $\mathcal{A}$ is the number actions and $\mathcal{F}$ the number of fluents. This is expensive and not desirable. In response to this, in [23], Reiter presented a solution to the frame problem building on the previous solutions provided by Davis, Haas and Schubert [28]. The following is a review of a more recent version of the solution taken from [24].

3.1.1 Review of Reiter's solution to the Frame Problem

The basic idea is illustrated using the following example which is a simplified version of one from [24]. There is a robot that can pick up, drop or hold objects. The formalization of the problem is as follows:

Action Precondition Axioms:

$$Poss(drop(x), s) \equiv holding(x, s)$$
$$Poss(repair(x), s) \equiv hasGlue(s) \wedge broken(x, s)$$

A set of *Effect Axioms* are used to formalize the effects of actions (effect axioms were being used before developing successor state axioms). For the fluent $broken(x, s)$ the effect axioms are as follows:

$$fragile(x, s) \supset broken(x, do(drop(x), s)) \tag{3.5}$$
$$\supset \neg broken(x, do(repair(x), s)) \tag{3.6}$$

We can rewrite (3.5) and (3.6) in the logically equivalent form:

$$(a = drop(x) \wedge fragile(x, s)) \supset broken(x, do(a, s)) \tag{3.7}$$

$$a = repair(x) \supset \neg broken(x, do(a, s)) \tag{3.8}$$

Reiter then employs a *completeness assumption*, that formulas (3.7) and (3.8) "characterize all the conditions under which fluent *broken* can become true or false" as a result of action a. In other words we assume all the different ways that the fluent $broken(x, s)$ can become true or false are captured in the effect axioms.

So if the truth value of *broken* changes, for example $\neg broken(x, s)$ and $broken(x, do(a, s))$ are true, this change must have occurred because the following was true:

$$(a = drop(x) \wedge fragile(x, s))$$

So we have the following *explanation closure axioms* [28]:

$$\neg broken(x, s) \wedge broken(x, do(a, s)) \supset (a = drop(x) \wedge fragile(x, s))$$
$$broken(x, s) \wedge \neg broken(x, do(a, s)) \supset a = repair(x)$$

For the general case, the effect axioms of a domain for a fluent F can be written in the form

$$\gamma_F^+(\vec{x}, a, s) \supset F(\vec{x}, do(a, s))$$
$$\gamma_F^-(\vec{x}, a, s) \supset \neg F(\vec{x}, do(a, s))$$

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

So explanation closure axioms resulting from the completeness assumption are of the form:

$$F(\vec{x}, s) \wedge \neg F(\vec{x}, do(a, s)) \supset \gamma_F^-(\vec{x}, a, s)$$
$$\neg F(\vec{x}, s) \wedge F(\vec{x}, do(a, s)) \supset \gamma_F^+(\vec{x}, a, s)$$

We have two axioms for each fluent. Note that explanation closure axioms yield the general form of frame axioms:

$$F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s) \supset F(\vec{x}, do(a, s)) \tag{3.9}$$

$$\neg F(\vec{x}, s) \wedge \neg\gamma_F^+(\vec{x}, a, s) \supset \neg F(\vec{x}, do(a, s)) \tag{3.10}$$

Therefore we need only $O(\mathcal{F})$ axioms to produce all frame axioms, and this provides a solution to the frame problem.

Reiter then introduced successor state axioms which merge effect and explanation closure axioms:

$$F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a, s) \vee [(F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s))] \tag{3.11}$$

3.2 Does the Event Calculus Suffer from a Similar Problem?

The problems described in the previous section were due to the fact that although action preconditions are part of the axiomatization, successor state axioms were not sensitive to possibility of actions. So a question that arises is that how does the Event Calculus deal with action preconditions?

In the Event Calculus literature there is very little discussion of action preconditions. In [12], which introduced the Event Calculus, they ignored handling the precondition of events and assumed all events are possible at any time. The first treatment of preconditions of actions that I found was in [17]. They make a distinction between *fluent preconditions* and *action preconditions*. For instance in the example

$$Initiates(insert, locked, t) \leftarrow \neg HoldsAt(locked, t) \wedge HoldsAt(hasKey, t)$$

they regard the right hand side as fluent preconditions. They then introduce the predicate $Impossible$ to characterize conditions under which actions are not possible. For example

$$Impossible(pickup, t) \leftarrow \neg HoldsAt(HasGrabber, t)$$

They state that this type of knowledge can be interpreted in two ways. Based on one reading of $Impossible(A, T)$, the effects of attempting to perform 'impossible' action A at time T is undefined (this reading can be enforced by a set of axioms). This reading is similar to the [23] version of the Situation Calculus where successor state axioms are of the form

$$Poss(a, s) \rightarrow \{F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a, s) \vee [(F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s))]\}$$

Note that in this version the truth value of F in $do(a, s)$ is undefined if performing a is not possible in s (i.e. $\neg Poss(a, s)$).

Based on another reading of $Impossible(A, T)$, the effects of attempting to perform 'impossible' action A at time T is that nothing happens. This can be captured by

$$Happens(a, t) \equiv [(perform(a, t) \wedge \neg Impossible(a, t)) \vee Occured(a, t)]$$
$$\neg Occurred(a, t) \leftarrow Impossible(a, t)$$

where $Occurred$ represents actions that have happened in the past and $perform$ represents actions that "the agent will (possibly) perform in the future"[18].

In the latter reading of $Impossible$, it seems more intuitive that instead of incorporating $Impossible$ into $Happens$ (as described above), we incorporate it into $Initiates$ and $Terminates$ predicates. In this case the reading of $Happens(a, t)$ becomes "an attempt to execute action a is done at time t".

3.3 A Note on the Axiomatization Policy

In section 2.4.1 when reviewing the standard model for epistemic Situation Calculus we mentioned that there is an assumption (they called it 'axiomatization policy') that "Knowledge-producing actions only affect the K fluent, and physical actions only affect other fluents". One place that this assumption can be utilized is the following theorem:

Theorem 3.3.1. *For relational fluent P the following holds:*

$$\textbf{\textit{KWhether}}(P, do(sense_P, s))$$

Proof. For the sake of obtaining a contradiction assume that (i) does not hold. Therefore based on definition of **KWhether** we have $\neg$**Knows**$(P, do(sense_P, s)) \wedge \neg$**Knows**$(\neg P, do(sense_P, s))$. In addition, in a given situation, a fluent is either true of false. So we have $P(s) \vee \neg P(s)$. We prove that $P(s)$ together with $\neg$**Knows**$(P, do(sense_P, s))$ leads to contradiction. A similar argument proves that $\neg P(s)$ together with $\neg$**Knows**$(\neg P, do(sense_P, s))$ leads to contradiction; and this completes the proof.

The proof is as follows

1) $P(s)$	Premise
2) $\neg\mathbf{Knows}(P, do(sense_P, s))$	Premise
3) $\exists s'.K(s', do(sense_P, s)) \wedge \neg P(s')$	(2), definition of $Knows$
4) $\exists s''.s' = do(sense_P, s'')$ $\wedge \text{SR}(sense_P, s) = \text{SR}(sense_P, s'')$	(3), successor state axiom of K fluent
5) $\neg P(do(sense_P, s''))$	(3), (4)
6) $\neg P(s'')$	(5), axiomatization policy
7) $\text{SR}(sense_P, s) \neq \text{SR}(sense_P, s'')$	(1), (6), definition of SR
8) $\text{SR}(sense_P, s) = \text{SR}(sense_P, s'')$	(4)
9) Contradiction	(7), (8) ■

Although the axiomatization policy can be used to derive intuitive results, we believe that it is needlessly restrictive. It involves two assumptions: 1) Physical Actions does not affect the K fluent, 2) Sensing actions can only affect the K fluent. Relaxing the first part requires rewriting the successor state axiom for fluent K by taking physical actions into consideration, which we do not do here. But we believe that the second part can be relaxed to some extent.

Obviously in real scenarios there are actions that are both informative and make changes in the world. The solution that was provided was to consider such actions as two actions, one sensing and one physical. But this doesn't seem to be very intuitive. For example consider a tasting action to determine the sweetness of a jar of honey. This is a sensing action, but it make changes to honey as well (e.x. affecting the amount of honey left). Splitting this action into two actions does not seem to capture the essence of this action. A better solution is to allow sensing actions to make changes in the world as well, and as a result take them into account when writing successor state axioms for all fluents (not just the K fluent). One problem with this approach is that we cannot prove some intuitive results such as the theorem above. Our solution to this problem is that we limit effect of sense actions in the following narrow way: Sensing for a fluent F does not change value of F (but it may change value of other fluents). For example tasting a jar of honey to determine how sweet it is does not affect its actual sweetness, but it can affect other fluents such as the amount of honey left. This way we allow sensing actions to affect other fluents and at the same time we are able to derive the intuitive results such as the theorem above. We believe that this approach not only makes a weaker assumption, but also it is more natural and intuitive.

Chapter 4

Proposed Framework

We saw in chapter 3 that insensitivity of successor state axioms to possibility of actions produced some problems. In section 3.1 we traced the source of the problem to Reiter's solution to the frame problem. In this section we provide a more general solution to the frame problem which results in a more general form for successor state axioms. The new form does not suffer from the aforementioned limitations.

4.1 A More General Solution to the Frame Problem

Our approach to the frame problem differs from Reiter's in the following way: Intuitively, if an action is not possible in a situation, then "executing" it should not make any changes in the model[1]. Therefore other than axioms (3.9) and (3.10), we believe that the following forms should be regarded as frame axioms as well:

$$F(\vec{x}, s) \wedge \neg Poss(a, s) \supset F(\vec{x}, do(a, s))$$
$$\neg F(\vec{x}, s) \wedge \neg Poss(a, s) \supset \neg F(\vec{x}, do(a, s))$$

To produce these extra frame axioms we need a generalized form of explanation closure axioms. Since these axioms were derived from effect axioms let's consider effect axioms where possibility of actions is taken into account. For example for the action $drop(x)$ (after rewriting) we have

$$a = drop(x) \wedge fragile(x, s) \wedge Poss(a, s) \supset broken(x, do(a, s))$$
$$a = repair(x) \wedge Poss(a, s) \supset \neg broken(x, do(a, s))$$

[1]There are very special cases of fluents that may change value even if the action is not possible (e.x. the functional fluent $NumberOfActionsAttemptedSoFar(s)$). We assume we don't have these kinds of fluents in our models.

From the completeness assumption we get the following explanation closure axioms:

$$\neg broken(x,s) \wedge broken(x, do(a,s)) \supset a = drop(x) \wedge fragile(x,s) \wedge Poss(a,s)$$
$$broken(x,s) \wedge \neg broken(x, do(a,s)) \supset a = repair(x) \wedge Poss(a,s)$$

So in the general case we have the following form for effect axioms

$$Poss(a,s) \wedge \gamma_F^+(\vec{x},a,s) \supset F(\vec{x}, do(a,s)) \tag{4.1}$$
$$Poss(a,s) \wedge \gamma_F^-(\vec{x},a,s) \supset \neg F(\vec{x}, do(a,s)) \tag{4.2}$$

and the following form for explanation closure axioms:

$$F(\vec{x},s) \wedge \neg F(\vec{x}, do(a,s)) \supset \gamma_F^-(\vec{x},a,s) \wedge Poss(a,s) \tag{4.3}$$
$$\neg F(\vec{x},s) \wedge F(\vec{x}, do(a,s)) \supset \gamma_F^+(\vec{x},a,s) \wedge Poss(a,s) \tag{4.4}$$

Note that explanation closure axioms yield all the frame axioms:

$$F(\vec{x},s) \wedge \neg Poss(a,s) \supset F(\vec{x}, do(a,s))$$
$$\neg F(\vec{x},s) \wedge \neg Poss(a,s) \supset \neg F(\vec{x}, do(a,s))$$
$$F(\vec{x},s) \wedge \neg \gamma_F^-(\vec{x},a,s) \supset F(\vec{x}, do(a,s))$$
$$\neg F(\vec{x},s) \wedge \neg \gamma_F^+(\vec{x},a,s) \supset \neg F(\vec{x}, do(a,s))$$

So explanation closure axioms (4.3) and (4.4) qualify as a solution to the more general form of frame problem.

4.2 A More General Form for Successor State Axioms

In this section we develop a more general form for successor state axioms where possibility of actions is taken into account.

Theorem 4.2.1. *Let* Γ *be the union of axioms* (4.1) $-$ (4.4) *and the formula* $\neg(\exists \vec{x}, a, s).(\gamma_F^+(\vec{x},a,s) \wedge \gamma_F^-(\vec{x},a,s))$. *Then* Γ *entails*

$$F(\vec{x}, do(a,s)) \equiv (\gamma_F^+(\vec{x},a,s) \wedge Poss(a,s)) \vee [F(\vec{x},s) \wedge (\neg\gamma_F^-(\vec{x},a,s) \vee \neg Poss(a,s))] \tag{4.5}$$

Proof: For convenience let $P, \gamma^+, \gamma^-, F_1, F_2$ denote $Poss(a,s)$, $\gamma_F^+(\vec{x},a,s)$, $\gamma_F^-(\vec{x},a,s)$, $F(\vec{x},s)$, $F(\vec{x}, do(a,s))$ respectively.

Rewriting $4.1 - 4.4$ using this notation yields

$$P \wedge \gamma^+ \supset F_2 \tag{4.6}$$
$$P \wedge \gamma_- \supset \neg F_2 \tag{4.7}$$
$$F_1 \wedge \neg F_2 \supset \gamma^- \wedge P \tag{4.8}$$
$$\neg F_1 \wedge F_2 \supset \gamma^+ \wedge P \tag{4.9}$$

Rewriting using logically equivalent formulas yields

$$P \wedge \gamma^+ \supset F_2 \tag{4.10}$$
$$\neg P \vee \neg\gamma^- \subset F_2 \tag{4.11}$$
$$F_1 \wedge (\neg\gamma^- \vee \neg P) \supset F_2 \tag{4.12}$$
$$F_1 \vee (\gamma^+ \wedge P) \subset F_2 \tag{4.13}$$

Rewriting using logically equivalent formulas yields

$$\text{From } (4.10), (4.12) \qquad (P \wedge \gamma^+) \vee (F_1 \wedge (\neg\gamma^- \vee \neg P)) \supset F_2 \tag{4.14}$$
$$\text{From } (4.11), (4.13) \qquad (\neg P \vee \neg\gamma^-) \wedge (F_1 \vee (\gamma^+ \wedge P)) \subset F_2 \tag{4.15}$$

Left hand side of (4.14) and (4.15) are equal:

$$\begin{aligned}
\text{LHS of } (4.14) &= (P \wedge \gamma^+) \vee (F_1 \wedge (\neg\gamma^- \vee \neg P)) \\
&\equiv ((P \wedge \gamma^+) \vee (F_1)) \wedge ((P \wedge \gamma^+) \vee (\neg\gamma^- \vee \neg P)) \\
&\equiv ((P \wedge \gamma^+) \vee (F_1)) \wedge (\neg\gamma^- \vee \neg P \vee \gamma^+) \\
&\qquad\qquad \wedge (\neg\gamma^- \vee \neg P \vee P) \\
&\equiv ((P \wedge \gamma^+) \vee (F_1)) \wedge (\neg\gamma^- \vee \neg P \vee \gamma^+) \wedge T
\end{aligned}$$

Given $\neg(\gamma^+ \wedge \gamma^-)$

$$\begin{aligned}
&\equiv ((P \wedge \gamma^+) \vee (F_1)) \wedge (\neg\gamma^- \vee \neg P) \\
&= \text{LHS of } (4.15)
\end{aligned}$$

Therefore

$$F_2 \equiv (P \wedge \gamma^+) \vee (F_1 \wedge (\neg\gamma^- \vee \neg P))$$

■

The successor state axiom (4.5) gives Reiter's successor state axiom (3.11) as a special case when we have $Poss(a, s) \equiv \top$ (i.e. when all actions are possible in every situation).

4.3 Consequences of New Form of Successor State Axioms

The new form of successor state axioms has several consequences; some conceptual changes and some changes in the formalizations. In the sequel we briefly review these changes.

1. Definition of $do(a, s)$. This is among the conceptual changes. In Reiter's Situation Calculus, $do(a, s)$ is defined as "the successor situation to s, resulting from performing (i.e. executing) action a". In other words, it is assumed that actions always succeed. If action a is not possible in situation s, then $do(a, s)$ and subsequent situation are what Reiter calls "ghost" situations. In these cases the actions still succeed but the situations are not physically realizable.

In the new framework, since possibility of actions are taken into account in successor state axioms, $do(a, s)$ now is defined as the successor situation to s, resulting from *attempting* action a. An *attempt* to do an action is different from *performing* (or executing) an action in the sense that it is not assumed that it succeeds. If it is possible to perform an action in a situation then the action has its expected effects, otherwise nothing happens (i.e. a null action is executed). This is reasonable because for example if it is not possible to execute action $pickup(x)$ (say, because x is glued to the floor), then after (attempting) the action $pickup(x)$, it is reasonable to assume that nothing happens.

We make this an assumption of our framework: If an action is not possible in a situation, "executing" it (in fact attempting to execute it) does not make any changes in the model. Formally

$$\neg Poss(a, s) \supset F(\vec{x}, do(a, s)) \equiv F(\vec{x}, s)$$

In fact this was the assumption that led to extra frame axioms in section 4.1.[2]

2. Projection Problem. In the new framework we can solve the more general form of the problem where the sequence of actions does not have to be executable. The solution for the projection problem is still the same and is equivalent to whether

$$Axioms \models G(do([a_1, \ldots, a_n], S_0),$$

but now we apply regression using the new successor state axiom (4.5).

Consider again example 2 where an object Obj is glued to the floor and we want to formalize the result of the agent picking up the object followed by moving to the next room.

[2]For more details on the alternative definition of $do(a, s)$ see [30].

The new successor state axioms are

$$holding(x, do(a, s)) \equiv [a = pickup(x) \wedge Poss(a, s)] \vee holding(x, s)$$
$$loc(x, y, do(a, s)) \equiv [a = moveTo(y) \wedge Poss(a, s) \wedge (holding(x, s) \vee x = Robot)]$$
$$\vee [loc(x, y, s) \wedge ((\neg\exists z \neq y).a = moveTo(z) \vee \neg holding(x, s) \vee \neg Poss(a, s))$$

The new formalization entails
$\neg holding(Obj, do([pickup(Obj), moveTo(R2)], S_0)$
$loc(Obj, R1, do([pickup(Obj), moveTo(R2)], S_0)$
$loc(Robot, R2, do([pickup(Obj), moveTo(R2)], S_0)$

So after the sequence of pickup-and-move actions, the agent is in room 2, not holding any object, and the object is still in room 1, which is the desired result.

3. Formalization of Knowledge. The ability of our framework to formalize the result of unexecutable actions (by means of the new form of successor state axioms) enables situations to correctly represent possible worlds. This solves one of the problems with formalization of knowledge in the Situation Calculus. To make it fully work, we need to develop a successor state axiom for fluent K as well.

First note that, if an agent in situation s considers n situations as possible worlds, no matter what action a she attempts, in situation $do(a, s)$ the number of situations she considers as possible worlds is less than or equal to n. In other words, as a result of actions, the number of indistinguishability (aka accessibility) relations either decreases or stays the same; it never increase. This is because physical actions do not change the agent's knowledge (as enforced in the definition of sensing result function) and sensing actions can only produce knowledge (by eliminating indistinguishability relations). As a result, $\gamma^+_K(\vec{x}, a, s)$ is always false.

If we assume that sensing actions are always possible (which is the case in the literature), then the following formula qualifies as the successor state axiom for fluent K

$$K(s'', do(a, s)) \equiv (\exists s').s'' = do(a, s') \wedge K(s', s) \wedge \text{SR}(a, s) = \text{SR}(a, s') \tag{4.16}$$

Which can be written in a simpler form as

$$K(do(a, s'), do(a, s)) \equiv K(s', s) \wedge \text{SR}(a, s) = \text{SR}(a, s')$$

Note that $\text{SR}(a, s) \neq \text{SR}(a, s')$ is $\gamma^-_K(\vec{x}, a, s)$ (i.e. it is what eliminates indistinguishability relations).

Consider again example 1 where an object was glued to the floor but the agent was uncertain about it, and we wanted to see what the agent's beliefs are after (attempting) to pick up the object . Using the above successor state axiom for K fluent and the below

successor state axiom for the fluent $holding(x)$, we see that after the *pickup* action, the agent is uncertain between two possible worlds, one where the object is on the floor and one where the agent is holding the object, which is the desired result.

$$holding(x, do(a, s)) \equiv (a = pickup(x) \wedge Poss(a, s)) \vee holding(x, s)$$

Let's now relax the assumption that sensing actions are always possible. Regarding what should happen in attempting an unexecutable sensing action, we choose the same policy as we chose for physical actions, namely nothing happens (i.e. a null action is executed). In other words, attempting a sensing action that is not possible (for example detecting the color of an object when there is no light) does not change the indistinguishability relations in the resulting situation and therefore does not add to the agent's knowledge.

In this case we need to account for the possibility of sensing actions in the successor state axiom of the K fluent. Following the pattern of (4.5) we arrive at the following formula

$$\begin{aligned} K(s'', do(a, s)) \equiv (\exists s').s'' = do(a, s') \wedge K(s', s) \\ \wedge [\text{SR}(a, s) = \text{SR}(a, s') \vee \neg Poss(a, s)] \end{aligned} \tag{4.17}$$

Which can be written in a simpler form as

$$K(do(a, s'), do(a, s)) \equiv K(s', s) \wedge [\text{SR}(a, s) = \text{SR}(a, s') \vee \neg Poss(a, s)]$$

As an example of the consequences of our more general form for successor state axioms mentioned above, here we prove a generalization of the theorem 2.4.1 where it is not limited to executable situations:

Theorem 4.3.1. *If the K relation on the set of initial situations is restricted to conform to the reflexive condition along with some subset of the symmetric, transitive and Euclidean properties, then the K relation at every situation, will satisfy the same set of properties.*

Proof. The proof is similar to that of [25], with the difference that now we use the new form for successor state axiom of K (for simplicity we use 4.16 instead of 4.17).

Using induction, it suffices to prove that if a property holds at s then it holds at $do(a, s)$ for arbitrary a and s. The proof for reflexivity is straightforward because $\text{SR}(a, s) = \text{SR}(a, s)$.

Regarding other properties, we prove the case for symmetry property, the other ones are proved similarly.

So suppose we have $(\forall s').K(s', s) \supset K(s, s')$. We need to prove that

$$(\forall s'').K(s'', do(a, s)) \supset K(do(a, s), s'')$$

For the sake of obtaining a contradiction suppose that it doesn't hold. So we have

$$(\exists s'').K(s'', do(a, s)) \wedge \neg K(do(a, s), s'')$$

The proof continues as follows

(1)$(\forall s').K(s', s) \supset K(s, s')$	Premise
(2) $K(s'', do(a, s))$	Premise
(3) $\neg K(do(a, s), s'')$	Premise
(4) $(\exists s^*).s'' = do(a, s^*) \wedge K(s^*, s)$	
$\wedge \text{SR}(a, s^*) = \text{SR}(a, s)$	(2), successor state axiom for K
(5)$K(s, s^*) \wedge \text{SR}(a, s^*) = \text{SR}(a, s)$	(1), (4)
(6)$K(do(a, s), do(a, s^*))$	(5), successor state axiom for K
(7)$K(do(a, s), s'')$	(6), (4)
(8)Contradiction	(3), (7)

4.4 Consistency Check

In this section we prove that the changes that we have made so far does not produce any problem, in the sense that whatever was holding in the Reiter's version of the Situation Calculus, holds in our generalized framework.

Recall from the background section the definition of a basic action theory was a collection of axioms $\mathcal{D} = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ where $\Sigma, \mathcal{D}_{ap}, \mathcal{D}_{ss}, \mathcal{D}_{una}$ are the set of foundational, action precondition, successor state and unique name axioms respectively, and $\mathcal{D}_{S_0}$ is the set of formulas describing the initial situation.

Note that the only part of the above definition we have been dealing with in the previous chapter is $\mathcal{D}_{ss}$ where we suggested using (4.5) as a generalized form for successor state axioms (compared to 3.11). All the other parts are untouched. So if our more general form for successor state axioms conforms to the required conditions of successor state axioms, we are done.

In fact there is a problem here. Recall from the background section the definition of a successor state axiom for fluent F was

$$F(\vec{x}, do(a, s)) \equiv \phi_F(\vec{x}, a, s)$$

where ϕ is *uniform* in s. And among the necessary conditions for uniformity of a formula was not having the predicate $Poss$ as part of the formula. But since $Poss$ appears in the right hand side of (4.5), our general form for successor state axioms is not uniform in s, and technically problematic. But this issue can be resolved.

As we will see, not allowing $Poss$ to appear in the right hand side is only necessary for action precondition axioms and not for the successor state axioms. We don't want $Poss$ to appear in the right hand side of action precondition axioms because (among other possible problems) it can result in circular definitions. For example we could have

$$Poss(A(x), s) \equiv P(s) \wedge Poss(B(x), s)$$
$$Poss(B(x), s) \equiv P(s) \wedge Poss(A(x), s)$$

which should not be allowed.

But having $Poss$ on the right hand side of successor state axioms does not produce any problem, as we prove in the sequel.

Lemma 4.4.1. *Let ϕ be a successor state axiom in the form of (4.5) and $\mathcal{A}$ be the set of action precondition axioms for the actions mentioned in ϕ. Let ϕ' be the formula obtained from ϕ where occurrences of Poss in ϕ is replaced with the right hand side of their corresponding action precondition axioms from $\mathcal{A}$. Then*

$$\mathcal{A}, \phi \vDash \phi',$$
$$\mathcal{A}, \phi' \vDash \phi$$

Proof: Let ϕ be the following formula (without loss of generality we assumed $\gamma_F^+(\vec{x}, a, s)$ and $\gamma_F^-(\vec{x}, a, s)$ contain only one action)

$$F(\vec{x}, do(a, s)) \equiv (a = A(\vec{x}) \wedge Poss(A(\vec{x}), s)) \vee [F(\vec{x}, s) \wedge (a \neq B(\vec{x}) \vee \neg Poss(B(\vec{x}), s))]$$

and $\mathcal{A}$ be the following two formulas

$$Poss(A(\vec{x}), s) \equiv \psi_A(\vec{x}, s)$$
$$Poss(B(\vec{x}), s) \equiv \psi_B(\vec{x}, s)$$

So ϕ' is (replacing $Poss(A(\vec{x}), s)$ with $\psi_A(\vec{x}, s)$ and $Poss(B(\vec{x}), s)$ with $\psi_B(\vec{x}, s)$ in ϕ):

$$F(\vec{x}, do(a, s)) \equiv (a = A(\vec{x}) \wedge \psi_A(\vec{x}, s) \vee [F(\vec{x}, s) \wedge (a \neq B(\vec{x}) \vee \neg \psi_B(\vec{x}, s))]$$

Now the proofs of $\mathcal{A}, \phi \vDash \phi'$ and $\mathcal{A}, \phi' \vDash \phi$ are similar to proof of

$$(\forall x).P(x) \equiv Q(x) \wedge R(x), (\forall x).R(x) \equiv R'(x) \vDash \forall(x).P(x) \equiv Q(x) \wedge R'(x)$$

We prove this using Resolution. The proofs of $\mathcal{A}, \phi \vDash \phi'$ and $\mathcal{A}, \phi' \vDash \phi$ can be done in a similar manner.

We convert the left-hand-side formulas and negation of the right-hand-side formula to clausal form and derive an empty clause using Resolution which completes the proof.

Negation of the right-hand-side formula: $(\exists x).(P(x) \wedge (\neg Q(x) \vee \neg R'(x))) \vee (Q(x) \wedge R'(x) \wedge \neg P(x))$
Negation of the right-hand-side formula in clausal form (after simplification and Skolemization): $[P(a), Q(a)], [P(a), R'(a)], [\overline{Q(a)}, \overline{R'(a)}, \overline{P(a)}]$
Left-hand-side formulas in clausal form: $[\overline{P(x)}, Q(x)], [\overline{P(x)}, R(x)], [\overline{Q(x)}, \overline{R(x)}, P(x)], [\overline{R(y)}, R'(y)], [\overline{R'(y)}, R(y)]$
Resolving $[P(a), Q(a)]$ and $[\overline{P(x)}, Q(x)]$ yields $[Q(a)]$
Resolving $[P(a), R'(a)]$ and $[\overline{P(x)}, R(x)]$ yields $[R(a), R'(a)]$
Resolving $[\overline{R'(y)}, R(y)]$ and $[R(a), R'(a)]$ yields $[R(a)]$
Resolving $[\overline{R(y)}, R'(y)]$ and $[R(a)]$ yields $[R'(a)]$
Resolving $[Q(a)]$ and $[\overline{Q(x)}, \overline{R(x)}, P(x)]$ yields $[\overline{R(a)}, P(a)]$
Resolving $[\overline{R(a)}, P(a)]$ and $[R(a)]$ yields $[P(a)]$
Resolving $[Q(a)]$ and $[\overline{Q(a)}, \overline{R'(a)}, \overline{P(a)}]$ yields $[\overline{R'(a)}, \overline{P(a)}]$
Resolving $[\overline{R'(a)}, \overline{P(a)}]$ and $[R'(a)]$ yields $[\overline{P(a)}]$
Resolving $[\overline{P(a)}]$ and $[P(a)]$ yields $[\ ]$ ■

Theorem 4.4.1. *Let $\mathcal{D}^* = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}^*_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ where $\Sigma, \mathcal{D}_{ap}, \mathcal{D}_{una}, \mathcal{D}_{S_0}$ are as defined before and $\mathcal{D}^*_{ss}$ is a set of successor state axioms in the form of (4.5). Let $\mathcal{D}_{ss}$ be the set of successor state axioms in $\mathcal{D}^*_{ss}$ where all occurrences of Poss predicate is replaced with the right hand side of their corresponding action precondition axioms from $\mathcal{D}_{ap}$. Then $\mathcal{D} = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is a basic action theory and for an arbitrary formula ψ the following holds*

$$\mathcal{D}^* \models \psi \quad \textit{iff} \quad \mathcal{D} \models \psi$$

Proof: Since the *Poss* predicates that appeared in formulas contained in $\mathcal{D}^*_{ss}$ do not appear in $\mathcal{D}_{ss}$ formulas (because *Poss* occurrences were replaced), the successor state axioms in $\mathcal{D}_{ss}$ conform to uniformity requirement and therefore $\mathcal{D}$ is a basic action theory.
For the second part, note that from the above lemma it follows that $\mathcal{D}^*_{ss} \models \mathcal{D}_{ss}$ and $\mathcal{D}_{ss} \models \mathcal{D}^*_{ss}$. Therefore every formula that is entailed by $\mathcal{D}^*$ is also entailed by $\mathcal{D}$ and vice versa. ■

4.5 A Note on Different Versions of Reiter's Solution to the Frame Problem

In general there are two accounts of Reiter's solution to the frame problem. The first one was in his 1991 paper [23], and the second one in his book [24]. We reviewed the latter in section 3.1.1. Those familiar with the former notice that our more general solution to the frame problem starts with the same form for effect axioms as [23]. So it could be helpful

in better understanding of the approaches if we compare our solution with the solution provided in [23]. We use terminology of [24] for consistency of representation.

Reiter starts with the same form for effect axioms as (4.1) and (4.2):

$$Poss(a, s) \wedge \gamma_F^+(\vec{x}, a, s) \supset F(\vec{x}, do(a, s)) \tag{4.18}$$

$$Poss(a, s) \wedge \gamma_F^-(\vec{x}, a, s) \supset \neg F(\vec{x}, do(a, s)) \tag{4.19}$$

The completeness assumption then states effect axioms (4.18) and (4.19) capture all conditions under which fluent F changes value as a result of action a.

Reiter continues as follows ([23], page 10):

> "Hence, if action a is possible and F's truth value changes from $false$ to $true$ as a result of doing a, then $\gamma_F^+(\vec{x}, a, s)$ must be $true$; similarly, if F's truth value changes from $true$ to $false$."

Formally

$$Poss(a, s) \wedge F(\vec{x}, s) \wedge \neg F(\vec{x}, do(a, s)) \supset \gamma_F^+(\vec{x}, a, s) \tag{4.20}$$

$$Poss(a, s) \wedge \neg F(\vec{x}, s) \wedge F(\vec{x}, do(a, s)) \supset \gamma_F^-(\vec{x}, a, s) \tag{4.21}$$

While this is a valid derivation from the completeness assumption, it is not the strongest statement we can derive. The stronger derivation would be

$$F(\vec{x}, s) \wedge \neg F(\vec{x}, do(a, s)) \supset \gamma_F^-(\vec{x}, a, s) \wedge Poss(a, s) \tag{4.22}$$

$$\neg F(\vec{x}, s) \wedge F(\vec{x}, do(a, s)) \supset \gamma_F^+(\vec{x}, a, s) \wedge Poss(a, s) \tag{4.23}$$

These are in fact (4.3) and (4.4), the ones we used to derive the more general form for successor state axioms.

To sum up, in the solution provided in [24], since the possibility of actions is not accounted for, the resulting successor state axiom becomes insensitive to action executability. On the other hand, in the solution provided in [23], although possibility of actions is accounted for, since the completeness assumption is not fully used, the resulting successor state axiom is silent on actions that are not executable.

Chapter 5

Conclusion

We reviewed two of the most important theories of action and discussed their epistemic intricacies. We provided a more nuanced and expressive version of the Situation Calculus by presenting a more general form for successor state axioms which was developed based on a more general solution to the frame problem in the Situation Calculus.

We described some advantages of the new framework. In the traditional framework we can solve the projection problem but only for executable situations. We can regard situations as possible worlds but only when the situation is executable. We showed that the current formalization of knowledge has some flaws (although it works fine if we assume all actions are executable in every situation starting from any of the initial situations which is a very strong assumption.). In our framework we don't have these limitations and it allows us to utilize the power of the Situation Calculus in a broader area.

Studying other impacts and applications of the new framework is subject of the future research.

Bibliography

[1] F. Bacchus, J.Y. Halpern, and H.J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence*, 111(1-2):171–208, 1999.

[2] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. Reasoning about noisy sensors in the situation calculus. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1933–1940. Morgan Kaufmann, 1995.

[3] Patrick Blackburn and Johan Van Benthem. Modal logic: A semantic perspective. In Johan Van Benthem Patrick Blackburn and Frank Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, 2007.

[4] B.F. Chellas. *Modal Logic*. Cambridge University Press, 1980.

[5] Ernest Davis. *Representations of commonsense knowledge*. Morgan Kaufmann, 1990.

[6] James P. Delgrande and Hector J. Levesque. A formal account of nondeterministic and failed actions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Beijing, China, 2013.

[7] Jeremy Forth and Murray Shanahan. Indirect and conditional sensing in the event calculus. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 900–904. IOS Press, 2004.

[8] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *The Journal of Logic Programming*, 17(2-4):301–321, 1993.

[9] Steffen Hölldobler and Josef Schneeberger. A new deductive approach to planning. *New Generation Computing*, 8(3):225–244, 1990.

[10] Ryan F. Kelly and Adrian R. Pearce. Knowledge and observations in the situation calculus. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 124:1–124:3, New York, NY, USA, 2007. ACM.

[11] Ryan F. Kelly and Adrian R. Pearce. Complex epistemic modalities in the situation calculus. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning*, KR'08, pages 611–620. AAAI Press, 2008.

[12] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[13] H.J. Levesque, F. Pirri, and R. Reiter. Foundations for the situation calculus. *Linköping Electronic Articles in Computer and Information Science*, 3(18), 1998.

[14] François Lévy and Joachim Quantz. Representing beliefs in a situated event calculus. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, 1997.

[15] J. McCarthy. Situations, actions and causal laws. *Stanford Artifcial Intelligence Project: Memo 2*, 1963.

[16] J. McCarthy and P.J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie and B. Meltzer, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[17] Rob Miller and Murray Shanahan. *The event calculus in classical logic-Alternative axiomatisations*. Linköping University Electronic Press, 1999.

[18] Rob Miller and Murray Shanahan. Some alternative formulations of the event calculus. In *Computational logic: Logic programming and beyond*, pages 95–111. Springer, 2002.

[19] R.C. Moore. Reasoning about knowledge and action. Technical Report 191, Artificial Intelligence Center, SRI International, 1980.

[20] R.C. Moore. A formal theory of knowledge and action. In *Formal Theories of the Commonsense World*. Ablex Publishing Corp., 1985.

[21] Leora Morgenstern. Knowledge preconditions for actions and plans. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 867–874, 1987.

[22] Theodore Patkos and Dimitris Plexousakis. A theory of action, knowledge and time in the event calculus. In *Proceedings of fifth Hellenic Conference on Artificial Intelligence*, volume 8, pages 226–238. Springer, 2008.

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

[24] R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Cambridge, MA, 2001.

[25] R .B. Scherl and H.J. Levesque. Knowledge, action, and the frame problem. *Artificial Intelligence*, 144(1-2):1–39, 2003.

[26] Richard B Scherl. Reasoning about the interaction of knowledge, time and concurrent actions in the situation calculus. In *Proceedings of 18th International Joint Conference on Artificial intelligence*, volume 18, pages 1091–1098, 2003.

[27] Richard B. Scherl and Hector J. Levesque. The frame problem and knowledge-producing actions. In Richard Fikes and Wendy Lehnert, editors, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 698–695, Menlo Park, California, 1993. American Association for Artificial Intelligence, AAAI Press.

[28] Lenhart K. Schubert. Monotonic solution of the frame problem in the situation calculus: An efficient method for worlds with fully specified actions. In *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer Academic Press, 1990.

[29] S. Shapiro, Y. Lespérance, and H. Levesque. The cognitive agents specification language and verification environment for multiagent systems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent System*, pages 19–26, 2002.

[30] Vahid Vaezian and James P Delgrande. On the role of possibility in action execution and knowledge in the situation calculus. In *Canadian Conference on Artificial Intelligence*, pages 155–161. Springer, 2017.