Non-uniform Knowledge In The Situation Calculus

by

James Twigg

B.Sc., University of Victoria, 2013

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

in the
Department of Computing Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY
Fall 2016

Approval

Name:	**James Twigg**
Degree:	**Master of Science (Computing Science)**
Title:	***Non-uniform Knowledge In The Situation Calculus***
Examining Committee:	**Chair:** Dr. Steven Pearce Lecturer

Dr. James Delgrande
Senior Supervisor
Professor _____

Dr. Eugenia Ternovska
Supervisor
Associate Professor _____

Dr. Aaron Hunter
External Examiner
Research Faculty _____

Date Defended: September 26, 2016

Abstract

Knowledge Representation and Reasoning is the field of AI concerned with storing information in a way which can be actioned upon by an agent. The situation calculus is a popular logical language for reasoning about action. A prior work by Scherl and Levesque demonstrates how the situation calculus can be used to model knowledge and knowledge-producing actions while solving the frame problem. This approach is limited in that it can only represent knowledge pertaining to the same situation in which it is held. Shapiro et al. have demonstrated how retrospection can be represented, but not so for prospection. We present an extension of Scherl and Levesque's approach which allows for prospection and reasoning hypothetically about the outcomes of actions before they are taken.

Keywords: Knowledge Representation, Non-uniform Knowledge, Reasoning About Action, Situation Calculus, Temporal Reasoning, Hypothetical Reasoning, Introspection, Retrospection, Prospection

Acknowledgements

We wish to thank Jim Delgrande for his indispensable counsel as supervisor in the development of this work. We also thank fellow Computational Logic Laboratory students Vahid Vaezian and Paul Vicol for their assistance.

Table of Contents

Approval ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables vii

List of Figures viii

1 Introduction 1

2 Background: The Situation Calculus 3

2.1 Notation . . . 3

2.2 Logical Elements . . . 3

2.3 Axioms and Definitions . . . 6

2.4 Knowledge: The Epistemic Extension to the Situation Calculus . . . 8

2.4.1 The Epistemic Fluent . . . 8

2.4.2 Axioms Related to The Epistemic Fluent . . . 10

3 Proposed Framework 12

3.1 Description and Motivation . . . 12

3.2 Relative Formulae . . . 14

3.3 Knowledge . . . 19

3.3.1 Definition of the Knowledge Operator and Knowable Formulae . . . 19

4 An Extended Example 28

4.1 Axioms . . . 29

4.1.1 Action Precondition Axioms . . . 29

4.1.2 Successor State Axioms . . . 30

4.2 Initial State . . . 31

4.2.1 Taking Actions . . . 32

5 Properties and Theorems **40**
5.1 Introspection 40
5.2 Default Persistence of Ignorance 42
5.3 Knowledge Incorporation 43
5.4 Memory 44
5.5 Knowledge of Effects of Actions 45
5.5.1 Sensing Actions 46

6 Conclusions **48**
6.1 Related Work 48
6.2 Future Work 49

Bibliography **51**

List of Tables

Table 4.1 The Initial State 32
Table 4.2 Closed Door 33
Table 4.3 Light Switch Flipped 34
Table 4.4 Light Sensed 35
Table 4.5 Room Sensed 36
Table 4.6 Door Open 37
Table 4.7 Changed Room 38
Table 4.8 Complete Knowledge 39

List of Figures

Figure 2.1 Knowledge of P, ignorance of Q 9
Figure 2.2 A sequence of three actions 9

Chapter 1

Introduction

The situation calculus[8] is a foundational work on action and change in the field of artificial intelligence owing originally to John McCarthy. In this thesis however we will be employing the situation calculus as described by Raymond Reiter[5]. Our approach to knowledge representation in the situation calculus closely follows that of Moore[9, 10] by using a relation to describe which situations the agent considers possible. When a fact is true in all possible cases then the agent knows that fact. The frame problem is that of efficiently updating one's representation of the state of the world as the result of an event — both what changed and what did not. An extention of Moore's approach to provide a compact solution to the frame problem was published by Reiter[13]. However this solution did not handle "knowledge-producing" actions — ones which have no effect other than to change the agent's knowledge. A generalization of Reiter's solution to include knowledge-producing actions is provided by Scherl and Levesque[14].

This culminates in a fairly complete theory of knowledge and action in the situation calculus, though it is still limited in that only knowledge about the current situation can be represented. Shapiro et al.[15] build on the previous works to describe belief as opposed to knowledge[1] and provide a representation of retrospection: belief about what was previously true. However their attempts to also model prospection (belief about what *will be* true) were defeated by knowledge-producing actions which revealed to the agent information in contradiction with its current beliefs. It was this discussion in [15] which inspired this thesis.

Initially an attempt was made to extend the approach of Shapiro et al. to also model prospection but this proved to require at least one undesirable axiom and was beyond the scope of a Master's thesis. Instead we reverted to extending Scherl and Levesque's work, considering knowledge rather than belief and eliminating the possibility of a sensing result contradicting the agent's knowledge. The result is a method for representing knowledge in

[1] The difference being that beliefs may be incorrect while knowledge is always accurate.

the situation calculus which allows introspection ("I know that I know x"), retrospection ("It was previously the case that x"), and prospection ("After action x it will be the case that y").

In particular we show that we are able to express the agent possessing knowledge of the behaviour of a sensing action without that knowledge revealing the result of the action before it is performed.

Chapter 2

Background: The Situation Calculus

2.1 Notation

In this paper we will adhere to the following notational standards.

- Functions of arity greater than 0 will be denoted with lower-case names (e.g. do, sr).
- Variables representing objects will be single lower-case Roman letters, possibly with some combination of primes, subscripts, and superscripts (s_1, a'). We will use s for situations and a for actions.
- Non-specific terms or logical formulae will be denoted with Greek letters. We will use φ, ψ for formulae and σ, τ, ρ for terms.
- Constants (i.e. functions of arity 0) and predicates will have capitalized names ($Knows, S_0, K$).

2.2 Logical Elements

The situation calculus (McCarthy[8]) is a theory of formal logic with sorts consisting of *situations, actions*, and *objects*. This description of the calculus is taken from Levesque et al. in [5] (ammended where necessary). Situations represent possible states of the environment (including their histories), actions represent how the agent can affect the environment (their actual effects are encoded in related axioms), and objects are anything else. Aside from

standard symbols such as $\wedge$, $\neg$, and $\exists$ which are used in their typical meanings, the alphabet is as follows.

- Countably many terms for actions, situations, and objects. There are also countably many predicates of any arity.
- Two functions which return situations:
  1. A constant symbol S_0, denoting the initial situation.
  2. A binary function symbol $do : action \times situation \rightarrow situation$. The intended interpretation is that $do(a, s)$ denotes the successor situation resulting from performing action a in situation s.
- A binary predicate symbol $\sqsubseteq$: $situation \times situation$, defining an ordering relation on situations. The intended interpretation of situations is as action histories, in which case $s \sqsubseteq s'$ means that s is a subhistory of s'.
- A binary predicate symbol $Poss : action \times situation$. The intended interpretation of $Poss(a, s)$ is that it is possible to perform the action a in situation s.
- A binary function symbol $sr : action \times situation \rightarrow object$ which determines the result of using a sensor in a particular situation.[1]
- For each $n \geqslant 0$, countably many predicate symbols with arity n, and arguments of sort $(action \cup object)^n$. These are used to denote situation independent relations like
  - $Human(John)$
  - $PrimeNumber(n)$
  - $MovingAction(run(person, loc1, loc2))$

  etc.
- For each $n \geqslant 0$, countably many function symbols of form $(action \cup object)^n \rightarrow object$. These are used to denote situation independent functions like
  - $sqrt(x)$
  - $height(MtEverest)$
  - $agent(run(person, loc1, loc2))$

  etc.

[1] It will frequently be the case that this function always evaluates to either TRUE or FALSE, in which case it can be replaced by a binary predicate to simplify axioms. We present the more general case here.

- For each $n \geqslant 0$, enumarably many function symbols of form $object^n \rightarrow action$. These are called *action functions*, and are used to denote actions like

  - $pickup(x)$
  - $move(a, b)$

  etc. In most applications, there will be just finitely many action functions, but we allow the possibility of an infinite number of them.

  Notice that we distinguish between function symbols taking values of sort *object* and those — the action functions — taking values of sort *action*. In what follows, the latter will be distinguished by the requirement that they be axiomatized in a particular way by what we shall call *action precondition axioms*.

- For each $n \geqslant 0$, countably many predicate symbols with arity n, and arguments of sort $(action \cup object)^n \times situation$. These predicate symbols are called *relational fluents*. In most applications, there will be just finitely many relational fluents, but we do not preclude the possibility of an infinite number of them. These are used to denote situation dependent relations like

  - $OnTable(x, s)$
  - $Husband(Mary, John, s)$

  etc. Notice that relational fluents take just one argument of sort situation, and this is always its last argument.

- For each $n \geqslant 0$, a countable number of function symbols of sort $(action \cup object)^n \times situation \rightarrow action \cup object$. These function symbols are called *functional fluents*. In most applications, there will be just finitely many functional fluents, but we do not preclude the possibility of an infinite number of them. These are used to denote situation dependent functions like

  - $age(Mary, s)$
  - $primeMinister(Italy, s)$

  etc. Notice that functional fluents take just one argument of sort situation, and this is always its last argument.

Observe that except for the reserved predicate symbol $\sqsubseteq$ we restrict fluents to exactly one situation argument.

2.3 Axioms and Definitions

We wish to be able to discuss to which situation(s) a formula pertains. The simplest case is that of a uniform formula; one which pertains only to a single situation, or is independent of any situation. Formally we define a uniform formula as follows.

Definition 1 (Uniform Formulae)**.** *Let σ be a situation term. There are three categories of terms which are uniform in σ:*

1. *Terms which do not mention situation terms.*
2. *σ.*
3. *If g is an n-ary function symbol taking values of sort other than situation, and $\tau_1, \ldots, \tau_n$ are terms uniform in σ whose sorts are appropriate for g, then $g(\tau_1, \ldots, \tau_n)$.*

Similarly there are three categories of formulae uniform in σ:

1. *$\tau_1 = \tau_2$, if τ_1 and τ_2 are terms uniform in σ.*
2. *Predicates other than $Poss$ and $\sqsubseteq$ whose arguments are terms uniform in σ.*
3. *Formulae $\neg\varphi, \varphi \wedge \psi$, and $(\exists v)\varphi$ where φ, ψ are formulae uniform in σ and v is an individual variable of sort other than situation.*

Some examples of uniform terms/formulae:

- $Dog(Fido)$
- $age(Mary, s)$
- $(husband(Mary, s) = John \wedge wife(John, s) = Mary) \rightarrow Married(John, Mary, s)$

Some examples of *non*-uniform terms/formulae:

- $distanceMoved(BlockA, s_1, s_2)$
- $age(Mary, s_1) \neq age(Mary, s_2)$
- $Poss(wed(John, Mary), s) \rightarrow Married(John, Mary, do(wed(John, Mary), s))$

This thesis builds on the concept of uniformity in later sections.

The following axiom is a unique names axiom for situations: it enforces the uniqueness of situations as action histories. With this there are no two distinct sequences of actions which can result in the same situation.

Axiom 1 (Unique Names for Situations)**.**
$do(a, s) = do(a', s') \rightarrow a = a' \wedge s = s'$

To enforce the distinctness of actions we give unique names axioms for actions as well:
Axiom 2 (Unique Names for Actions)**.** *For distinct action names a and b:*
$a(\overrightarrow{x}) \neq b(\overrightarrow{y})$
$a(\overrightarrow{x}) = a(\overrightarrow{y}) \rightarrow \overrightarrow{x} = \overrightarrow{y}$

The next axiom inductively enforces the intuitive meaning of $\sqsubseteq$ as an ordering on situations as determined by *do*. The situation s preceeds another situation s' exactly if there is some sequence of actions $a_1, \ldots, a_n$ such that $do(a_n \ldots do(a_2, do(a_1, s)) \ldots) = s'$.
Axiom 3.
$s \sqsubseteq s' \equiv s = s' \vee \exists a.do(a, s) \sqsubseteq s'$

We will also use $\sqsubset$ as a notational shorthand for denoting proper subhistories.
Definition 2. *Where s and s′ are situations*
$s \sqsubset s' \stackrel{\text{def}}{=} s \sqsubseteq s' \wedge \neg(s' \sqsubseteq s)$

This axiom restricts the $Init$ predicate to its intuitive meaning: those situations which have no predecessor.
Axiom 4. *For a situation s:*
$Init(s) \equiv \neg\exists s'.s' \sqsubset s$

By convention the initial state of the world is named "S_0". It would be counterintuitive to model situations prior to the initial situation, so we enforce with this axiom that the $Init$ predicate holds for S_0.
Axiom 5.
$Init(S_0)$

However we diverge from Reiter by not preculding situations other than S_0 from also being initial. In the standard situation calculus it is the intention that *do* (and $\sqsubseteq$) impose a directed tree over the situations in the theory with S_0 as the root; that is, all situations are reachable from S_0 via some sequence of actions. We instead have that *do* and $\sqsubseteq$ impose a forest where each initial situation is the root of a distinct tree. To this end we must give an induction axiom. Reiter's axiom[12] was as follows:

$$\forall P.\{P(S_0) \wedge \forall a, s.[P(s) \rightarrow P(do(a, s))]\} \rightarrow \forall s.P(s)$$

Unfortunately this axiom is unsuitable for our purposes as it assumes S_0 to be the only initial situation. Instead we have the following which states that Reiter's induction axiom holds for each initial situation.
Axiom 6 (Induction)**.** *Where P is a predicate:*
$\forall s.Init(s) \rightarrow \{\forall P.P(s) \wedge \forall a, s'.[(s \sqsubseteq s' \wedge P(s')) \rightarrow P(do(a, s'))] \rightarrow \forall s'.s \sqsubseteq s' \rightarrow P(s')\}$

For each action we require an axiom of the form:

Definition 3 (Action Precondition Axiom)**.** *For an action function a and variables* $x_1, \ldots, x_n$ *of sort object*

$$Poss(a(x_1, \ldots, x_n), s) \stackrel{\text{def}}{=} \Pi_a(x_1, \ldots, x_n, s),$$

where $\Pi_a(x_1, \ldots, x_n, s)$ *is a formula whose free variables are among* $x_1, \ldots, x_n$ *and* s*.*

Additionally we require axioms to describe which fluents are satisfied after performing an action. We call these Successor State Axioms and they have the following forms:

Definition 4. *For a relational fluent F and non-situation variables* $x_1, \ldots, x_n$

$$F(x_1, \ldots, x_n, do(a, s)) \stackrel{\text{def}}{=} \Phi_F(x_1, \ldots, x_n, a, s),$$

where Φ_F *is a formulae whose free variables are among* $x_1, \ldots, x_n, a,$ *and* s*.*

Definition 5. *For a functional fluent f and non-situation variables* $x_1, \ldots, x_n$

$$f(x_1, \ldots, x_n, y, do(a, s)) \stackrel{\text{def}}{=} \phi_f(x_1, \ldots, x_n, y, a, s),$$

where ϕ_f *is a formulae whose free variables are among* $x_1, \ldots, x_n, y, a,$ *and* s*.*

Finally axioms which define the values of the sensing result function sr must be given. The function $sr : action \times situation \rightarrow object$ describes the result of using a sensor in a situation. For each action there is a sensor result axiom of the form:

Definition 6. *Where a is an action,* $x_1, \ldots, x_n$ *non-situation variables, and s a situation:*

$$sr(a(x_1, \ldots, x_n), s) = r \equiv \phi_a(x_1, \ldots, x_n, a, r, s),$$

where ϕ_a *is a formula whose free variables are among* $x_1, \ldots, x_n, a, s$*.*

The theory which results from the union of these axioms is the Basic Action Theory and we denote it $\mathcal{D}$.

2.4 Knowledge: The Epistemic Extension to the Situation Calculus

Scherl and Levesque provide an extension to the situation calculus which is capable of representing knowledge. This accomplished with an epistemic relational fluent $K : situation \times situation$.

2.4.1 The Epistemic Fluent

As in previous works[14, 9, 10], the accessibility relation $K : situation \times situation$ is used to model the agent's knowledge about the state of the environment. The relation represents

uncertainty between distinct possible cases for the state of the world. Thus (simply stated) we say that a formula is known in a situation if it is satisfied in every K-related situation. The definition is as follows, where F is a fluent:

$$Knows(F, s) \stackrel{\text{def}}{=} \forall s'.K(s', s) \rightarrow F(s')$$

K is the only symbol other than $\sqsubseteq$ for which we make an exception to the restriction on the number of situation arguments in fluents.

Consider the following figure where P and Q are fluents, the circles represent situations, and the arrows the valuation of K. In this case S_0 is the actual situation, but S_1 is accessible from S_0 so the agent believes either could be the actual situation. The agent knows P to be true as it is satisfied in every accessible situation, but the same cannot be said for Q or $\neg Q$.

Figure 2.1: Knowledge of P, ignorance of Q

Figure 2.2 shows a more in-depth example.

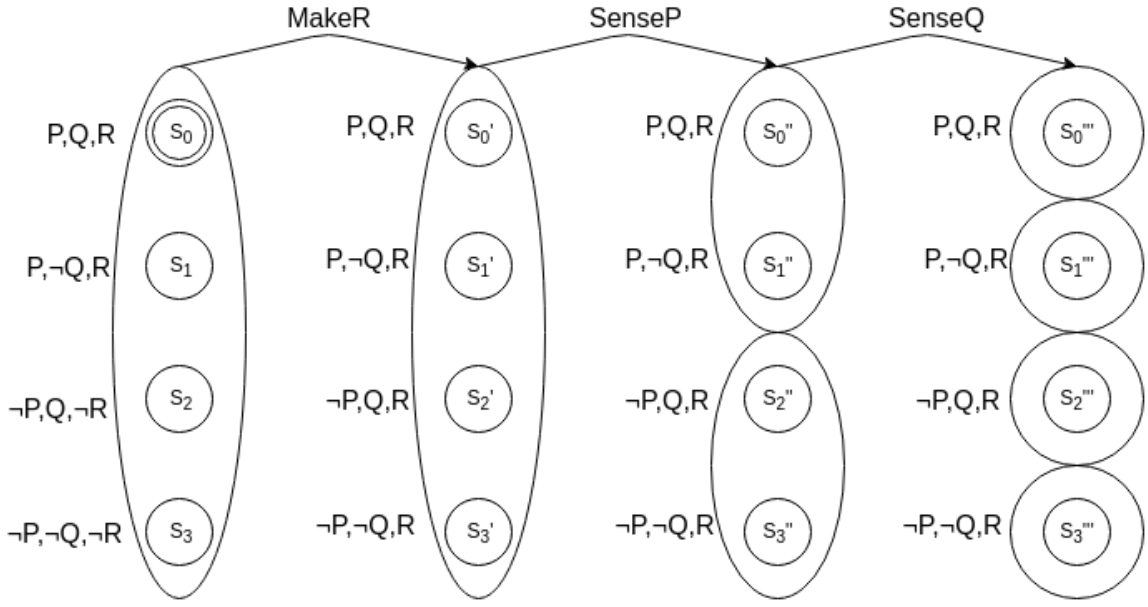

Figure 2.2: A sequence of three actions

Again S_0 is the actual situation as indicated by the double border. Here two situations are K-related exactly if they are in the same elipse. The arrows indicate the $\sqsubset$ relation and are labeled with the actions taken.

In the initial situation there are related situations (S_0, S_3) which disagree on each of the three fluents, thus the agent has no knowledge of the value of any fluents in S_0 (except tautological statements such as $P \vee \neg P$). The agent then performs an action $MakeR$ (which has the effect of making the R fluent true) to arrive in situation S_0'. Observe how R is now satisfied in $S_2' = do(MakeR, S_2)$ and $S_3' = do(MakeR, S_3)$. Since it is the case that R is then satisfied in all accessible situations the agent now knows R.

The second action is $SenseP$, the effect of which is to alter the K fluent so that only situations which agree on the value of the P fluent are related. Thus we have that in S_0'' the agent knows P is indeed true. Additionally had it been the case that P was false (had S_2 or S_3 been the actual situation) the agent would have correctly learned this instead. Finally there is an analogous action $SenseQ$, which senses the Q fluent.

2.4.2 Axioms Related to The Epistemic Fluent

The successor state axiom for K is given as follows:
Axiom 7 (Successor State Axiom for K)**.** [2]
$K(s'', do(a, s)) \equiv \exists s'.K(s', s) \wedge Poss(a, s') \wedge s'' = do(a, s') \wedge sr(a, s) = sr(a, s')$

We adopt the convention of calling an action which modifies the K fluent a "sensing action", and ones which do not we call "non-sensing", "normal", or "physical" actions. The purpose of this division between classes of actions is to guarantee that no physical fluents are affected by sensing actions. This property is useful in proving certain other properties, as will be seen later. Thus in common with Scherl and Levesque[14] we adopt the following axiomatization policy.

Axiomatization Policy: *All actions are to be axiomatized as affecting only either the K fluent or other fluents. The former are to satisfy the IsSensingAct predicate and the latter are to dissatisfy it.*

Unfortunately in order to impose this policy as an axiom we would need to quantify over fluents, which we have not introduced as a sort, making our language second order. Instead we state this as a policy which we require the axiomatizer to conform to rather than as a formal axiom. We will however diverge from Scherl and Levesque by making this distinciton formally part of the language via the predicate $IsSensingAct$. While this predicate may not strictly be necessary it will make some proofs more formally appealing.

[2] This is the same as in Scherl and Levesque[14]

The axiomatizer can guarantee that physical actions do not alter the K fluent by axiomatizing sr to have the same value in all situations for any physical action. This is one direction of the equivalency in the axiomatization policy, and is enforced by the following axiom:

Axiom 8. *For actions a, situations s, and an arbitrary domain object TRUE:*
$\forall a, s. \neg IsSensingAct(a) \rightarrow (sr(a(x_1, \ldots, x_n), s) = \text{TRUE})$

Note that this axiom alone is not sufficient to guarantee the policy is adhered to, as the axiomatizer must still define the $IsSensingAct$ predicate according to the policy.

The effect of Axiom 7 is to divide situations into K-related groups which yield the same sensor result. Given this and the axiomatization policy the following theorems are entailed.

Theorem 1. [3] *Where F is a fluent other than K:*
$\forall a, s. IsSensingAct(a) \rightarrow (F(s) \equiv F(do(a, s)))$

Theorem 2.
$\forall a, s, s'. \neg IsSensingAct(a) \rightarrow (K(do(a, s'), do(a, s)) \equiv K(s', s))$

Proof. Both theorems are immediate from Axiom 7 and the axiomatization policy. □

[3] This is Theorem 1 in [14]

Chapter 3

Proposed Framework

3.1 Description and Motivation

Scherl and Levesque[14] provide a framework in the situation calculus including "knowledge-producing actions" (which we call sensing actions), the purpose of which is to model the agent learning facts about its environment. They demonstrate how their system can be used to prove whether or not the agent knows some fact after some finite sequence of actions. They also demonstrate the agent's positive introspection: if the agent knows something, it knows that it knows it. Finally [14] shows that the agent will be aware of the effects of its actions, including the truth of a fluent after it is sensed. We extend this framework while preserving all of these properties.

Notice that in Scherl and Levesque's definition of knowledge the fluents in the formula in question have no situation arguments. — This is referred to as situation suppression, the reason for which because the situation argument (s) is provided by the definition of $Knows$. However this behaviour of suppression introduces an implicit limitation on the type of knowledge which can be expressed: knowledge of formulae uniform in the situation argument s. Knowledge of non-uniform formulae, as might be written $Knows(F(do(a, s)), s)$, is simply inexpressible. It is our goal to allow for the expression of this type of knowledge.

Particularly we are interested in a method for an agent to know what the effects of its actions will be *before* performing the action. Shapiro et al.[15] discuss how a similar agent can introspect about its *past* beliefs[1], but also describe how unfortunate properties could arise when the agent is allowed to reason about the future. Specifically it was shown how to prove the formula

[1]Shapiro et al. discuss beliefs while Scherl and Levesque discuss knowledge. The difference in naming was chosen as Shapiro models beliefs which may be incorrect, while knowledge is always correct.

$$Bel(\neg\phi \wedge Bel(\phi, do(sense_\phi, now)), S),$$

which reads that the agent believes $\neg\phi$ while simultaneously believing that should it sense the value of the ϕ fluent it would then believe ϕ. This is counterintuitive to Theorem 1, to which Shapiro et al. have a similar property.

It is the goal of this thesis to provide an extension of the approach to knowledge representation presented in Scherl and Levesque[14] in which the agent has introspective knowledge about the future in an intuitively pleasing way. Specifically as in the above example, should the agent be ignorant of the value of a fluent F it will know that should it sense F there would be two mutually exclusive possible outcomes; one where F is true and one where it is false. However the agent does not know which outcome will result from taking the action until it is taken. We refer to this type of reasoning as *prospection*.

More formally we say prospection is having, in a situation s, knowledge of formulae which contain situation terms s' such that $s \sqsubset s'$.

Contrast prospection with progression[6]. Progression and prospection are similar in appearance as both are solutions to the projection problem, which is the problem of deciding whether some formula will be satisfied after a given sequence of executable actions. However they are quite different in operation. Lin and Reiter[6] discuss progression as the process of computing from the basic action theory $\mathcal{D}$ and the set of theorems uniform in S_0 (which they call $\mathcal{D}_{\mathcal{S}_,}$) the set $\mathcal{D}_{\mathcal{S}_\dashv}$: the set of theorems which are uniform is $do(a, S_0)$ for an action a. Prospection requires significantly less logical machinery than progression while natively handling (many) non-uniform formulae. What's more, the progressed database $\mathcal{D}_{\mathcal{S}_\dashv}$ contains no theorems about S_0, while prospection preserves such information. Progression however includes native support for proving model-spanning constraints, such as $\forall s.C(s)$.

Our framework is also able to express *retrospection*: knowledge of formulae containing situations s' such that $s' \sqsubset s$. Shapiro et al.[15] introduce the terminology "prev" to discuss formulae being true in a previous situation. They write $Prev(\phi, s)$ to mean that ϕ held in the situation immediately prior to s. We also wish to reference prior situations, however we take a different approach. Here we introduce the *prev* function as one akin to an inverse to *do*, except with only a situation argument. As a result of Axiom 1 each situation has at most one immediate predecessor and thus the action argument is superfluous. The following axiom defines the *prev* function:

Axiom 9.

$prev(s) = s' \equiv \exists a.s = do(a, s')$

We show in the following sections how we achieve this. However, as is the case in Scherl and Levesque[14], introspection only holds when the K fluent satisfies certain properties. Thus for the remainder of this thesis we will operate in modal logic S5.

Axiom 10 (S5)**.**
$Init(s) \wedge Init(s') \rightarrow K(s', s)$

This axiom together with Axiom 7 are sufficient to show that K is an equivalence relation.

3.2 Relative Formulae

Similar to the previously developed notion of uniform formulae, we introduce relative formulae. Intuitively stated, a formula or term is relative to a situation term σ if all situation terms in that formula are written in terms of σ. That is, any situation term in the formula contains σ as a sub-term. Formally the relative formula are defined as follows.
Definition 7 (Relative Formulae)**.** *The terms relative to a situation term σ are:*

1. *Any term which does not mention a situation term.*
2. *σ.*
3. *S, if S is a constant of sort situation and σ is a situation variable s s.t. $s = S$. E.g. s is relative to S_0 when $s = S_0$.*
4. *If g is an n-ary function symbol and $\tau_1, \ldots, \tau_n$ are terms relative to σ whose sorts are appropriate for g, then $g(\tau_1, \ldots, \tau_n)$.*

Additionally there are three categories of formula relative to σ:

1. *$\tau_1 = \tau_2$, if τ_1 and τ_2 are terms relative to σ.*
2. *Predicates other than $Poss$ and $\sqsubseteq$ whose arguments are terms relative to σ.*
3. *Formulae $\neg\varphi, \varphi \wedge \psi$, and $\exists v.\varphi$ where φ, ψ are formulae relative to σ and v is an individual variable of sort other than situation.*

It is evident that any formula uniform in a situation term σ is also relative to σ. Below are some examples of formulae which are relative to s but not uniform in any situation:

- $F(do(a, s))$
- $(F(s) \rightarrow P(do(a, s)))$
- $F(prev(s)) \equiv F(s)$

We will use the notation $\varphi\langle\sigma\rangle$ to denote the formula which results from modifying φ to be relative to σ. Typically this involves subsituting σ for all situation variables, but there are exceptional cases as defined below. We call $\langle\rangle$ the relativization operator and the expression φ relativized to σ. Formally we define the operator as follows:

Definition 8 (Relativization Operator)**.** *We define* $\langle\rangle$ *recursively in the following cases:*

1. $s\langle\sigma\rangle \stackrel{\text{def}}{=} \sigma$, *when* s *is a variable of sort situation*
2. $v\langle\sigma\rangle \stackrel{\text{def}}{=} v$, *when* v *is a variable of sort other than situation*
3. $f(\tau_1,\ldots,\tau_n)\langle\sigma\rangle \stackrel{\text{def}}{=} f(\tau_1\langle\sigma\rangle,\ldots,\tau_n\langle\sigma\rangle)$, *when* f *is a function or predicate whose arguments are the terms* $\tau_1,\ldots,\tau_n$.
4. $(\tau_1=\tau_2)\langle\sigma\rangle \stackrel{\text{def}}{=} \tau_1\langle\sigma\rangle=\tau_2\langle\sigma\rangle$
5. $(\neg\varphi)\langle\sigma\rangle \stackrel{\text{def}}{=} \neg(\varphi\langle\sigma\rangle)$, *when* φ *is a formula*
6. $(\varphi\wedge\psi)\langle\sigma\rangle \stackrel{\text{def}}{=} \varphi\langle\sigma\rangle\wedge\psi\langle\sigma\rangle$, *when* φ,ψ *are formulae*
7. $(\exists v.\varphi)\langle\sigma\rangle \stackrel{\text{def}}{=} \exists v.(\varphi\langle\sigma\rangle)$, *when* v *is a variable of sort other than situation and* φ *is a formula*
8. *If* ψ *is a formula of the form* $\exists s.\varphi(\tau_1,\ldots,\tau_n)$ *where* s *is a variable of sort situation and* $\varphi(\tau_1,\ldots,\tau_n)$ *is a formula containing situation terms* $\tau_1,\ldots,\tau_n$ *then* $\psi\langle\sigma\rangle \stackrel{\text{def}}{=} \exists s.\varphi(\rho_1,\ldots,\rho_n)$ *where each* ρ_i *is defined as follows:*
   (a) *If* τ_i *is not relative to* s *then* $\rho_i \stackrel{\text{def}}{=} \tau_i\langle\sigma\rangle$.
   (b) *Otherwise* $\rho_i \stackrel{\text{def}}{=} \tau_i\langle\sigma\langle s\rangle\rangle$

Some examples:

- $F(s)\langle s'\rangle = F(s')$
- $F(s)\langle do(a,s)\rangle = F(do(a,s))$
- $F(do(a,s))\langle s'\rangle = F(do(a,s'))$
- $(F(s)\rightarrow P(do(a,s)))\langle s'\rangle = F(s')\rightarrow P(do(a,s'))$
- $(\exists s.F(s))\langle do(a,s')\rangle = \exists s.F(do(a,s))$

Observe that if the relativization operator is applied to a formula which is not relative to begin with then the result is not guaranteed to be a relative formula. The final example given above demonstrates this. The operator is not intended to be applied to non-relative formula in most cases and it advised to take care when doing so. Rather the relativization operator's purpose is to change the situation term to which a formula is relative to the operator's argument, and as such applying the operator to a non-relative formula produces an often counter-intuitive change in meaning.

Formulae which contain quantified situation variables are particularly tricky. Relativization is essentially an action of substitution so strictly speaking we should not be able to substitute

for the quantified variable as it is necessarily bound. However simply substituting all the free situation variables and leaving the bound ones unchanged had undesirable effects on the behaviour of knowledge representation. A previously attempted approach was to simply leave the relativization operator $\langle\rangle$ undefined for formulae quantifying over a situation variable, but if certain useful properties of the overall framework were to be preserved this approach required us to later add special exceptions for formulae of the form given in Definition 11. Instead we have chosen to define the relativization operator such that the desired behaviours are preserved, whence the somewhat unusual Case 8 of Definition 8.

The associative property of the relativization operator is instrumental in proofs in the following section, so we provide it as a theorem.

Theorem 3 (Associativity of $\langle\rangle$)**.** *For a formula φ and situation terms σ, τ:*
$\varphi\langle\sigma\rangle\langle\tau\rangle \equiv \varphi\langle\sigma\langle\tau\rangle\rangle$

Proof. We will show the property holds for each of the cases in Definition 8. The first two cases we can show directly, while the others reduce to these two via recursive invocations of this Theorem.

Case 1

Let $\varphi = s$, where s is a variable of sort situation. Then:

$s\langle\sigma\rangle = \sigma$ by Definition 8 Case 1.
$s\langle\sigma\langle\tau\rangle\rangle = \sigma\langle\tau\rangle$ also by Definition 8 Case 1.
Thus $s\langle\sigma\rangle\langle\tau\rangle = \sigma\langle\tau\rangle = s\langle\sigma\langle\tau\rangle\rangle$

Case 2

Let $\varphi = v$, where v is a variable of sort other than situation. $v\langle\sigma\rangle = v$ for any σ and therefore $v\langle\sigma\rangle\langle\tau\rangle = v\langle\sigma\langle\tau\rangle\rangle$

Case 3

Let $\varphi = f(\rho_1, \ldots, \rho_n)$, where f is a fucntion or predicate whose arguments are terms $\rho_1, \ldots, \rho_n$. We then have:

$f(\rho_1, \ldots, \rho_n)\langle\sigma\rangle\langle\tau\rangle =$
$f(\rho_1\langle\sigma\rangle, \ldots, \rho_n\langle\sigma\rangle)\langle\tau\rangle =$
$f(\rho_1\langle\sigma\rangle\langle\tau\rangle, \ldots, \rho_n\langle\sigma\rangle\langle\tau\rangle)$

We also have:

$f(\rho_1, \ldots, \rho_n)\langle\sigma\langle\tau\rangle\rangle =$
$f(\rho_1\langle\sigma\langle\tau\rangle\rangle, \ldots, \rho_n\langle\sigma\langle\tau\rangle\rangle)$

It suffices to show that for each $\rho_i, \rho_i\langle\sigma\rangle\langle\tau\rangle \equiv \rho_i\langle\sigma\langle\tau\rangle\rangle$, which can be done by n recursive applications of this theorem.

Case 4

Let φ be $\rho_1 = \rho_2$. Then we have:

$(\rho_1 = \rho_2)\langle\sigma\rangle\langle\tau\rangle$
$(\rho_1\langle\sigma\rangle = \rho_2\langle\sigma\rangle)\langle\tau\rangle$
$(\rho_1\langle\sigma\rangle\langle\tau\rangle = \rho_2\langle\sigma\rangle\langle\tau\rangle)$

and also:

$(\rho_1 = \rho_2)\langle\sigma\langle\tau\rangle\rangle =$
$(\rho_1\langle\sigma\langle\tau\rangle\rangle = \rho_2\langle\sigma\langle\tau\rangle\rangle)$

Thus it suffices to show that $\rho_1\langle\sigma\rangle\langle\tau\rangle \equiv \rho_1\langle\sigma\langle\tau\rangle\rangle$ and $\rho_2\langle\sigma\rangle\langle\tau\rangle \equiv \rho_2\langle\sigma\langle\tau\rangle\rangle$, which is done recursively.

Case 5

By Case 5 of Definition 8 we have:

$(\neg\varphi)\langle\sigma\rangle\langle\tau\rangle \equiv$
$(\neg\varphi\langle\sigma\rangle)\langle\tau\rangle \equiv$
$(\neg\varphi\langle\sigma\rangle\langle\tau\rangle)$

and also:

$(\neg\varphi)\langle\sigma\langle\tau\rangle\rangle \equiv$
$(\neg\varphi\langle\sigma\langle\tau\rangle\rangle)$

Since $(\neg\varphi\langle\sigma\rangle\langle\tau\rangle) \equiv (\neg\varphi\langle\sigma\langle\tau\rangle\rangle)$ exactly if $(\varphi\langle\sigma\rangle\langle\tau\rangle) \equiv (\varphi\langle\sigma\langle\tau\rangle\rangle)$, it suffices to show that the former holds. This is done recursivley.

Case 6

Case 6 of Definition 8 gives us:

$(\varphi \wedge \psi)\langle\sigma\rangle\langle\tau\rangle \equiv$
$(\varphi\langle\sigma\rangle \wedge \psi\langle\sigma\rangle)\langle\tau\rangle \equiv$
$(\varphi\langle\sigma\rangle\langle\tau\rangle \wedge \psi\langle\sigma\rangle\langle\tau\rangle)$

as well as

$(\varphi \wedge \psi)\langle\sigma\langle\tau\rangle\rangle \equiv$
$(\varphi\langle\sigma\langle\tau\rangle\rangle \wedge \psi\langle\sigma\langle\tau\rangle\rangle)$

Thus it is sufficient to show that $\varphi\langle\sigma\rangle\langle\tau\rangle \equiv \varphi\langle\sigma\langle\tau\rangle\rangle$ and $\psi\langle\sigma\rangle\langle\tau\rangle \equiv \psi\langle\sigma\langle\tau\rangle\rangle$. Each is a recursive application of this Theorem.

Case 7

Case 7 of Definition 8 allows us to show, where v is of a sort other than situation:

$(\exists v.\varphi)\langle\sigma\rangle\langle\tau\rangle \equiv$
$(\exists v.\varphi\langle\sigma\rangle)\langle\tau\rangle \equiv$
$(\exists v.\varphi\langle\sigma\rangle\langle\tau\rangle)$

and

$(\exists v.\varphi)\langle\sigma\langle\tau\rangle\rangle \equiv$
$(\exists v.\varphi\langle\sigma\langle\tau\rangle\rangle)$

Thus is remains only to show that $\varphi\langle\sigma\rangle\langle\tau\rangle \equiv \varphi\langle\sigma\langle\tau\rangle\rangle$, which is a direct (recursive) application of this Theorem.

Case 8

Let φ be a formula of the form $\exists s.\psi(\rho_1, \ldots, \rho_n)$ where $\rho_1, \ldots, \rho_n$ are terms. By Case 8 of Definition 8 we have the following:

$\varphi\langle\sigma\rangle\langle\tau\rangle \equiv$
$\exists s.\psi(\rho_i\langle\sigma\rangle\langle\tau\rangle, \ldots, \rho_j\langle\sigma\rangle\langle\tau\langle s\rangle\rangle)$

where ρ_j is some ρ which is relative to s, and ρ_i some ρ which is not.

Similarly we have that

$\varphi\langle\sigma\langle\tau\rangle\rangle \equiv$
$\exists s.\psi(\rho_i\langle\sigma\langle\tau\rangle\rangle, \ldots, \rho_j\langle\sigma\langle\tau\langle s\rangle\rangle\rangle)$

Thus it is sufficient to show that

$$\rho_i\langle\sigma\rangle\langle\tau\rangle \equiv \rho_i\langle\sigma\langle\tau\rangle\rangle$$

and that

$$\rho_j\langle\sigma\rangle\langle\tau\langle s\rangle\rangle \equiv \rho_j\langle\sigma\langle\tau\langle s\rangle\rangle\rangle$$

each of which is a recursive application of this Theorem.

□

3.3 Knowledge

3.3.1 Definition of the Knowledge Operator and Knowable Formulae

If we are to achieve our goal of representing knowledge of non-uniform formulae, we cannot suppress fluent situation arguments. Another approach is that by Shapiro et al.[15] which is to use a "dummy" argument *now* for which the situation argument (s) is substituted. Because *now* exists as merely a target for substitution it too is typically suppressed for readability. This is likewise unsuitable for our goal. Consider the following definition of knowledge:

Definition 9 (Knowledge of Relative Formulae)**.** *Let σ be a situation term, φ a formula relative to σ, and s a situation variable which does not appear in φ or σ:*
$Knows(\varphi, \sigma) \stackrel{\text{def}}{=} \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle s\rangle$

Were φ to be situation-suppressed or the situation variables replaced with *now*, the relationship between σ and any situation terms in φ could be lost. It is simplest to avoid suppression entirely.

This definition would work for most formulae and we have nearly arrived at our goal of representing knowledge of non-uniform formulae. However recall from Definition 7 that a formula which quantifies over a variable of sort situation is not a relative formula. For this reason any formula containing $Knows$ as a clause would not be relative to any σ, and thus we cannot express knowledge of it with this definition. This means that with this definition we cannot express introspective knowledge. We will expand the relative formulae as little as possible to also include introspection and call these formulae "knowable".

Definition 10 (Knowable Formulae)**.** *A formula is* knowable *in a situation term σ in the following cases:*

1. *It is relative to σ.*
2. *It is of the form $Knows(\varphi, \tau)$, when φ is knowable in τ and τ is relative to σ.*
3. *$\neg\varphi, \varphi \wedge \psi$, or $\exists v.\varphi$ where φ, ψ are knowable in σ and v is a variable of sort other than situation.*

As is evident from Case 1, all formulae relative to a situation term are also knowable in that situation term. For examples of relative formulae review Definition 7. Some examples of formula which are knowable in s but not relative to s:

- $Knows(F(s), s)$
- $Knows(F(do(a, s)), s)$

- $Knows(F(do(a,s)), do(a,s))$
- $Knows(F(do(a,s)), do(a,s)) \vee Knows(\neg F(do(a,s)), do(a,s))$

With this we can finally give the definition of knowledge.

Definition 11 (Knowledge of Knowable Formulae)**.** *Where φ is a formula knowable in the situation term σ and s a new situation variable:*
$Knows(\varphi, \sigma) \stackrel{\text{def}}{=} \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle s\rangle$

This is identical to Definition 9 except with knowable formula permitted, allowing introspective knowledge. Some examples:

- $Knows(F(s), s)$
- $Knows(Knows(F(do(a,s)), do(a,s)), s)$
- $Knows(\neg Knows(F(s), s) \wedge \neg Knows(\neg F(s), s) \wedge$
  $[Knows(F(do(SenseF, s)), do(SenseF, s)) \vee$
  $Knows(\neg F(do(SenseF, s)), do(SenseF, s))], s)$

While the $Knows$ operator is ultimately a notational shorthand or macro for the RHS of its definition, it would be convenient to be able to act as though it were a predicate when possible. To this end we give the following theorem:

Theorem 4. *Where σ, τ are situation terms and φ a formula knowable in σ:*
$Knows(\varphi, \sigma)\langle\tau\rangle \equiv Knows(\varphi\langle\tau\rangle, \sigma\langle\tau\rangle)$

Proof. We expand the LHS of the theorem to obtain:

$$(\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle s\rangle)\langle\tau\rangle$$

Next we apply the $\langle\tau\rangle$ operator and it is Case 8 of Definition 8 which applies. The situation terms in this existentially quantified formula are $\sigma\langle s\rangle, \sigma$, and $\varphi\langle s\rangle$; the first and third are relative to s invoking Definition 8 Case 8 (b), while the second is not relative to s invoking Case 8 (a). The result is the following:

$$\begin{aligned}
&\forall s.K(\sigma\langle s\rangle\langle\tau\langle s\rangle\rangle, \sigma\langle\tau\rangle) \rightarrow \varphi\langle s\rangle\langle\tau\langle s\rangle\rangle \\
&\equiv \forall s.K(\sigma\langle s\langle\tau\langle s\rangle\rangle\rangle, \sigma\langle\tau\rangle) \rightarrow \varphi\langle s\langle\tau\langle s\rangle\rangle\rangle \\
&\equiv \forall s.K(\sigma\langle\tau\langle s\rangle\rangle, \sigma\langle\tau\rangle) \rightarrow \varphi\langle\tau\langle s\rangle\rangle \\
&\equiv \forall s.K(\sigma\langle\tau\rangle\langle s\rangle, \sigma\langle\tau\rangle) \rightarrow \varphi\langle\tau\rangle\langle s\rangle \\
&\equiv Knows(\varphi\langle\tau\rangle, \sigma\langle\tau\rangle)
\end{aligned}$$

□

We provide the following notational shorthands for convenience.

Definition 12.
$KWhether(\varphi, \sigma) \stackrel{\text{def}}{=} Knows(\varphi, \sigma) \vee Knows(\neg\varphi, \sigma)$
Definition 13.
$Ignorant(\varphi, \sigma) \stackrel{\text{def}}{=} \neg KWhether(\varphi, \sigma)$

This allows the third example of an expression of knowledge above to be rewritten in a significantly more compact and readable form:

$$Knows(Ignorant(F(s), s) \wedge KWhether(F(do(SenseF, s)), do(SenseF, s)), s)$$

We see that Scherl and Levesque's definition[14] of $Knows$ remains as a special case of this definition: the case where σ is a situation variable and φ is uniform in σ. Additionally we can derive several new, useful forms from this definition.

Knowledge About the Future

Let us consider the simple case of knowing something about the immediate future. This form allows us to express hypothetical knowledge about the results of an action.
Theorem 5. *Let σ be a situation term, $\varphi\langle do(a, \sigma)\rangle$ be a formula knowable in $do(a, \sigma)$, and a an action:*
$Knows(\varphi\langle do(a, \sigma)\rangle, \sigma) \equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle do(a, \sigma\langle s\rangle)\rangle$

Observe that $do(a, \sigma)$ is relative to and thus knowable in σ. This theorem states that exactly if in all situations accessible from σ, φ will hold after performing a in that situation, then the agent knows in σ that φ will hold after performing a. The agent knows about the future.

Proof. The LHS expression $Knows(\varphi\langle do(a, \sigma)\rangle, \sigma)$ is by Definition 11 a shorthand for

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle do(a, \sigma)\rangle\langle s\rangle$$

By Theorem 3 this is equivalent to:

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle do(a, \sigma)\langle s\rangle\rangle$$
$$\equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle do(a, \sigma\langle s\rangle)\rangle$$

□

Example: Consider again Figure 2.2. Initially the agent has no knowledge of R:

$$Ignorant(R(S_0), S_0)$$

However we can prove

$$Knows(R(do(MakeR, S_0)), S_0)$$

The expression $R(do(MakeR, S_0))$ is relative to and thus knowable in S_0, so we can apply Theorem 5 to expand this expression to:

$$\forall s.K(s, S_0) \rightarrow R(do(MakeR, s))$$

This states that if R would be satisfied after executing $MakeR$ in all situations accessible from S_0, then the agent knows in S_0 that R would become true should it perform $MakeR$.

Introspection on Future Knowledge

Theorem 5 does not exclude facts about the agent's own knowledge. If we let φ in the previous theorem be a $Knows$ expression we can express the agent having hypothetical knowledge about what it would know or learn as a result of an action even before that action is taken. We write the corollary as follows:

Corollary 1. *Let σ be a situation term, a an action, and φ a formula knowable in do(a,σ):* $Knows(Knows(\varphi, do(a, \sigma)), \sigma) \equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow Knows(\varphi\langle s\rangle, do(a, \sigma\langle s\rangle))$

This reads that the agent knows in situation σ that it will know φ in the situation resulting from performing action a exactly when in all situations accessible from σ the agent will know φ after performing a.

Proof. Substitute the formula $Knows(\varphi, do(a, \sigma))$ for $\varphi\langle do(a, \sigma)\rangle$ into Theorem 5 to produce:

$$\begin{aligned}
&Knows(Knows(\varphi, do(a, \sigma)), \sigma) \equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow Knows(\varphi, do(a, \sigma))\langle s\rangle \\
&\equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow Knows(\varphi\langle s\rangle, do(a, \sigma)\langle s\rangle) \\
&\equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow Knows(\varphi\langle s\rangle, do(a, \sigma\langle s\rangle))
\end{aligned}$$

□

Example: Similar to the previous example, consider the first action in Figure 2.2. We wish to show that the agent knows in the initial situation that should it perform $MakeR$, it would then know R. One might attempt to write this as $Knows(Knows(R(S_0'), S_0'), S_0)$, but this is not in the form required by Corollary 1. Indeed the term $Knows(R(S_0'), S_0')$ is not knowable in S_0 as S_0' is not relative to S_0. The correct form is the equivalent statement:

$$Knows(Knows(R(do(MakeR, S_0)), do(MakeR, S_0)), S_0)$$

By Corollary 1 this expands to:

$$\begin{aligned}
&\forall s.K(S_0\langle s\rangle, S_0) \rightarrow Knows(R(do(MakeR, S_0))\langle s\rangle, do(MakeR, S_0\langle s\rangle)) \\
&\equiv \forall s.K(s, S_0) \rightarrow Knows(R(do(MakeR, s)), do(MakeR, s)) \\
&\equiv \forall s.K(s, S_0) \rightarrow (\forall s'.K(do(MakeR, s'), do(MakeR, s) \rightarrow R(do(MakeR, s')))
\end{aligned}$$

The situations K-related to S_0 are $S_0 \ldots S_3$, and we see that in each of their successor situations ($S'_0 \ldots S'_3$) it is the case that R is known in that situation (R is satisfied in every related situation $S'_0 \ldots S'_3$) as desired.

Knowledge of the Effects of Sensing

Related to the previous corollary, a particularly interesting result occurs when the agent performs a sensing action. We can express that the agent is aware of how the sensing action will hypothetically affect its knowledge: it knows that it will have correct knowledge of the sensed fluent after sensing it. This is our answer to the previously discussed problem posed by Shapiro et al. in [15] where knowledge of the affects of sensing actions revealed to the agent the value of the sensed fluent before the action was even performed — not so here. The corollary is as follows:

Corollary 2. *Let σ be a situation term, a an action, and φ a formula knowable in $do(a, \sigma)$*
$Knows(KWhether(\varphi, do(a, \sigma)), \sigma) \equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow KWhether(\varphi\langle s\rangle, do(a, \sigma\langle s\rangle))$

This reads that the agent will know in σ that should it perform a, it will then know whether φ is true or false exactly when it knows whether φ in the situations $do(a, s)$ for every situation s accessible from σ. Observe that the corrollary does not strictly require the action a to be a sensing action ($IsSensingAct(a)$). This corollary is slightly more general and there are three possible cases where it will hold:

1. The agent already knows whether φ in σ and a does not affect this.
2. The effect of a is to make φ true (or false), in which case this corollary trivially follows from Corollary 1.
3. The effect of a is to reveal whether φ is true or false (it is a sensing action for φ).

It is primarily Case 3 which we find interesting.

Proof. This is similar to Corollary 1 except with $KWhether$ in place of the inner $Knows$. As $KWhether$ is merely a notational shorthand for the disjunction of two $Knows$ terms the proof is analogous to that for Corollary 1. □

Example: Case 2 is exemplified in example for Corollary 1 where the agent knows that it will know R, and thus know whether R. For the other two cases observe in Figure 2.2 when the agent performs $SenseP$.

We will exemplify Case 1 by considering the fluent R. In S'_0 the agent already knows whether R (it knows R to be true), and the $SenseP$ action has no affect on this — R is still true in every successor situation. Thus the corollary holds and we can prove

$$Knows(R(do(SenseP, S_0')), S_0')$$

For Case 3 consider the fluent P. In S_0' the agent is ignorant of P:

$$Ignorant(P(S_0'), S_0') \equiv \neg Knows(P(S_0'), S_0') \wedge \neg Knows(\neg P(S_0'), S_0')$$

However the agent does know that should it perform $SenseP$, it will know whether P is true or false afterwards. We can see this is the case because in each of the situations K-related to S_0' ($S_0' \dots S_3'$) it is the case that after performing $SenseP$ the agent will either know P or know $\neg P$.

However the agent does not know which until the action is performed, as either case could be true according to the agent's knowledge in S_0'. Specifically we will show that $Knows(P(do(SenseP, S_0')), S_0') \vee Knows(\neg P(do(SenseP, S_0')), S_0')$ does not hold. By Theorem 5 we have the following expansion:

$$Knows(P(do(SenseP, S_0')), S_0') \equiv \forall s.K(s, S_0') \rightarrow P(do(SenseP, s))$$

Let $s = S_2'$ however and we see the statement does not hold. A similar deduction with $s = S_0'$ shows the right disjunct does not hold either, so the statement is false as desired.

Equivalent Foreknowledge

The final form we will present regarding knowledge of the future is that (bi-)conditional on the present. We will use this form to express knowledge of the form "I know that Q will be true after I perform a exactly if P is true now (even though I don't currently know whether P)".

Theorem 6. *Where $\psi\langle\sigma\rangle$ is a formula knowable in situation term σ, a is an action, and $\varphi\langle do(a, \sigma)\rangle$ is a formula knowable in $do(a, \sigma)$*
$Knows(\psi\langle\sigma\rangle \equiv \varphi\langle do(a, \sigma)\rangle, \sigma) \equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\psi\langle\sigma\langle s\rangle\rangle \equiv \varphi\langle do(a, \sigma\langle s\rangle)\rangle)$

Consider an informal example. Suppose the agent and some friends are playing a game where each player has a canned drink, exactly one of which has been shaken violently but no player knows which. The players open their cans simultaneously resulting in one player getting wet. With this theorem we can prove a formula which models the following reasoning: "I don't know if my can is the shaken one or it is not. If it is, then when I pull this tab I will get wet and know that my can was the shaken one all along. Otherwise I will remain dry and know that my can was unshaken."

Proof. Substituting the formula $\psi\langle\sigma\rangle \equiv \varphi\langle do(a, \sigma)\rangle$ for φ into Definition 11:

$$Knows(\psi\langle\sigma\rangle \equiv \varphi\langle do(a, \sigma)\rangle, \sigma)$$
$$\equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\psi\langle\sigma\rangle \equiv \varphi\langle do(a, \sigma)\rangle)\langle s\rangle$$

$$\equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\psi\langle\sigma\rangle\langle s\rangle \equiv \varphi\langle do(a,\sigma)\rangle\langle s\rangle)$$
$$\equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\psi\langle\sigma\langle s\rangle\rangle \equiv \varphi\langle do(a,\sigma)\langle s\rangle\rangle)$$
$$\equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\psi\langle\sigma\langle s\rangle\rangle \equiv \varphi\langle do(a,\sigma\langle s\rangle)\rangle)$$

□

Example: Let us examine the second action taken in Figure 2.2: $SenseP$. The goal of the theorem is to express that in S_0' the agent knows that, conditional on P being true, it will know after performing $SenseP$ that not only is P true, but that it was true all along — even though the agent does not know whether P is true before sensing.

We write the conditional knowledge as:

$$Knows(P(S_0') \equiv Knows(\\ P(do(SenseP, S_0')) \wedge P(prev(do(SenseP, S_0'))), do(SenseP, S_0')), S_0')$$

which by Theorem 6 expands to:

$$\forall s.K(S_0'\langle s\rangle, S_0') \rightarrow \\ (P(S_0'\langle s\rangle) \equiv Knows(\\ P(do(SenseP, S_0'\langle s\rangle)) \wedge P(prev(do(SenseP, S_0'\langle s\rangle))), do(SenseP, S_0'\langle s\rangle)))$$
$$\equiv \forall s.K(s, S_0') \rightarrow \\ (P(s) \equiv (\forall s'.K(do(SenseP, s)\langle s'\rangle, do(SenseP, s)) \rightarrow \\ (P(do(SenseP, s)\langle s'\rangle) \wedge P(prev(do(SenseP, s))\langle s'\rangle))))$$
$$\equiv \forall s.K(s, S_0') \rightarrow \\ (P(s) \equiv (\forall s'.K(do(SenseP, s'), do(SenseP, s)) \rightarrow \\ (P(do(SenseP, s')) \wedge P(prev(do(SenseP, s'))))))$$

In all situations related to S_0' where P is satisfied (S_0', S_1'), it is the case that in the successor situation resulting from performing $SenseP$ (S_0'', S_1'') all related situations satisfy P (meaning the agent knows P). Additionally in all of these situations P is satisfied in the previous situation meaning the agent knows P was true before taking the action, as desired. Inversely in those situations which dissatisfy P (S_2', S_3'), after performing $SenseP$ all related situations (and their predecessors) also dissatisfy P, satisfying the equivalency.

Clearly a similar theorem for $\rightarrow$ instead of $\equiv$ follows immediately from Theorem 6. We state it explicitly for future use:

Corollary 3.
$Knows(\psi\langle\sigma\rangle \rightarrow \varphi\langle do(a,\sigma)\rangle, \sigma) \equiv \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\psi\langle\sigma\langle s\rangle\rangle \rightarrow \varphi\langle do(a,\sigma\langle s\rangle)\rangle)$

Proof. Analogous to the proof of Theorem 6. □

Memory of Past Knowledge

Theorem 7. *For a situation σ and formula $\varphi\langle prev(\sigma)\rangle$ knowable in $prev(\sigma)$:*
$(\neg Init(\sigma) \wedge Knows(\varphi\langle prev(\sigma)\rangle, prev(\sigma))) \rightarrow Knows(Knows(\varphi\langle prev(\sigma)\rangle, prev(\sigma)), \sigma)$

This states that if the agent knew something in a given situation, then it knows after performing an action that it previously knew that something.

Before proving the theorem we will first show the lemma that if two situations are K-related then so were their predecessors (if they exist):
Lemma 1. [2] $\neg(Init(s) \vee Init(s')) \rightarrow K(s', s) \rightarrow K(prev(s'), prev(s))$

Proof. The successor state axiom for K (Axiom 7) is an equivalency, but we will restate just one direction of the implication:

$$K(s'', do(a, s)) \rightarrow \exists s'.K(s', s) \wedge Poss(a, s') \wedge s'' = do(a, s') \wedge sr(a, s) = sr(a, s')$$

Assuming the action a to be possible we simplify to:

$$K(do(a, s'), do(a, s)) \rightarrow K(s', s) \wedge sr(a, s) = sr(a, s')$$

By Axiom 9 we restate this as:

$$K(s', s) \rightarrow K(prev(s'), prev(s)) \wedge sr(a, prev(s)) = sr(a, prev(s'))$$

Ignoring the right-hand conjunct of the consequent gives the lemma. □

We may now prove the theorem.

Proof. The theorem is vacuously true when the antecedent is false, so let us consider the case where it is true. Thus σ is not an initial situation and the situation $prev(\sigma)$ exists. By Definition 11 the consequent expands to:

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow Knows(\varphi\langle prev(\sigma)\rangle\langle s\rangle, prev(\sigma)\langle s\rangle)$$

If $\sigma\langle s\rangle$ is not K-related to σ for some s then the quantified statement is vacuously true. If they are K-related then by the lemma $prev(\sigma\langle s\rangle)$ is K-related to $prev(\sigma)$. As we are assuming the antecedent of the theorem to be true $\varphi\langle prev(\sigma)\rangle$ is known in $prev(\sigma)$. Thus it is satisfied and known in every situation K-related to $prev(\sigma)$, namely $prev(\sigma\langle s\rangle)$, satisfying the quantified statement. Therefore the statement is satisfied for all values of s. □

Example: Consider for a final time Figure 2.2. After performing $SenseP$, the agent knows that it previously knew R. We write this as:

[2] It follows from this Lemma that actions cannot introduce uncertainty. This was also the case in Scherl and Levesque[14]. While not explicitly claimed it is a clear consequence of the proven theorems.

$Knows(Knows(R(prev(S_1'')), prev(S_1'')), S_1'')$

As shown previously the agent knows R in $prev(S_1'') = S_1'$. In each of the situations accessible from S_1'' it is the case that the agent knew R in the previous situation (S_1' and S_2').

Chapter 4

An Extended Example

In this chapter we will model a simple world and follow the agent through a sequence of actions which take it from complete ignorance to knowledge of all fluents. Several of these steps will demonstrate the agent posessing non-uniform knowledge as is not representable in Scherl and Levesque[14]. This world consists of an agent in one of two rooms, each of which has a light that may be on or off. We further add a light switch to each room and a door separating the rooms. We model this world with four fluents:

- InR_1
- $Light_1$
- $Light_2$
- $DoorOpen$

InR_1 represents which room the agent occupies in a given situation; when true the agent is in Room 1, and when false the agent is in Room 2. $Light_i$ indicates the status of the light (and light switch) in Room i. When true the light is on and the switch is in the "on" position, and when false the light is off with the switch in the "off" position. We do not have fluents for (or otherwise explicitly model) the light switches as their state is immediately implicit from the state of the lights and they are not pertinent to the example. Finally $DoorOpen$ is true when the door between rooms is open, and false when it is closed. We make simplifying assumptions such as the lights or light switches never burning out or otherwise malfunctioning, the agent being unable to get stuck halfway between rooms, and the door being always either completely open permitting traversal or completely closed prohibiting it.

The actions available to the agent are:

- $SwitchRoom$
- $FlipSwitch$
- $OpenDoor$
- $CloseDoor$
- $SenseLight$
- $SenseRoom$
- $SenseDoor$

$SwitchRoom$ should be interpreted as the agent leaving whichever room it is in to enter the other one, which can only be done if the door is open at that time. $FlipSwitch$ toggles the light switch (and thus light) in whichever room the agent is currently in. $OpenDoor$ and $CloseDoor$ cause the door between rooms to become open or closed respectively, and have no effect if the door is already in that position. $SenseLight$ informs the agent of whether or not the light in its current room is on or off at the time of execution, though does not reveal which light is being sensed. $SenseRoom$ informs the agent of which room it is currently occupying. Finally $SenseDoor$ informs the agent of whether the door is open or closed.

4.1 Axioms

4.1.1 Action Precondition Axioms

First we give the action precondition axioms as per Definition 3. Since in this example the only action with preconditions is $SwitchRoom$ we will omit most of the precondition axioms for brevity.

$$Poss(SwitchRoom, s) \equiv DoorOpen(s)$$
$$Poss(FlipSwitch, s) \equiv \top$$
$$\ldots$$
$$Poss(SenseDoor, s) \equiv \top$$

We justify this by assuming that the door is always within reach, as is the light switch in the current room (but not the light switch in the other room), and that the agent's sensors are always available to it.

4.1.2 Successor State Axioms

The effects of these actions on the world (fluents) is given by the successor state axioms.

$$InR_1(do(a,s)) \equiv a = SwitchRoom \not\equiv InR_1(s)$$

The agent will be in Room 1 if it is currently not in Room 1 (i.e. it is in Room 2) and it performs the *SwitchRoom* action, or if it is already in Room 1 and any other action is performed.

$$\begin{aligned} &Light_1(do(a,s)) \equiv \\ &\quad (InR_1(s) \wedge \neg Light_1(s) \wedge a = FlipSwitch) \vee \\ &\qquad (Light_1(s) \wedge \neg(InR_1(s) \wedge a = FlipSwitch)) \end{aligned}$$

$$\begin{aligned} &Light_2(do(a,s)) \equiv \\ &\quad (\neg InR_1(s) \wedge \neg Light_2(s) \wedge a = FlipSwitch) \vee \\ &\qquad (Light_2(s) \wedge \neg(\neg InR_1(s) \wedge a = FlipSwitch)) \end{aligned}$$

A light will be on if it is currently off when the agent is in the same room as that light and performs the *FlipSwitch* action, or if the light is already on and the agent performs any action other than flipping this switch.

$$\begin{aligned} &DoorOpen(do(a,s)) \equiv (\neg DoorOpen(s) \wedge a = OpenDoor) \vee \\ &\quad (DoorOpen(s) \wedge a \neq CloseDoor) \end{aligned}$$

The door will be open if it is currently closed and the action is to open it, or if it is already open and the action is any other than to close it.

Sensor Results

Here we give axioms defining the *sr* function for this domain.

$$sr(SenseDoor, s) = r \equiv (DoorOpen(s) \wedge r = \text{TRUE}) \vee (\neg DoorOpen(s) \wedge r = \text{FALSE})$$

$$sr(SenseRoom, s) = r \equiv (InR_1(s) \wedge r = \text{TRUE}) \vee (\neg InR_1(s) \wedge r = \text{FALSE})$$

The sensing actions *SenseDoor* and *SenseRoom* are simple: *sr* returns TRUE when the relevant fluent is true and FALSE otherwise, effectively separating the two possible cases.

$$\begin{aligned} &sr(SenseLight, s) = r \equiv \\ &\quad (((InR_1(s) \wedge Light_1(s)) \vee (\neg InR_1(s) \wedge Light_2(s))) \wedge r = \text{TRUE}) \vee \\ &\quad (((InR_1(s) \wedge \neg Light_1(s)) \vee (\neg InR_1(s) \wedge \neg Light_2(s))) \wedge r = \text{FALSE}) \end{aligned}$$

The *SenseLight* sensing action is slightly more complicated, separating situations into those where the light in the agent's current room is on, and those where it is off.

$sr(SwitchRoom, s) = \text{TRUE}$
$sr(FlipSwitch, s) = \text{TRUE}$
$sr(OpenDoor, s) = \text{TRUE}$
$sr(CloseDoor, s) = \text{TRUE}$

As per the axiomatization policy and Axiom 8, sr always returns TRUE for non-sensing actions and we define the $IsSensingAct$ predicate accordingly:

$IsSensingAct(SenseDoor)$
$IsSensingAct(SenseRoom)$
$IsSensingAct(SenseLight)$
$\neg IsSensingAct(CloseDoor)$
$\neg IsSensingAct(OpenDoor)$
$\neg IsSensingAct(FlipSwitch)$
$\neg IsSensingAct(SwitchRoom)$

4.2 Initial State

We wish to begin with the agent having no knowledge (other than of tautologies). We do this by defining an initial situation for each possible truth assignment of the fluents, e.g.:

$$\exists s_4.Init(s_4) \wedge \neg DoorOpen(s_4) \wedge \neg Light_1(s_4) \wedge Light_2(s_4) \wedge InR_1(s_4)$$

We would need 16 such statements but we will omit them for brevity. Instead, please refer to the following table. Note that we identify the second situation as the actual situation.

Label	DoorOpen	Light_1	Light_2	InR_1
S_1	F	F	F	F
$S_2 = S_0$	F	F	F	T
S_3	F	F	T	F
S_4	F	F	T	T
S_5	F	T	F	F
S_6	F	T	F	T
S_7	F	T	T	F
S_8	F	T	T	T
S_9	T	F	F	F
S_{10}	T	F	F	T
S_{11}	T	F	T	F
S_{12}	T	F	T	T
S_{13}	T	T	F	F
S_{14}	T	T	F	T
S_{15}	T	T	T	F
S_{16}	T	T	T	T

Table 4.1: The Initial State

By having all of these situations be mutually K-related (including reflexively), it is the case that for any fluent F there is a situation s such that $K(s, S_0) \wedge F(S_0) \not\equiv F(s)$. Therefore there is no fluent F for which $KWhether(F(S_0), S_0)$, and the agent has no knowledge — as desired.

4.2.1 Taking Actions

The ongoing example which comprises this chapter will demonstrate a single sequence of actions. For the remainder of the chapter we will use a superscript on situation variables to indicate the number of actions which have occurred prior to that situation. E.g. the successor situation of S_0 in this sequence will be denoted S_0^1.

Observe that we have $\neg Knows(Poss(SwitchRoom, S_0), S_0)$ and so we will not examine $do(SwitchRoom, S_0)$.[1] Let us then consider a non-sensing action which is known to be possible, such as $CloseDoor$. By the successor state axioms, $CloseDoor$ will have the following result:

[1]This is what Reiter refers to as a "ghost situation". The situation exists but cannot be accessed by an executable sequence of actions.

Label	DoorOpen	Light_1	Light_2	InR_1
S_1^1	F	F	F	F
S_0^1	F	F	F	T
S_3^1	F	F	T	F
S_4^1	F	F	T	T
S_5^1	F	T	F	F
S_6^1	F	T	F	T
S_7^1	F	T	T	F
S_8^1	F	T	T	T
S_9^1	F	F	F	F
S_{10}^1	F	F	F	T
S_{11}^1	F	F	T	F
S_{12}^1	F	F	T	T
S_{13}^1	F	T	F	F
S_{14}^1	F	T	F	T
S_{15}^1	F	T	T	F
S_{16}^1	F	T	T	T

Table 4.2: Closed Door

As this is a non-sensing action it follows from Theorem 2 that all of these situations are K-related.

As the door is closed in all possible situations we have by Theorem 5:

$$Knows(\neg DoorOpen, do(CloseDoor, S_0))$$

Further, by Corollary 1, the agent knows in S_0 that it will know the door is closed after closing it:

$$Knows(Knows(\neg DoorOpen(do(CloseDoor, S_0)), do(CloseDoor, S_0)), S_0)$$

Conditional Knowledge

Consider now the action of flipping the light switch. By the successor state axioms the result of this is as follows:

Label	DoorOpen	Light_1	Light_2	InR_1
S_1^2	F	F	T	F
S_0^2	F	T	F	T
S_3^2	F	F	F	F
S_4^2	F	T	T	T
S_5^2	F	T	T	F
S_6^2	F	F	F	T
S_7^2	F	T	F	F
S_8^2	F	F	T	T
S_9^2	F	F	T	F
S_{10}^2	F	T	F	T
S_{11}^2	F	F	F	F
S_{12}^2	F	T	T	T
S_{13}^2	F	T	T	F
S_{14}^2	F	F	F	T
S_{15}^2	F	T	F	F
S_{16}^2	F	F	T	T

Table 4.3: Light Switch Flipped

Again all of these situations are K-related. We see that while the agent has not learned with certainty anything about the environment after performing this action, the agent still has conditional knowledge.

The successor state axiom for InR_1 entails that only the action SwitchRoom can alter which room the agent is in. Thus we have

$$\forall s.InR_1(s) \equiv InR_1(do(FlipSwitch_1, s))$$
$$\equiv \forall s.K(s, S_0^1) \rightarrow [InR_1(s) \equiv InR_1(do(FlipSwitch, s))]$$

Which Theorem 6 states is equivalent to:

$$Knows(InR_1(S_0^1) \equiv InR_1(do(FlipSwitch, S_0^1)), S_0^1).$$

Similar deductions yield similar conditional knowledge about the state of the lights:

- $Knows((InR_1(S_0^1) \equiv (Light_1(S_0^1) \not\equiv Light_1(do(FlipSwitch, S_0^1)))), S_0^1)$
- $Knows((\neg InR_1(S_0^1) \equiv (Light_2(S_0^1) \not\equiv Light_2(do(FlipSwitch, S_0^1)))), S_0^1)$

Which is to say that while the agent doesn't know which room it is in, which light it just toggled, or whether that light is now on or off, the agent does know that it is in the same room as before and the light in whichever room it is in has the opposite state as before. What's more it knew all these effects of the $FlipSwitch$ action before taking it.

Sensing Actions

At last we will examine a sensing action, *SenseLight*. As a sensing action, only the epistemic fluent K is affected (Theorem 1). Given the definition of sr for *SenseLight* the successor state axiom for K will separate the next situations into two groups: those where the light in the current room is on, and those where it is off. Below we have grouped mutually K-related situations by row colour and re-ordered the rows to place related situations next to each other. To support monochrome printing and the colourblind we also add a sixth column for the K fluent wherein two situations are K-related exactly when they have the same value in this column.

Label	DoorOpen	Light_1	Light_2	InR_1	K
S_1^3	F	F	T	F	A
S_0^3	F	T	F	T	A
S_4^3	F	T	T	T	A
S_5^3	F	T	T	F	A
S_9^3	F	F	T	F	A
S_{10}^3	F	T	F	T	A
S_{12}^3	F	T	T	T	A
S_{13}^3	F	T	T	F	A
S_3^3	F	F	F	F	B
S_6^3	F	F	F	T	B
S_7^3	F	T	F	F	B
S_8^3	F	F	T	T	B
S_{11}^3	F	F	F	F	B
S_{14}^3	F	F	F	T	B
S_{15}^3	F	T	F	F	B
S_{16}^3	F	F	T	T	B

Table 4.4: Light Sensed

In this case we can apply a combination of Corollaries 2 and 3 to show that the agent knows before sensing that it will know whether the light in its current room is on or not after using the sensor:

- $Knows(InR_1(S_0^2) \rightarrow$
  $KWhether(Light_1(do(SenseLight, S_0^2)), do(SenseLight, S_0^2)), S_0^2)$
- $Knows(\neg InR_1(S_0^2) \rightarrow$
  $KWhether(Light_2(do(SenseLight, S_0^2)), do(SenseLight, S_0^2)), S_0^2)$

Yet before sensing the agent does not know which it will be!

$Ignorant(InR_1(S_0^2), S_0^2)$

Completing Conditional Knowledge: Sensing the Room

In S_0^3 the agent has only conditional knowledge about the light in its room. Let us now sense which room we are in and see how this affects the agent's knowledge. As *SenseRoom* is a sensing action, only the K fluent is affected. By the successor state axiom for K and the definition of sr for *SenseRoom*, the result is as follows.

Label	DoorOpen	Light_1	Light_2	InR_1	K
S_0^4	F	T	F	T	A
S_4^4	F	T	T	T	A
S_{10}^4	F	T	F	T	A
S_{12}^4	F	T	T	T	A
S_1^4	F	F	T	F	B
S_5^4	F	T	T	F	B
S_9^4	F	F	T	F	B
S_{13}^4	F	T	T	F	B
S_3^4	F	F	F	F	C
S_7^4	F	T	F	F	C
S_{11}^4	F	F	F	F	C
S_{15}^4	F	T	F	F	C
S_6^4	F	F	F	T	D
S_8^4	F	F	T	T	D
S_{14}^4	F	F	F	T	D
S_{16}^4	F	F	T	T	D

Table 4.5: Room Sensed

We have again grouped situations which are K-related by colour and re-ordered the rows of the table to place related situations together. From this we see that the agent knew before taking this action that it would know after taking it which room it was in, by Corollary 2:

$$Knows(KWhether(InR_1(do(SenseRoom, S_0^3)), do(SenseRoom, S_0^3)), S_0^3)$$

Indeed in S_0^3 the agent knows that in S_0^4 it will either know it is in Room 1 and that Light 1 is on while maintaining ignorance of Light 2, or that it is in Room 2 and that Light 2 is on while maintaining ignorance of Light 1.

$$\begin{array}{l} Knows(\\ \quad Knows(\\ \quad\quad (InR_1(do(SenseRoom, S_0^3)) \wedge Light_1(do(SenseRoom, S_0^3)) \wedge \\ \quad\quad\quad Ignorant(Light_2(do(SenseRoom, S_0^3)), do(SenseRoom, S_0^3))) \vee \\ \quad\quad (\neg InR_1(do(SenseRoom, S_0^3)) \wedge Light_2(do(SenseRoom, S_0^3)) \wedge \\ \quad\quad\quad Ignorant(Light_1(do(SenseRoom, S_0^3)), do(SenseRoom, S_0^3))), \\ \quad do(SenseRoom, S_0^3)), \\ S_0^3) \end{array}$$

This confirms that the agent's previous conditional knowledge was correct.

In this case it is the fact that the agent is in room 1. The only fluent remaining to complete the agent's knowledge is the light in Room 2.

Opening the Door

We must make our way into Room 2 to determine the status of the light, and to do this the door must be open. The K fluent is not changed by a non-sensing action and so by the successor state axiom for *OpenDoor* we have the following result:

Label	DoorOpen	Light_1	Light_2	InR_1	K
S_0^5	T	T	F	T	A
S_4^5	T	T	T	T	A
S_{10}^5	T	T	F	T	A
S_{12}^5	T	T	T	T	A
S_1^5	T	F	T	F	B
S_5^5	T	T	T	F	B
S_9^5	T	F	T	F	B
S_{13}^5	T	T	T	F	B
S_3^5	T	F	F	F	C
S_7^5	T	T	F	F	C
S_{11}^5	T	F	F	F	C
S_{15}^5	T	T	F	F	C
S_6^5	T	F	F	T	D
S_8^5	T	F	T	T	D
S_{14}^5	T	F	F	T	D
S_{16}^5	T	F	T	T	D

Table 4.6: Door Open

It is an immediate application of Theorem 5 to show that the agent knows before opening the door that it will be open after doing so.

$$Knows(Knows(DoorOpen(do(OpenDoor(S_0^4))), do(OpenDoor(S_0^4))), S_0^4)$$

Moving Between Rooms

Now that the door is open $SwitchRoom$ is possible. By the successor state axiom for InR_1, we have the following result:

Label	DoorOpen	Light_1	Light_2	InR_1	K
S_0^6	T	T	F	F	A
S_4^6	T	T	T	F	A
S_{10}^6	T	T	F	F	A
S_{12}^6	T	T	T	F	A
S_1^6	T	F	T	T	B
S_5^6	T	T	T	T	B
S_9^6	T	F	T	T	B
S_{13}^6	T	T	T	T	B
S_3^6	T	F	F	T	C
S_7^6	T	T	F	T	C
S_{11}^6	T	F	F	T	C
S_{15}^6	T	T	F	T	C
S_6^6	T	F	F	F	D
S_8^6	T	F	T	F	D
S_{14}^6	T	F	F	F	D
S_{16}^6	T	F	T	F	D

Table 4.7: Changed Room

We see that the agent now knows that it is in Room 2, but is still ignorant of the status of the light.

Completing The Agent's Knowledge

Finally the agent can sense the light in Room 2:

Label	DoorOpen	Light_1	Light_2	InR_1	K
S_0^7	T	T	F	F	A
S_{10}^7	T	T	F	F	A
S_4^7	T	T	T	F	B
S_{12}^7	T	T	T	F	B
S_1^7	T	F	T	T	C
S_9^7	T	F	T	T	C
S_5^7	T	T	T	T	D
S_{13}^7	T	T	T	T	D
S_3^7	T	F	F	T	E
S_{11}^7	T	F	F	T	E
S_7^7	T	T	F	T	F
S_{15}^7	T	T	F	T	F
S_6^7	T	F	F	F	G
S_{14}^7	T	F	F	F	G
S_8^7	T	F	T	F	H
S_{16}^7	T	F	T	F	H

Table 4.8: Complete Knowledge

Again, K-related situations are grouped together and given the same colour. We see that at every step of this sequence of actions the agent has maintained correct knowedge and behaved in an intuitively pleasing way, even when considering uncertain knowledge about the future.

Chapter 5

Properties and Theorems

In this chapter we will demonstrate several theorems or properties of this framework. Each theorem which appears in section 6.1 of Scherl and Levesque[14] is reproduced here in an analogous (and possibly more general) form in order to show that all epistemic properties of that framework also exist in ours. Where possible Scherl and Levesque's proofs of these theorems are used in whole or in part.

5.1 Introspection

We wish to show that if our agent knows some fact, then they know that they know it. Likewise if they do not know something they know that they do not know it. These properties are respectively known as positive and negative introspection. This is the same result as in Scherl and Levesque[14] and we present a similar proof.

Theorem 8 (Positive Introspection)**.** *For a formula φ knowable in a situation term σ:* $Knows(\varphi, \sigma) \rightarrow Knows(Knows(\varphi, \sigma), \sigma)$

Theorem 9 (Negative Introspection)**.** *For a formula φ knowable in a situation term σ:* $\neg Knows(\varphi, \sigma) \rightarrow Knows(\neg Knows(\varphi, \sigma), \sigma)$

Proof. We begin by expanding the definition of positive introspection. The antecedent $Knows(\varphi, \sigma)$ expands by Definition 11 to:

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle\sigma\rangle$$

By a recursive application of same definition the consequent expands to:

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\forall s'.K(\sigma\langle s\rangle\langle s'\rangle, \sigma\langle s\rangle) \rightarrow \varphi\langle s\rangle\langle s'\rangle))$$

Thus the complete formula expands to:

$$[\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle s\rangle)] \rightarrow$$
$$[\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\forall s'.K(\sigma\langle s'\rangle, \sigma\langle s\rangle) \rightarrow \varphi\langle s'\rangle)]$$

Negative introspection would have a similar form. From this we can see that to prove that our agent has complete introspection, it is sufficient to prove that the epistemic fluent K is an equivalence relation (reflexive, transitive, and symmetric). The proof is by induction on the *do* function, with the induction hypothesis being that the K fluent imposes an equivalence relation on the initial situations. It is for this purpose that we have Axiom 10. Thus we need only examine the successor state axiom for K (Axiom 7). As we only consider actions which are possible we will restate the axiom as follows:

$$K(s'', do(a, s)) \equiv \exists s'.K(s', s) \wedge s'' = do(a, s') \wedge sr(a, s) = sr(a, s')$$

First we prove reflexivity by examining the case where $s' = s$. We make this substitution into the axiom to produce:

$$K(s'', do(a, s)) \equiv K(s, s) \wedge s'' = do(a, s) \wedge sr(a, s) = sr(a, s)$$
$$K(do(a, s), do(a, s)) \equiv K(s, s) \wedge sr(a, s) = sr(a, s)$$
$$K(do(a, s), do(a, s)) \equiv K(s, s)$$

Which reads that a situation's successor is related to itself exactly when the situation is related to itself, which is the desired result.

To prove symmetry, let s' and s be distinct situations for which $\neg K(s', s)$. It is immediate from the successor state axiom for K that we have $\neg K(do(a, s'), do(a, s))$. By symmetry we also have $\neg K(s, s')$ and thus $\neg K(do(a, s), do(a, s'))$.

Now consider distinct situations s, s' such that $K(s', s)$. Given this, Axiom 7 reduces to:

$$K(do(a, s'), do(a, s)) \equiv sr(a, s) = sr(a, s').$$

By symmetry we have $K(s, s')$ and thus:

$$K(do(a, s), do(a, s')) \equiv sr(a, s') = sr(a, s).$$

Which finally, by symmetry of $=$, gives us:

$$K(do(a, s'), do(a, s)) \equiv K(do(a, s), do(a, s'))$$

As desired.

Finally, to prove transitivity let s, s', s'' be situations such that $K(s', s), K(s'', s')$, and $K(s'', s)$ all hold, satisfying transitivity. From Axiom 7 we have both of the following:

$$K(do(a, s'), do(a, s)) \equiv sr(a, s) = sr(a, s')$$
$$K(do(a, s''), do(a, s')) \equiv sr(a, s') = sr(a, s'')$$

We are concerned only with the case in which both of these K-relationships hold, as we must then show that $K(do(a, s''), do(a, s))$. We see though that by transitivity of $=$, we would have $sr(a, s) = sr(a, s'')$, and thus

$$K(do(a, s''), do(a, s))$$

□

5.2 Default Persistence of Ignorance

We wish to show that if the agent does not know some fluent F to be true, then it will only learn of the truth of F if the action affects F or is a sensing action which reveals information about F. To this end we present the following theorem. Recall that $\mathcal{D}$ is the basic action theory.

Theorem 10. *For an action a, fluent F, and situation σ.*

$\{\mathcal{D}, \neg Knows(F(\sigma), \sigma), \forall s.F(s) \equiv F(do(a, s)),$
$\quad \forall x.\neg Knows((Poss(a, \sigma) \wedge sr(a, \sigma) = x) \rightarrow F(\sigma), \sigma)\}$
$\models \neg Knows(F(do(a, \sigma)), do(a, \sigma))$

This is identical to Theorem 2 in Scherl and Levesque[14], except we allow σ to be an arbitrary situation term while Scherl and Levesque required it be an individual situation variable s.

The second premise states that the action a will never change the value of fluent F. The third states that the agent does not know in σ that if the sensor result of action a were some value then F would be true, meaning that a will not reveal any information about F (i.e. it is not a sensing action for F). The theorem states that in this case the agent will not know the value of F after performing action a.

Proof. The third premise is an abbreviation for

$$\forall x \exists s.K(\sigma\langle s\rangle, \sigma) \wedge Poss(a, \sigma\langle s\rangle) \wedge sr(a, \sigma\langle s\rangle) = x \wedge \neg F(\sigma\langle s\rangle)$$

Let $x = sr(a, \sigma\langle s\rangle)$ to obtain:

$$\exists s.K(\sigma\langle s\rangle, \sigma) \wedge Poss(a, \sigma\langle s\rangle) \wedge \neg F(\sigma\langle s\rangle)$$

Given this, by the successor state axiom for K we have $K(do(a, \sigma\langle s\rangle), do(a, \sigma))$. From this and the second premise we have $\neg F(do(a, \sigma\langle s\rangle))$.

Thus $\exists s'.K(do(a, \sigma\langle s'\rangle), do(a, \sigma\langle s\rangle)) \wedge \neg F(\sigma\langle s'\rangle)$ therefore

$$\neg Knows(F(do(a, \sigma)), do(a, \sigma))$$

□

5.3 Knowledge Incorporation

Certainly we wish for the agent to know the consequences of its knowledge. Here we show that if P is true whenever F is true (and the agent knows this), then the agent will learn P after sensing F.

Theorem 11. *Let a be an action, F and P (possibly negated) fluents, and σ a situation term:*

$\{\mathcal{D}, \exists x.Knows(F(\sigma) \equiv sr(a,\sigma) = x, \sigma),$
$\quad F(\sigma), Poss(a,\sigma),$
$\quad Knows(F(\sigma) \rightarrow P(\sigma), \sigma)\}$
$\models Knows(P(do(a,\sigma)), do(a,\sigma)) \wedge Knows(F(\sigma) \rightarrow Knows(P(do(a,\sigma)), do(a,\sigma)), \sigma)$

This is an extension of Theorem 3 in Scherl and Levesque[14], which is identical except it derives only the first conjunct in the conclusion. We introduce conditional prospection on P. Also we allow an arbitrary situation term σ while Scherl and Levesque require a single situation variable s.

The first premise states that a is a sensing action for the fluent F. Additionally we have that F is satisfied in σ (though the agent may not know it), that a is possible in σ, and that the agent knows in σ that if F is satisfied then so is P. The theorem states that given these conditions the agent will learn P as a result of action a. Moreso the agent knows prior to action a that, conditional on F being true, it will know after performing a that P is then satisfied.

Proof. The first premise is an abbreviation for

$$\exists x \forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (F(\sigma) \equiv sr(a,\sigma) = x)\langle s\rangle$$

For any situation term τ there exists an atomic situation term s such that $\tau\langle s\rangle = \tau$. If we let s be the term such that $\sigma\langle s\rangle = \sigma$ then we have

$$\exists x.K(\sigma,\sigma) \rightarrow (F(\sigma) \equiv sr(a,\sigma) = x)$$

Recall from the proof of Theorem 8 that if K is reflexive for initial situations, then it will always be reflexive. Axiom 10 guarantees this, thus the antecedent of this formula holds. Also it is a premise that $F(\sigma)$. Thus $sr(a,\sigma) = x$. Given $Poss(a,\sigma)$ and Theorem 1, the successor state axiom for K yields that

$$\forall s.F(do(a,\sigma\langle s\rangle)) \equiv F(\sigma\langle s\rangle)$$

Thus for every $do(a, \sigma\langle s\rangle)$ such that $K(do(a, \sigma\langle s\rangle), do(a, \sigma))$ holds, $F(do(a, \sigma\langle s\rangle))$, $F(\sigma\langle s\rangle)$ and $K(\sigma\langle s\rangle, \sigma)$ also hold. I.e. every situation K-related to $do(a, \sigma)$ satisfies F and by the final premise also satisfies P, thus

$$Knows(P(do(a, \sigma)), do(a, \sigma))$$

Given this and Corollary 3 we also have

$$Knows(F(\sigma) \rightarrow P(do(a, \sigma)), \sigma)$$

□

5.4 Memory

This theorem shows that an agent will never forget a fact as the result of an action, unless that action's effect is to make the negation of that fact true.

Theorem 12. *For a fluent F and situation σ*

$\{\mathcal{D}, Knows(F(\sigma), \sigma),$
$\quad \forall s.F(s) \equiv F(do(a, s))\}$
$\models Knows(Knows(F(do(a, \sigma)), do(a, \sigma)), \sigma)$

This is a more general form of Theorem 4 in Scherl and Levesque[14], which from the same premises concluded $Knows(F(do(a, s)), do(a, s))$. Here we have an arbitrary situation term rather than the single variable s. What's more this formula follows trivially from the conclusion of our theorem, which states that not only will the agent still know F after performing a in σ, it knows before performing a that it will still know F afterwards.

Proof. The second premise expands to

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow F(\sigma\langle s\rangle)$$

This combined with the final premise yields

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow F(do(a, \sigma\langle s\rangle))$$

Which allows an application of Corollary 1 to show

$$Knows(Knows(F(do(a, \sigma)), do(a, \sigma)), \sigma)$$

□

5.5 Knowledge of Effects of Actions

Finally we demonstrate that the agent knows the effects of its actions.

Theorem 13. *Let a be an action, φ a formula knowable in situation term σ, and F a fluent:*

$\{\mathcal{D}, \neg IsSensingAct(a), \forall s.\varphi\langle s\rangle \rightarrow F(do(a, \sigma\langle s\rangle))\}$
$\models Knows(Poss(a, \sigma), \sigma) \rightarrow$
$\quad (Knows(\varphi \rightarrow F(do(a, \sigma)), \sigma) \wedge$
$\quad\quad Knows(Knows(\varphi, \sigma) \rightarrow Knows(F(do(a, \sigma)), do(a, \sigma)), \sigma))$

This is an expanded version of Theorem 5 in Scherl and Levesque[14] which required a single situation variable instead of the arbitrary situation term σ, and derived only the following conclusion:

$$Knows(Poss(a) \wedge \varphi, s) \rightarrow Knows(F(do(a, s)), do(a, s))$$

Which states that if the agent knows a is possible and that φ is true, then it will learn F as a result of a. We show in addition to this that the agent knows, conditional on φ, that F will be true after a. What's more the agent knows that it will learn F after performing a if φ is *certainly* true in σ (i.e. the agent knows φ).

Proof. From the second premise it trivially follows that

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (\varphi\langle s\rangle \rightarrow F(do(a, \sigma\langle s\rangle)))$$

Given this and assuming the action a is actually possible, Corollary 3 gives the first part of the conclusion:

$$Knows(\varphi \rightarrow F(do(a, \sigma)), \sigma)$$

Additionally, if the agent knows φ in σ, then with the second premise we have

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow F(do(a, \sigma\langle s\rangle))$$

This says that F is satisfied in every situation resulting from performing a in a situation K-related to σ, and therefore F is known in those situations:

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow (Knows(\varphi, \sigma) \rightarrow Knows(F(do(a, \sigma\langle s\rangle)), do(a, \sigma)))$$

This is equivalent by Corollary 3 to

$$Knows(Knows(\varphi, \sigma) \rightarrow Knows(F(do(a, \sigma)), do(a, \sigma)), \sigma)$$

which is the second part of the conclusion. Conjunction gives the theorem. □

5.5.1 Sensing Actions

A particularly interesting special case is that of sensing actions. Consider a similar case to the above theorem, but instead with a sensing action. Then following theorem would apply:

Theorem 14. *Let a be an action, and φ, ψ formulae knowable in situation term σ*

$$\begin{aligned}
&\{\mathcal{D}, IsSensingAct(a)\\
&\quad \forall s.\varphi\langle s\rangle \rightarrow sr(a, \sigma\langle s\rangle) = R\\
&\quad \forall s.\psi\langle s\rangle \rightarrow sr(a, \sigma\langle s\rangle) = R',\\
&\quad R \neq R',\\
&\quad \forall s.Poss(a, \sigma) \wedge K(\sigma\langle s\rangle, \sigma) \rightarrow Poss(a, \sigma\langle s\rangle),\\
&\quad \forall s.(\varphi \vee \psi)\langle s\rangle\}\\
&\models Knows(Knows(\varphi, do(a, \sigma)) \vee Knows(\psi, do(a, \sigma)), \sigma)
\end{aligned}$$

The premises state that a is a sensing action which has sensing result R in situations which satisfy φ, and sensing result R' in those which satisfy ψ. Additionally they state that any situation satisfies at least one of φ and ψ, and that if a is possible it is known to be possible. The theorem states that in these circumstances the agent will know that the effect of action a is that it will learn either φ or ψ. This is the general definition of the scenario proposed by Shapiro et. al which was previously discussed in section 3.1.

Proof. Given the second through fourth premises and that sr is a well-defined function, we have that $(\varphi \wedge \psi)\langle s\rangle \models \bot$. Keeping this in mind consider the trivial case where the agent already knows φ; i.e. $\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow \varphi\langle s\rangle$. Thus we have

$$\forall s.K(\sigma\langle s\rangle, \sigma) \rightarrow sr(a, \sigma\langle s\rangle) = sr(a, \sigma) = R$$

From this, Theorem 1, and Axiom 7, it follows that a will have no effect. A similar argument applies if the agent already knows ψ.

If the agent does not know φ or ψ, then we have

$$\exists s.K(\sigma\langle s\rangle, \sigma) \wedge \sigma\langle s\rangle \neq \sigma \wedge (\varphi \not\equiv \varphi\langle s\rangle)$$

As such we have $sr(a, \sigma) \neq sr(a, \sigma\langle s\rangle)$ and thus by Axiom 7

$$\neg K(do(a, \sigma\langle s\rangle), do(a, \sigma))$$

Indeed we have that all K-related successor situations share the same sensor value, and thus the same value of φ:

$$\forall s.K(do(a, \sigma\langle s\rangle), do(a, \sigma)) \equiv (K(\sigma\langle s\rangle, \sigma) \wedge \varphi \equiv \varphi\langle s\rangle)$$

And thus by Theorem 1

$$\forall s.K(do(a, \sigma\langle s\rangle), do(a, \sigma)) \equiv (\varphi\langle do(a, \sigma)\rangle \equiv \varphi\langle do(a, \sigma\langle s\rangle)\rangle)$$

Which by Definition 11 gives either $Knows(\varphi, do(a, \sigma))$ or $Knows(\neg\varphi, do(a, \sigma))$ (and therefore $KWhether(\varphi, do(a, \sigma))$). An analogous argument applies for ψ, which considering the final premise gives

$$Knows(\varphi, do(a, \sigma)) \vee Knows(\psi, do(a, \sigma))$$

This formula is knowable in σ and is satisfied in every situation related to σ, thus by Theorem 5 we have our theorem:

$$Knows(Knows(\varphi, do(a, \sigma)) \vee Knows(\psi, do(a, \sigma)), \sigma)$$

□

Chapter 6

Conclusions

This work is an extension on the method of knowledge representation in the situation calculus described by Scherl and Levesque in [14]. We have introduced an expanded definition of knowledge which can express what the agent knows about one situation while in another. Particularly we show that the agent can have knowledge of the possible effects of an action, including which of those results are mutually exclusive, before the action is performed. Results about the agent's own future knowledge are not excluded; the agent may know what it could possibly learn from a sensing action. This additional functionality does not come at a sacrifice of any functionality or expressivity from Scherl and Levesque's method.

6.1 Related Work

In the case of actions with multiple possible outcomes consistent with the agent's knowledge (including the above case of sensing actions), the agent considers each of those outcomes as distinct possible cases, but does not know which will result until the action is performed. A similar approach is taken by Mateus et al.[7] — and indeed they solve the problem of modelling actions with multiple outcomes much more robustly — but they do so by extending the situation calculus with significantly more new machinery than this thesis to achieve that result. Further the behaviour of prospection is the primary contribution of this thesis and is not discussed by Mateus et al.[7].

Pinto and Reiter[11] also discuss time in the situation calculus, allowing the agent to reason about events[1] occurring at absolute times, effectively introducing a clock to the situation calculus. This allows the expression of concepts such as "before", "after", and "during",

[1] Pinto and Reiter do not distinguish between events and actions in [11].

and the inference of the occurrence of exogenous events[2] which the agent did not directly observe. The temporal logic TL[4] was incorporated into the situation calculus for this end. However knowledge is not discussed at all, let alone prospection.

Finally Lin and Reiter[6] demonstrate "progression" of a situation calculus action theory as a method to decide whether a formula is satisfied in a future situation. However they do not explicitly discuss knowledge or non-uniform formulae. De Giacomo et al.[2] expand on progression to be able to express (and verify) statements such as "in every situation of every execution path beginning at this situation the fluent F is satisfied". Similar to Pinto and Reiter's[11] use of TL, De Giacomo et al. use notation like that in the temporal logic CTL*[1] to express these statements. To perform this progression they also present a variant of the μ-calculus[3, 16]. However De Giacomo et al.[2] does not discuss prospection explicitly, and while CTL* and the μ-calculus are powerful and well established, our approach has the benefit of simplicity; it does not need any of this additional machinery.

6.2 Future Work

Reiter[13] introduces the notion of regression, which allows the truth of a formula in an arbitrary situation to be determined by generating a logically equivalent formula which is relative to the initial situation. Similarly De Giacomo et al.[2] discuss progression of action theories. The relationships between regression, progression, and prospection have yet to be explored.

This paper expands the class of formulae of which the agent can have knowledge using a representation like that in Scherl and Levesque[14] from the uniform formulae to the knowable formulae (as defined in this thesis). Whether this approach is more or less expressive than the μ-calculus variant introduced in [2] is currently unknown.

Shapiro et al. present a method of "Iterated Belief Revision in the Situation Calculus"[15]. This permits a more complete theory of knowing by allowing for changing of potentially mistaken beliefs. Indeed it is this work which first discussed the problem of prospection allowing an agent to know the results of a sensing action before performing it. Previous drafts of this thesis attempted to instead extend Shapiro et al.'s[15] approach with correct introspection of future beliefs, though this proved to be an endeavour of unsuitable scale. It is believed by the author of this thesis that this type of belief is not expressible with axioms in only first-order logic. Further attempts may be made in the future.

Finally we include Axiom 5 not because it is strictly necessary, but because there appears to be no use to modelling situations prior to the initial situation as they are irrelevant to

[2]Events which were not caused by the agent.

the agent's actions. However in reality we are often concerned with events which transpired before we were born despite the fact that we cannot affect them. It may be interesting to interpret S_0 as the situation in which the agent is "born", and when (if ever) it is useful to model situations prior to S_0.

Bibliography

[1] C. Baier and J.P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[2] Giuseppe De Giacomo, Yves Lespérance, Fabio Patrizi, and Stavros Vassos. Progression and verification of situation calculus agents with bounded beliefs. *Studia Logica*, 104(4):705–739, 2016.

[3] E. Allen Emerson. Model checking and the mu-calculus. In *DIMACS Series in Discrete Mathematics*, pages 185–214. American Mathematical Society, 1997.

[4] R. Goldblatt. *Logics of Time and Computation*. CSLI, 1987.

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

[6] Fangzhen Lin and Ray Reiter. How to progress a database. *Artificial Intelligence*, 92(1):131 – 167, 1997.

[7] Paulo Mateus, António Pacheco, Javier Pinto, Amílcar Sernadas, and Cristina Sernadas. Probabilistic situation calculus. *Annals of Mathematics and Artificial Intelligence*, 32(1):393–431, 2001.

[8] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

[9] R.C. Moore. Reasoning about knowledge and action. Technical Note 191, SRI International, October 1980.

[10] R.C. Moore. A formal theory of knowledge and action. In J.R. Hobbs R.C. Moore, editor, *Formal Theories of the Commonsense World*, pages 319–358. Ablex, Norwood, NJ, 1985.

[11] Javier Pinto and Raymond Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2):251–268, 1995.

[12] R. Reiter. Proving properties of states in the situation calculus. *Artificial Intelligence*, 64:337–351, 1993.

[13] Raymond 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, San Diego, CA, 1991.

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

[15] S. Shapiro, M. Pagnucco, Y. Lesperance, and H.J. Levesque. Iterated belief change in the situation calculus. *Artificial Intelligence*, 2009.

[16] Colin Stirling. *Modal and Temporal Properties of Processes*. Springer-Verlag New York, Inc., New York, NY, USA, 2001.