Two Aspects of Automating Logics of Action and Change — Regression and Tractability

Marcus Bjäreland

February 6, 1998

acknowledgements

The following people have contributed to this thesis with support, help, comments, or patience:
Patrick Doherty, Lars Karlsson, Thomas Drakengren, Joakim Gustafsson, Jonas Kvarnström, Patrik Haslum, Silvia Coradeschi, Peter Jonsson, Christer Bäckström, Simin Nadjm-Tehrani, my parents, and my girlfriend Annika Strandman.
I am grateful to you all.

Marcus Bjäreland, Linköping, January 1998

abstract

The autonomy of an artificial agent (e.g. a robot) will certainly depend on its ability to perform "intelligent" tasks, such as learning, planning, and reasoning about its own actions and their effects on the enviroment, for example predicting the consequences of its own behaviour. To be able to perform these (and many more) tasks, the agent will have to represent its knowledge about the world.

The research field "Logics of Action Change" is concerned with the modelling of agents and dynamical, changing environments with logics.

In this thesis we study two aspects of *automation* of logics of action and change. The first aspect, regression, is used to "reason backwards", i.e. to start with the last time point in a description of a course of events, and moving backwards through these events, taking the effects of all actions into consideration. We discuss the consequences for regression of introducing nondeterministic actions, and provide the logic PMON with pre- and postdiction procedures. We employ the classical computer science tool, the *weakest liberal precondition* operator (*wlp*) for this, and show that logical entailment of PMON is equivalent to *wlp* computations.

The second aspect is computational complexity of logics of action and change, which has virtually been neglected by the research community. We present a new and expressive logic, capable of expressing continuous time, nondeterministic actions, concurrency, and memory of actions. We show that satisfiability of a theory in this logic is NP-complete. Furthermore, we identify a tractable subset of the logic, and provide a sound, complete, and polynomial algorithm for satisfiability of the subset.

Contents

Chapter 1

Introduction

One of the major goals of *Artificial Intelligence* (AI) has, since the emergence of the field in the late 50s, been to construct agents (e.g. robots) that are able to observe and act in dynamically changing environments. The autonomy of such a device will certainly depend on its ability to perform "intelligent" tasks, such as learning, planning, and reasoning about itself and the environment, and, for example, to predict the consequences of its actions.

In this thesis we are interested in logical representations of the dynamic behaviour of agents interacting with an environment. This is studied in the area *Logics of Action and Change*, which is a subarea of *Reasoning about Action and Change* (RAC). In particular we are concerned with how such logics can be implemented in efficient ways.

The first work on "logical" AI was probably generated by John McCarthy [1958]. He presented the idea of a program, the ADVICE TAKER, that would manipulate sentences in formal languages. It is clear that he aimed at a learning system, a system that would be able to get smarter and smarter to be able to perform more and more complex tasks in the real world. However, McCarthy bumped into a major problem: How do we represent the knowledge gained in the system?

Since then an abundance of philosophies, formalisms, and systems has been developed in relation to this question. Many of them can be proved to have the same expressivity (e.g. Turing equivalence), while they actually make the users look at the world they want to model in different ways.

In this chapter we will present philosophical frameworks for knowledge representations, and pose a number of questions that are relevant for representation designers. We will also discuss the research methodologies em-

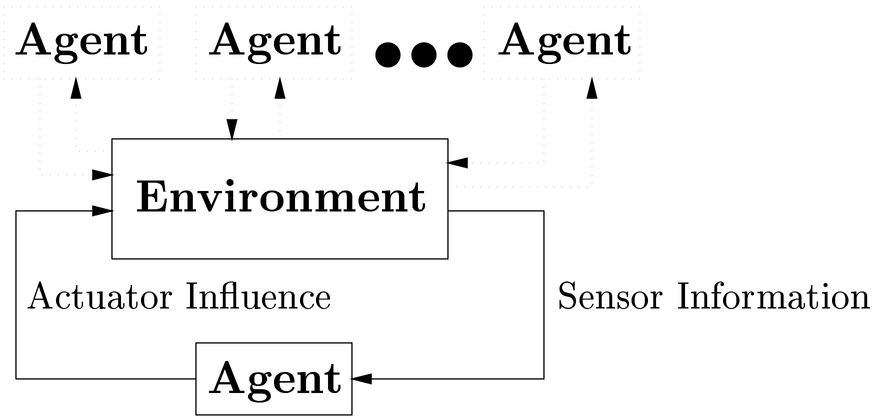

Figure 1.1: Relations between agents and the environment.

ployed by researchers working with logics of action and change.

1.1 Knowledge Representation

Figure 1.1 shows a generic view of multiple agents sensing and interacting with an environment. The figure could equally well explain agent-environment relations for a mobile robot as for a "softbot", that is, a software agent acting in a computer system.

The agents get information about the currents status of the environment through their sensors, for example, visual information through cameras, or by reading some output from an operating system. The sensor information is then processed by the agent that decides what to do next. The agent realizes its decision by performing some action in the environment, for example by moving forward or by compressing a file. In this thesis, we are interested in logical models of Figure 1.1. However, there are a number of feasible ways of how such a system can be modelled, and it is important to make the purpose of the model clear. We distinguish between the following three levels of representation (see Figure 1.2):

1. Representation of the whole system with agents, the environment, and interactions between the agents and the evironment.

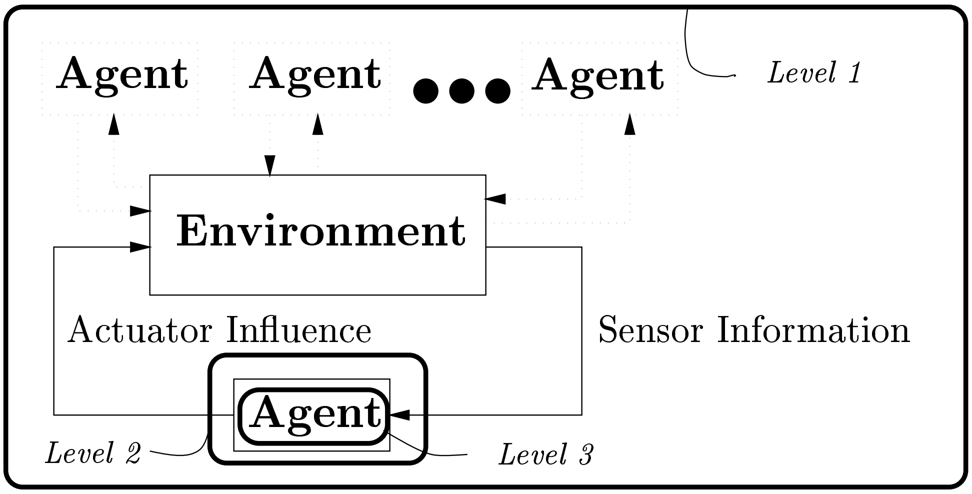

Figure 1.2: The three levels of representation.

2. Representations of one agent and its interactions with other agents and the environment.

3. Representations implemented in and used by one agent for decision making.

Israel [1994] focuses on two broad functions that *logics* (not representations in general) can have in representations:

- as a source of languages and logics for artificial reasoners;

- as a source of analytical tools and techniques, such as providing the underlying mathematical and conceptual framework within which much AI research is done.

Israel's second function is not applicable to the last level of representation, other than in a datatype perspective. Otherwise, the levels and the roles are orthogonal. It could, for example, be of interest to implement a system that reasons about the complete system in Figure 1.1, even though it is more common to see such representations used for analytical purposes.

The RAC community has to a high degree been interested in using logics as in the second function, and it is clear that Israel thinks that the second function motivates the use of logics more than the first.

1.1.1 Three Criteria of Adequacy

In their seminal paper [1969], McCarthy and Hayes introduced the following three criteria of adequacy for representations of the world:

- **Metaphysical Adequacy** – for representations whose form is consistent with the aspect of the world we are interested in.
- **Epistemological Adequacy** – for representations that can be practically used by a person or a machine to express the facts one actually has about the aspect of the world.
- **Heuristical Adequacy** – for representations whose language is capable of expressing the reasoning processes actually gone through in solving a problem.

McCarthy and Hayes are aiming at representations that are intended to be used in agents for real-world applications. Thus, it is the second or third level of representation and Israel's first function that primarily interests them.

Heuristical adequacy is not discussed in any detail in McCarthy and Hayes' paper (they admit that this concept is "tentatively proposed"), and the concept might appear somewhat confusing. A plausible explanation could be that McCarthy and Hayes describe "layered" representations, i.e. representations in which it is possible to reason about, and change, reasoning mechanisms on lower layers. For humans, it is not strange to be able to make a plan and then discuss the process of making the plan, and so on. We will, however, not use this criterion further in this thesis.

1.1.2 Five Roles of a Knowledge Representation

Another conceptual framework for knowledge representation, was presented by Davis *et al.* [1993] where five distinct roles of a particular representation are identified. Thus, a knowledge representation can be seen as a

- a **surrogate**, a substitute for the thing itself, enabling an entity to determine consequences by thinking rather than acting,

- a **set of ontological commitments**, an answer to the question: In what terms should I think about the world?
- a **fragmentary theory of intelligent reasoning**, expressed in terms of three components:
    1. the representation's fundamental conception of intelligent reasoning,
    2. the set of inferences[1] *allowed*[2] by the representation,
    3. the set of inferences *intended*[3] by the representation,
- a **medium for pragmatically efficient computation**, and
- a **medium of human expression**, a language in which we can say things about the world.

It is obvious that that every knowledge representation is a surrogate for something else. This view leads to at least two questions. First, we must ask what it is a surrogate for, and what the conceptual correspondences between the reality and the representation are. The second question is how close the surrogate is to the real thing. Obviously a representation is inaccurate, but in what way, and how much? Note that the surrogate view can serve equally well for abstract, philosophical objects, like causality or beliefs, as for concrete objects like chairs and ventilation ducts. It is important to note that depending on the level of representation we are interested in, we have different views on the surrogacy of the representation. For the first level, where we we represent the entire system, we might focus more on the interaction between the agents than on the internal representation of a single agent. This means that we need an accurate and fine-grained representation of the environment (where the interactions take place). If, on the other hand, we would be on the second level, we would be more interested in the accuracy of representation of the particular agent we are interested in.

In selecting any representation we are in the same act unavoidably making a set of decisions about how and what to see in the world. That is, selecting a representation means making a set of ontological commitments.

[1]The word "inference" is here used in a generic sense, i.e. it is the way in which a representation may yield new expressions from old. We are not confined to deductive inference.

[2]In their paper, the term *sanctioned* is used.

[3]In their paper, the term *recommended* is used.

The commitments may accumulate in layers since, if we start with a KR for describing some basic notion of the world we make a commitment, and if we then represent composition of the basic notions, we might make a new commitment, but on a higher level. It is desirable that our commitments are metaphysically adequate. For example, a rule-based Knowledge Representation (KR) and semantic networks share a number of features and can be combined, but they force a user to emphasize and filter away different aspects of reality.

The third role concerns the inferences that can and should be made from a representation. The three components implicitly define what it means to reason intelligently, what can be inferred, and what ought to be inferred, from the representation.

Since the ultimate goal of any work in KR is to implement the representations and use them in systems, it of great interest to study computational properties of the representations.

The final role deals with how comfortable humans are in using a particular representation for expressing things about the world. This is important if we have to instruct our machines (and other people) about the world, or if we want to use the representation to analyze aspects of the world.

1.1.3 A Comparison and Discussion of the Three Frameworks

It is beyond the scope of this thesis to completely relate the three frameworks to each other. However, a number of points can be made.

The frameworks can be viewed as providing a number of highly relevant questions that any representation designer should answer. In Chapters 2 and 3, we will use these questions to analyze the some approaches to RAC.

1. What will I use the representation for? Analysis or implementation? Both?

2. What level of representation do I require for the chosen use? Agent, environment, and other agents? Agent and its interaction with the environment? The internal state of the agent?

3. What aspect of the world am I interested in?

4. What structure or part of the salient aspect do I want to emphasize, and what can I filter out, that is, what ontological commitments am I making?

5. Are my ontological commitments consistent with the real world?

6. How close to the real world is my representation of it?

7. Does the representation come with a mechanism to draw non-trivial conclusions about a represented fragment of the world? Does it yield "intelligent" conclusions?

8. Are the computational properties of the inference mechanism pragmatically useful?

9. Who will use the representation, a person or a machine?

10. Is it possible for the user to express facts about the salient aspect easily in the formalism?

Question (1) comes from Israel's framework, questions (5) and (10) from McCarthy and Hayes, and questions (4), (6), (7), (8), and (10) from Davis *et al.*

The questions above create a basis for an analysis from which further investigations have to be made. For example, question (3) is likely to have to be more specific, especially if the aspect is broad and/or philosophical. If we want to model causality, we need to specify what we mean by the concept, otherwise it is impossible to assess how close the representation is to what we are after.

1.2 Logics of Action and Change

The results presented in this thesis are concerned with one particular branch of Knowledge Representation: Reasoning about Action and Change, or more specifically "Logics of Action and Change". Typically, this area is described as "formal studies in representations of dynamical systems" or "modeling of common sense notions of action and change" (see for instance [Sandewall and Shoham, 1994]). Clearly, RAC is central to KR; it would not suffice to only have a representation of the statics of the world in an agent, the dynamics have to be present, too.

Before we examine particular logical approaches to RAC in Chapters 2 and 3, we can – from the viewpoint of the questions above – ask what the choice of logic does in terms of possible aspects of the world, ontological commitments, reasoning mechanisms, and so on?[4]

- Logics have been thoroughly studied historically, and provide a natural medium of expression.
- Logics do not make heavy ontological commitments for *static* representations of the world. For example, first-order logic assumes that we view the world in terms of objects and relations between objects. However, though this does not change for attempts to describe the *dynamics* of the world, in this case we will have to be somewhat more precise, as we will see below.
- Logics provide a good mechanism for intelligent reasoning: Logical Consequence. As discussed by McDermott [1987], this is not the only way to perform intelligent reasoning, but we claim that it is a good place to start.
- First-order logic is hard to compute with; it is only semi-decidable. To achieve pragmatically efficient computational mechanisms we will have to rely on subsets, approximations, or heuristics. This also implies that logics generally are highly expressive computationally. For example, first-order logic is Turing equivalent.

1.2.1 The Frame Problem

John McCarthy's role in logical AI cannot be overestimated. The research track that he founded in [McCarthy, 1958], and which he and Pat Hayes developed in [McCarthy and Hayes, 1969], has had many followers. Their ultimate goal, however, to make intelligent programs, has been somewhat shadowed by one of the hardest problems in AI — The Frame Problem.

There have been a number of interpretations and definitions of the Frame Problem since it was introduced by McCarthy and Hayes [1969]. We do not intend to get involved in the debate, and since we are only interested in a subproblem of the general problem (the *persistence problem*), we hope that the following definition of the frame problem is uncontroversial:

> How do we represent aspects of the world that change, and aspects that do not change in a succinct manner?

[4] We will, for now, use a generic notion of logic. A logic is a formal system with well-defined syntax and semantics.

It is fairly easy to describe a static scenario in any formalism. For example, I could describe my office in geometric terms ("A photo with dimensions $1 \times 40 \times 60$ hangs on the northern wall with coordinates ...") or in qualitative terms ("The computer is on top of the desk, which is to the right of ..."). If I were to move the coffee cup 2 cm to the left on my desk, the resulting room in the world would be equally easy to describe. But it would be very inefficient to exhaustively describe the complete room every time something in changed. I would like to represent the change, and assume that everything else remained unchanged (persisted). Basically, solutions to the Frame Problem are ways of fulfilling that assumption.

In the framework of Davis *et al.*, it is clear that the two goals of RAC – to make intelligent programs, and to solve the Frame Problem – emphasize different roles of representations.

In the formulation above, the Frame Problem can be related to the role of finding pragmatically efficient computational mechanisms for the representations (since we, for example, do not want to fill our computers' memories with axioms saying what does not change when an action is executed), and to the role of good medium for human expression (somehow, humans are able to make good abstractions).

The goal of writing intelligent programs shares the emphasis on pragmatically efficient computation and focuses on this, but also emphasizes good surrogacy and good fragmentary theories of intelligent reasoning.

1.2.2 The Logics in this Thesis

In Chapters 3 and 5 we present two logics for action and change. Both of them can (and, perhaps, should) be used for descriptions on the second level of representation; that is, we are interested in the representation and reasoning processes of a single agent, and we are not concerned with a fine-grained description of the environment.

Both logics are based on *narratives*, i.e, partial specifications of an initial and other states of the system, combined with descriptions of actions, in terms of their preconditions, effects, and timing. An example (which is formalized in two different formalisms in Chapter 2 and 3 respectively) is the *Pin Dropping Scenario*:

OBS1	`At time point 0 the pin is held over the board, and it is not on a black or a white square.`
ACT	`If the pin is dropped between time points` s `and` t `then, if the pin is held over the board at` s`, it will be on a white or a black square, or both, at` t `and it will not be held over the board any longer at` t`.`
SCD	`The pin is dropped between time points 3 and 5`
OBS2	`At time point 6 the pin is on a white but not on a black square.`

An alternative to narrative-based formalisms is presented in Sections 2.1.1 and 2.1.2.

For the logics in Chapters 3 and 5 we assume that the agent of interest is the only agent in the environment, and that it has complete knowledge about when events occur and what effects the events have on the environment (*action omniscience*). Moreover, we assume that nothing changes in the environment unless the agent has performed some action (*action-based inertia*). The version of PMON in Chapter 3 represents the flow of time with natural numbers $(0, 1, \ldots)$, while in Chapter 5 time is continuous. For PMON we also restrict the scenarios so that actions cannot overlap in time.

1.3 Topics of this Thesis

This thesis consist of two themes with one issue in common: computational theories for logics of action and change.

1.3.1 Regression, Nondeterminism, and Computational Mechanisms for PMON

One of the most popular reasoning mechanisms in Reasoning about Action and Change is *regression*, where the basic idea is to move backwards through a course of events, taking the effects of all occurring actions into consideration, to achieve a description of the entire scenario at the initial time point. Its popularity comes from the intuitive appeal of the mechanism, and the fact that complexity is reduced by removing the temporal component (by *projection*, in the mathematical sense of the word).

In Chapter 4 we define a classical computer science tool, Dijkstra's *weakest liberal precondition* operator for PMON (which is properly introduced in Chapter 3). We analyze the consequences for regression when nondeterministic actions are allowed, and show that we must distinguish between sufficient and necessary conditions before an action is executed for the action to end up in a fixed state. This distinction is then used to define a prediction and a postdiction procedure for PMON.

1.3.2 Computational Complexity

An interesting question that has almost been ignored[5] by the Logics of Action and Change community is if it possible to find *provably* efficient computational mechanisms for RAC (as opposed to *pragmatically* efficient computations). Many interesting RAC problems are (at least) NP-hard, and tractable subproblems that are easily extracted tend to lack expressiveness. This has led a large part of the RAC community to rely on heuristics and incomplete systems to solve the problems (see for example [Ginsberg, 1996] for a discussion). It is clear that very expressive logical formalisms provide difficult obstacles when it comes to efficient implementation.

We feel, however, that the tractability boundary for sound and complete reasoning about action has not yet been satisfactorily investigated. We show this in Chapter 5 by introducing a nontrivial subset of a logic with semantics closely related to the *trajectory semantics* of Sandewall [1994], for which satisfiability is tractable. The logic relies solely on the two above-mentioned assumptions of action omniscience and explicit action-based change.

Our logic can handle examples involving not only nondeterminism, but continuous time, concurrency and memory of actions as well, thus providing a conceptual extension of Sandewall's framework.

1.4 Organization

The thesis is organized as follows:

In **Chapter** 2 we present the details of related work. More specifically, we discuss research by Fangzen Lin [1996], and Witold Łukaszewicz and Ewa Madalińska-Bugaj [1995].

Chapter 3 presents the version of PMON used in this thesis.

[5]An exception is Paolo Liberatore's *The Complexity of the Language* $\mathcal{A}$ which is currently being reviewed. It is available at `http://www.ep.liu.se/ea/cis/1997/006/`.

In **Chapter** 4 we adapt *wlp* for PMON, we show that PMON entailment and *wlp* computations coincide, and we show that it is possible to fit both sufficient and necessary conditions before an action for a statement to hold after the execution of the action into a framework of *wlp*. Furthermore, pre- and postdiction procedures for PMON are presented. A larger part of the results have been published in [Bjäreland and Karlsson, 1997].

Chapter 5 includes and extends [Drakengren and Bjäreland, 1997]. A new logic of action and change is introduced, in which continuous time, nondeterministic actions, concurrent actions, and memory of actions are expressible. We show that satisfiability of the logic is NP-complete. A tractable subset, based on results from Temporal Constraint Reasoning, is identified.

In **Chapter** 6 the thesis is summarized and some possible future directions are discussed.

Chapter 2

Related Work

In this chapter we will provide a detailed presentation of two approaches closely related to our approach presented in Chapter 4. The approaches are

- Situation Calculus approaches, with emphasis on Reiter's [1991] and Lin's [1996] work, and
- The approach of Łukaszewicz and Madalińska-Bugaj [1995], where Dijkstra's semantics of programming languages is used.

There are other approaches to RAC than the ones presented here (for example the Event Calculus [Kowalski and Sergot, 1986] and Thielscher's Logic of Dynamic Systems [Thielscher, 1995]). However, since they are not of immediate interest to the results presented in Chapters 4 and 5, we have omitted a presentation of them.

2.1 Situation Calculus Approaches

The situation calculus (SitCalc) is arguably the most wide spread formalism for reasoning about action and change today. Its present form was originally suggested by McCarthy and Hayes [1969], and has been widely studied ever since (see [Sandewall and Shoham, 1994] for an overview). The most sophisticated and developed flavour of SitCalc studied today is, probably, the "Herbrand[1] flavour" where situations are considered to be sequences of actions. Reiter [1991] combined the theories of Pednault [1986] and Schubert [1990] by introducing a suitable closure assumption so that his theory solved

[1]A term used by Sandewall and Shoham [1994]

the frame problem for a larger class of scenarios than before. Reiter also adapted a reasoning mechanism, *Goal Regression*, which was introduced by Waldinger [1977] and further developed by Pednault [1986], to his theory.

There exists a number of extension of the situation calculus (or the related formalism $\mathcal{A}$) for handling nondeterministic actions (for example [Giunchiglia *et al.*, 1997, Baral, 1995]), but Lin [1996] is the only one providing a regression operator for the logic.

In this section we will present Reiter's work, and Lin's extension to it.

2.1.1 Reiter

SitCalc (in this version) is a many-sorted second-order language with equality. We assume the following vocabulary:

- Infinitely many variables of every sort.
- Function symbols of sort *situations*. There are two function symbols of this sort: The constant S_0 denoting the initial situation, and the binary function symbol *do*, which takes arguments of sort *actions* and *situations*, respectively. The term $do(a, s)$ denotes the situation resulting from executing the action a in situation s.
- Finitely many function symbols of sort *actions*.
- Infinitely many function symbols of sort other that *actions* and *situations*. These symbols will be referred to as *fluents*.
- Predicate symbols:
  - A binary predicate *Poss* taking arguments of sorts *actions* and *situations*, respectively. $Poss(a, s)$ denotes that it is possible to execute action a in situation s.
  - A binary predicate h taking arguments of sorts *fluents* and *situations*, respectively. $h(f, s)$ denotes that the fluent f is true (*h*olds) in situation s.
- The usual logical connectives, quantifiers, and punctuations.

For a fluent R we let $R(s)$ mean exactly the same thing as $h(R, s)$, for readability.

For readability we will only consider fluents and actions with arity 0. The theory can easily be generalized. The basic idea of Reiter's approach is that the user provide axioms that state the circumstances under which it is possible to execute actions (Action Precondition Axioms), and axioms that state the circumstances when the value of a fluent may change (General Positive/Negative Effect Axioms). Then, under the assumption that the effect axioms completely describe all ways a fluent may change value, it is possible to generate *successor state axioms*, which characterizes the possible changes of fluent values.

The predicate $Poss(a, s)$ denotes that it is possible to execute action a in situation s, and it is defined as follows[2]:

Action Precondition Axioms (APA)
For each action a

$$Poss(a, s) \equiv \Psi_a(s),$$

where $\Psi_a(s)$ is a formula describing the condition under which it is possible to execute action a in situation s.

For the fluents we state one axiom for circumstances when an action may change its value to **T** and one for **F**. Formally, with each fluent, R, we associate two *general effects axioms*:

General Positive Effect Axiom for Fluent R (PEA_R)

$$Poss(a, s) \wedge \gamma_R^+(a, s) \rightarrow h(R, do(a, s)).$$

General Negative Effect Axiom for Fluent R (NEA_R)

$$Poss(a, s) \wedge \gamma_R^-(a, s) \rightarrow \neg h(R, do(a, s)).$$

The formula $\gamma_R^+(a, s)$ $(\gamma_R^-(a, s))$ characterize the conditions and actions a that make fluent R true (false) in situation $do(a, s)$.

Next, we assume that PEA_R and NEA_R completely characterizes the conditions under which action a can lead to R becoming true (or false) in the successor state. This assumption can be formalized as follows:

[2] In this and the following section, we assume that all free variables are universally quantified

Explanation Closure Axioms (ECA)

$$Poss(a,s) \wedge h(R,s) \wedge \neg h(R, do(a,s)) \rightarrow \gamma_R^-(a,s),$$

$$Poss(a,s) \wedge \neg h(R,s) \wedge h(R, do(a,s)) \rightarrow \gamma_R^+(a,s).$$

These axioms state that if R changes value from **T** to **F** while executing action a, the formula $\gamma_R^-(a,s)$ has to be true, and analogously for when R changes from **F** to **T**.

Reiter shows that with the set of the above-mentioned axioms for all fluents R, $\Gamma = \Sigma \cup \mathrm{PEA}_R \cup \mathrm{NEA}_R \cup \mathrm{ECA}$ we can deduce the following axiom:

Successor State Axiom for Fluent R (SSA_R)

$$Poss(a,s) \rightarrow [h(R, do(a,s)) \equiv \gamma_R^+(a,s) \vee (h(R,s) \wedge \neg\gamma_R^-(a,s))],$$

as long as the formula $\gamma_R^+(a,s) \wedge \gamma_R^-(a,s) \wedge Poss(a,s)$ is not entailed by Γ, that is, as long as all γ for every fluent are mutually exclusive. In fact, under this condition the effect axioms and explanation closure axioms for R are logically equivalent to the successor state axiom.

Successor state axioms play a crucial role in the construction of regression operators. For a specific fluent R, the corresponding successor state axioms specifies exactly what has to hold before an action a is executed, for R to be true (false) after the execution of a.

By substituting fluents in the goal with the right-hand side of the biconditional in the successor state axioms, the nesting of the *do* function can be reduced until there is finally a formula only mentioning the situation S_0, on which a classical atemporal theorem prover can be used. Since Reiter's approach cannot handle nondeterministic actions, we will now turn our attention to Lin's extension to it.

2.1.2 Lin

As an extension to Reiter's SitCalc flavour, Lin introduces a predicate *Caused* which "assigns" truth values to fluents, and a sort, *truth values*, consisting of constant symbols **T** and **F**. The formula $Caused(p,v,s)$ denotes that the fluent p is made to have the truth value v in situation s. There are two axioms for *Caused*:

$$\begin{aligned} Caused(p, \mathbf{T}, s) &\rightarrow h(p, s) \\ Caused(p, \mathbf{F}, s) &\rightarrow \neg h(p, s) \end{aligned}$$

In the minimization policy, the extension of $Caused$ is minimized and a nochange premise is added to the theory. To illustrate his approach Lin uses the dropping-a-pin-on-a-checkerboard example: There are three fluents, $white$ (to denote that the pin is partially or completely within a white square), $black$, and $holding$ (to denote that the pin is not on the checkerboard). There are two actions, $drop$ (the pin is dropped onto the checkerboard) and *pickup*. In this thesis we will only consider the $drop$ action (which is the only one with nondeterministic effects), which is formalized as follows:

$$\begin{aligned} &\forall s. Poss(drop, s) \rightarrow \\ &\quad [Caused(white, true, do(drop, s)) \wedge \\ &\quad Caused(black, true, do(drop, s))] \vee \\ &\quad [Caused(white, false, do(drop, s)) \wedge \\ &\quad Caused(black, true, do(drop, s))] \vee \\ &\quad [Caused(white, true, do(drop, s)) \wedge \\ &\quad Caused(black, false, do(drop, s))]. \end{aligned} \tag{2.1}$$

The $Poss$ predicate is defined, with an action precondition axiom, as

$$\begin{aligned} &\forall s. Poss(drop, s) \equiv \\ &\quad h(holding, s) \wedge \neg h(white, s) \wedge \neg h(black, s) \end{aligned}$$

A problem with this is that the effects of the action have to be explicitly enumerated in the action definition. The number of disjuncts will grow exponentially in the number of fluents, and may become problematic in implementations.

To be able to use goal regression, Lin has to generate successor state axioms. However, this cannot be done in a straightforward way for nondeterministic scenarios. The reason for this is that there are no constraints *before* a nondeterministic action in a biconditional relation on what holds *after* the action has taken effect. Lin deals with this by introducing an "oracle[3]" predicate, $Case(n, a, s)$, which is true iff the nth disjunct of action

[3]Constructs that "know" in advance what the effect of nondeterministic actions will be.

a has an effect (is true) in situation s. Consequently, the *drop* action is defined as follows:

$$\begin{aligned}
&\forall s.Poss(drop, s) \wedge Case(1, drop, s) \rightarrow \\
&\quad Caused(white, true, do(drop, s)) \wedge \\
&\quad Caused(black, true, do(drop, s)), \\
&\forall s.Poss(drop, s) \wedge Case(2, drop, s) \rightarrow \\
&\quad Caused(white, false, do(drop, s)) \wedge \\
&\quad Caused(black, true, do(drop, s)), \\
&\forall s.Poss(drop, s) \wedge Case(3, drop, s) \rightarrow \\
&\quad Caused(white, true, do(drop, s)) \wedge \\
&\quad Caused(black, false, do(drop, s)),
\end{aligned}$$

Lin shows that the new theory is a conservative extension of the previous circumscribed one.

Since the actions now are deterministic, it is possible to generate successor state axioms, for example for *white*:

$$\begin{aligned}
&\forall a, s.Poss(a, s) \rightarrow [h(white, do(a, s)) \equiv \\
&\quad a = drop \wedge (Case(1, drop, s) \vee Case(3, drop, s))]
\end{aligned}$$

To capture that exactly one of the *Case* statements is true, Lin adds the axiom

$$\forall s.Case(1, drop, s) \oplus Case(2, drop, s) \oplus Case(3, drop, s),$$

where $\oplus$ denotes exclusive disjunction[4].

Discussion

Unfortunately, this approach is not as general as it may seem. In fact, goal regression is not possible unless we restrict the problem by disallowing the effects of nondeterministic action to be conditional, that is, The *Poss* predicate makes it possible to express qualifications for the action to be invoked, but it is not possible to express that an invoked action has different effects depending on the the situation it was invoked in.

[4] It is necessary to add more axioms, which Lin does, but they are of no immediate interest to the discussions in this thesis.

The key here to Lin's "oracle" approach is that a previously nondeterministic action is transformed into a deterministic action, where nondeterminism is simulated by the oracles. This means that he transforms a theory with nondeterministic actions to an equivalent theory with deterministic actions and an incompletely specified initial state. Interestingly, Sandewall predicted a similar equivalence when he stated that "[Occluded features and oracles are different in a number of ways] *but still it should be no surprise that they are interchangable*" ([Sandewall, 1994], page 255). The concept "Occlusion" will be properly introduced in Chapter 4.

Lin suggests that the oracles could be viewed as probabilities of certain effects to take place. However, since he does not develop this idea, the introduction of oracles is not completely convincing, at least not from a knowledge representation point of view. In Chapter 4 we will show how regression can be defined for nondeterministic theories without the use of oracles, but with Occlusion.

2.1.3 Situation Calculus as a Knowledge Representation

In Section 1.1.3 we posed a number of questions that could be asked about a knowledge representation. Here, we will analyze SitCalc from that perspective.

The effort that Lin puts into finding successor state axioms for SitCalc with nondeterministic actions implies that that he is not only interested in a theoretical tool, but also in the possibility of implementing the logic with the regression operator as the primary reasoning mechanism. Yet, it is clear that it is nondeterminism that is the aspect he sets out to study. The ontological commitments set by SitCalc are primarily of interest for their way of representing the flow of time — as a branching time structure. A situation is a sequence of actions, which means that every finite sequence of actions is possible. This implies that it is possible to view the temporal structure as a tree, where the initial situation S_0 is the root and the sequences are the branches. This temporal structure enables hypothetical reasoning on the object level, since two different branches denote two different possible courses of events which can be compared on the object level. Lin and Reiter [1997] argue that their version of SitCalc is consistent with the dynamics of database update, and that it is very close to how the database community views relational databases.

2.2 The approach of Łukaszewicz and Madalińska-Bugaj

An approach similar to our work described in Chapter 4 is that of Łukaszewicz and Madalińska-Bugaj [1995] who apply Dijkstra's semantics for programming languages to formalize reasoning about action and change. They define a small programming language with an assignment command, sequential composition command, alternative command, and skip command, and define the semantics of the commands in terms of the *weakest liberal precondition*, *wlp*, and *strongest postcondition*, *sp*, such that for a command S and a description of a set of states α, $wlp(S, \alpha)$ describes a set of states such that if the command is executed in any of them, the effect of S will be one of the states described by α. For *sp*, we have that $sp(S, \alpha)$ describes a set of states such that if S is executed in any of the states described by α, the effects will belong to $sp(S, \alpha)$. The "descriptions" of states mentioned are simply formulae in propositional logic. We present the approach in more detail to facilitate a comparison to our approach, presented in Chapter 4. The programming language consists of a skip command, an assignment command, a sequential composition command, and an alternative command. The semantics of the commands are defined as follows:

- *skip*:
$$wlp(skip, \alpha) = sp(S, \alpha) = \alpha,$$
that is, *skip* denotes the empty command, the command with no effects.

- *Assignment*: Let α be a propositional formula. Then $\alpha[f \leftarrow V]$ denotes the simultaneous substitution of all occurences of the symbol f for $V \in \{\mathbf{T}, \mathbf{F}\}$[5] in the formula α. The effect of the assignment command, $x := V$, should be that x is true in all states after the command has been executed, if $V = \mathbf{T}$, else false.
$$wlp(x := V, \alpha) = \alpha[x \leftarrow V],$$
and for *sp*:
$$sp(x := V, \alpha) = \begin{cases} x \wedge (\alpha[x \leftarrow \mathbf{T}] \vee \alpha[x \leftarrow \mathbf{F}]) & \text{If } V = \mathbf{T} \\ \neg x \wedge (\alpha[x \leftarrow \mathbf{T}] \vee \alpha[x \leftarrow \mathbf{F}]) & \text{If } V = \mathbf{F} \end{cases},$$

[5]The symbols $\mathbf{T}$ and $\mathbf{F}$ henceforth denote the truth values *True* and *False*, respectively.

that is, x should have a value according to V after the execution of the command, but only in states in which α is true regardless of the value of x in them.

- *Sequential composition*: For two commands S_1 and S_2, we let $S_1; S_2$ denote that the commands are executed in sequence, with S_1 first.

$$wlp(S_1; S_2, \alpha) = wlp(S_1, wlp(S_2, \alpha)),$$

$$sp(S_1; S_2, \alpha) = sp(S_2, sp(S_1, \alpha)).$$

- *Alternative*: This command is of the form

$$\textbf{if } B_1 \rightarrow S_1 [] \ldots [] B_n \rightarrow S_n \textbf{ fi},$$

where $B_1, \ldots, B_n$ are boolean expressions (*guards*) and $S_1, \ldots, S_n$ are any commands. The semantics of this command (henceforth referred to as IF) is given by

$$wlp(\mathrm{IF}, \alpha) = \bigwedge_{i=1}^{n} [B_i \supset wlp(S_i, \alpha)],$$

$$sp(\mathrm{IF}, \alpha) = \bigvee_{i=1}^{n} [sp(S_i, B_i \wedge \alpha)],$$

where $\supset$ denotes logical implication. Thus, if none of the guards is true the execution aborts, else one of the expressions $B_i \rightarrow S_i$ with true B_i, is randomly selected and S_i is executed.

Łukaszewicz and Madalińska-Bugaj are mostly interested in a particular class of computations, namely *initially α and finally β under control of S*, that is, computations S that start in one of the states described by α and end in one of the states described by β. Typically, the problems they want to model consist of given (partial) descriptions of the initial and final states, and a given sequence of commands (actions). The reasoning tasks might then be prediction (to prove that something holds after the sequence has been executed) or postdiction (to prove that something holds before the sequence), or that something holds somewhere in the middle of the sequence. Here, we will only briefly explain how the reasoning is performed.

For a "pure" prediction problem, where a statement φ is to be proven to hold after a command S, given an initial constraint α, they check if

$sp(S, \alpha) \models \varphi$, where $\models$ denotes classical deductive inference in propositional logic. Thus, prediction is handled by *progression.*

For pure postdiction, where a statement φ is to be proven to hold before a command S, given an final constraint β, they check if $\neg wlp(S, \neg\beta) \models \varphi$. The use of negations here will be carefully investigated in Chapter 4. Postdiction is handled by *regression.*

For the third case, where $S = S_1; \ldots; S_n$ and φ is to be proven to hold after S_k but before S_{k+1}, for $1 < k < n$, with initial constraint α and final constraint β, they check if $sp(S_1; \ldots; S_k, \alpha) \wedge \neg wlp(S_{k+1}; \ldots; S_n, \neg\beta) \models \varphi$. This case is handled by progression *and* regression.

2.2.1 Knowledge Representation Issues

The work by Łukaszewicz and Madalińska-Bugaj focuses strongly on theoretical aspects of KR, and especiallly the possibilities of using Dijkstra's semantics for RAC purposes. The ontological commitments are the same as for any imperative programming language, thus its consistency with the real world cannot be questioned. Since the semantics of the language is given by *wlp* and *sp*, and the proposed reasoning mechanisms also are *wlp* and *sp*, it is hard to comment on how much of a theory of intelligent reasoning the formalism is. The intended inferences are hard-wired in the semantics, via *wlp* and *sp*[6].

[6] In chapter 5 we present a formalism where the intended conclusions are hard-wired in the semantics.

Chapter 3

Preliminaries

Sandewall [1994] proposed a systematic approach to RAC that includes a framework in which it is possible to assess the range of applicability of existing and new logics of action and change. As part of the framework, several logics are introduced and assessed correct for particular classes of action scenario descriptions. The most general class covered by the framework, $\mathcal{K}$-**IA** and one of its associated entailment methods, PMON, permits scenarios with nondeterministic actions, actions with duration, partial specification at any state in the scenario, context dependency, and incomplete specification of the timing and order of actions. Doherty and Łukaszewicz [1994] showed how the entailment methods assessed in Sandewall's framework could be described by circumscription policies, and Doherty [1994] gave a first-order formulation of PMON which uses a second-order circumscription axiom, showing that the second-order formula always could be reduced to first-order. In [Gustafsson and Doherty, 1996] PMON was extended to deal with certain types of ramification.

In [Karlsson, 1997] fundamental notions from PMON were used to extend SitCalc to facilitate planning with nondeterministic actions and incomplete information.

Kvarnström and Doherty [1997] have developed a visualization tool, VTAL, for PMON, which is currently used for research purposes.

Our version of PMON is propositional, and we allow nondeterministic action, actions with duration, and arbitrary observations not inside action-duration intervals. We will also require actions to be totally ordered; actions are not allowed to overlap. Furthermore, we view actions as "encapsulated", that is, that we are not interested in what goes on during the execution of an action. The interested reader should consult [Doherty, 1997] for details of the full flavoured logics.

3.1 PMON

We will use the language $\mathcal{L}$ which is a many-sorted first-order language. For the purpose of this thesis we assume two sorts: A sort $\mathcal{T}$ for time and a sort $\mathcal{F}$ for propositional fluents. The sort $\mathcal{T}$ will be the set of natural numbers.

The language includes two predicate symbols H[1] and *Occlude*, both of type $\mathcal{T} \times \mathcal{F}$. The numerals $0, 1, 2, \ldots$ and the symbols $\mathbf{s}$ and $\mathbf{t}$, possibly subscripted, denote natural numbers, i.e. constants of type $\mathcal{T}$ (*time points*).

PMON is based on *action scenario descriptions* (scenarios), which are narratives consisting of three components: OBS, a set of observations stating the values of fluents at particular time points, ACT, a set of action laws, and SCD, a set of schedule statements that state which and when (in time) actions occur in the scenario.

Example 3.1.1 A scenario where the pin was initially held over the checkerboard, then dropped and finally observed to be on a white square (and not on a black one) is formalized as

OBS1	$H(0, over\_board) \wedge \neg H(0, white) \wedge \neg H(0, black)$
ACT	$[s,t]Drop \Rrightarrow H(s, over\_board) \rightarrow [s,t]\gamma$
SCD	$[3,5]Drop$
OBS2	$H(6, white) \wedge \neg H(6, black)$,

where $[s,t]\gamma$ is the formula

$$\begin{aligned}&\neg H(t, over\_board) \wedge (H(t, white) \vee H(t, black)) \wedge \\ &\forall t'.s < t' \leq t \rightarrow Occlude(t', holding) \wedge \\ &Occlude(t', white) \wedge Occlude(t', black),\end{aligned} \tag{3.1}$$

The first observation OBS1 states that at time point 0 the pin is held over the board, and is not on a black or a white square. The sole action of the scenario *Drop* states that it is executed between time points s and t and then, if the pin is over the board at time point s, it will no longer be over the board and it will be on a black or a white squre at time point t. At every time point between (not including) s and t the fluents *over_board*, *white*, and *black* are *Occluded*, which means that they are allowed to change their

[1] Not to be confused with h which is the corresponding situation calculus predicate.

value during that interval. Formally, this means that they are excluded from the minimization-of-change policy. This will be discussed in detail below. The schedule statement SCD states that the action *Drop* will be executed between time points 3 and 5. The schedule statements of a scenario are used to instantiate temporal variables of the action laws. The instantiated action law (*action instance*) for *Drop* in Example 3.1.1 will be

$$\begin{aligned}
&H(3, over_board) \rightarrow \\
&\quad \neg H(5, over_board) \wedge (H(5, white) \vee H(5, black)) \wedge \\
&\quad Occlude(4, over_board) \wedge Occlude(5, over_board) \\
&\quad Occlude(4, white) \wedge Occlude(5, white) \\
&\quad Occlude(4, black) \wedge Occlude(5, black),
\end{aligned}$$

where we have expanded the universal quantification over the temporal variable.□

The way nondeterministic actions are formalized in PMON certainly deals with the problem of compact and intuitive representation, discussed in Section 2.1.2.

In Section 2.1.2 we saw that in Lin's formalism $h(holding, s)$ is a *qualification* of the action *drop*. To illustrate conditions for the action to have effects (context dependency), we introduce *over_board*, that denotes that *Drop* only has effects if the pin is dropped onto the board. Such conditions can be modelled in SitCalc, but not in Lin's framework, at least not when regression is to be used.

H formulae and Observations

Boolean combinations of H literals (i.e. possibly negated atomic formulae), where every literal mentions the same time point, are called *H formulae.* An H formula, δ, where every literal only mentions time point $\mathbf{s}$, will be written $[\mathbf{s}]\delta$. An *observation* is an H formulae.

Action Laws

A *reassignment* is a statement $[\mathbf{s}, \mathbf{t}]f := b$, where $f \in \mathcal{F}$ and b is a truth value symbol ($\mathbf{T}$ for true, and $\mathbf{F}$ for false). The statement $[\mathbf{s}, \mathbf{t}]f := \mathbf{T}$ is an abbreviation of $H(\mathbf{t}, f) \wedge \forall t.\mathbf{s} < t \leq \mathbf{t} \rightarrow Occlude(t, f)$, and $[\mathbf{s}, \mathbf{t}]f := \mathbf{F}$ expands to $\neg H(\mathbf{t}, f) \wedge \forall t.\mathbf{s} < t \leq \mathbf{t} \rightarrow Occlude(t, f)$. These formulae will be called *expanded* reassignments. A *reassignment formula* $[\mathbf{s}, \mathbf{t}]\alpha$, is a boolean combination of reassignments, all mentioning the same temporal constants $\mathbf{s}$ and $\mathbf{t}$. Expanded reassignment formulae are constructed from expanded reassignments.

An *action law* is a statement

$$\begin{aligned}&[s,t]A \Rrightarrow \\ &\quad \textstyle\bigwedge_{i=1}^{n}[[s]precond_i^A \to [s,t]postcond_i^A],\end{aligned}$$

where A is an action designator (the name of the action), s and t are variables (*temporal* variables) over the natural numbers, each $precond_i^A$ is either a conjunction of H literals only mentioning the temporal variable s, or the symbol $\mathbf{T}$, such that the preconditions $precond_i^A$ are mutually exclusive, and every $postcond_i^A$ is a reassignment formula. Intuitively, ACT contains *rules* for expanding schedule statements into action axioms, in which the temporal variables are instantiated.

An action is said to be *deterministic* iff none of its postconditions include disjunctions. An action is *admissible* if no precondition or postcondition is a contradiction. We will call the conjuncts of an action law *branches.*

Definition 3.1.2 (Effect Formulae)
Every expanded reassignment formula can be written as

$$[\mathbf{t}]\psi \wedge (\forall t.\mathbf{s} < t \leq \mathbf{t} \to \theta),$$

where ψ is an H formula, and θ a conjunction of positive *Occlude* literals. The ψ part of an expanded reassignment formula will be called an *effect formula.*$\Box$

Schedule Statements

A *schedule statement* is a statement, $[\mathbf{s}, \mathbf{t}]A$, such that $\mathbf{s}$ and $\mathbf{t}$ are natural numbers, $\mathbf{s} < \mathbf{t}$, and A is an action designator for which there exists an action law in ACT.

No Change Premises

The occlusion of fluents that are reassigned captures the intuition of possible change, but we also need a syntactic component that captures the intuition that a fluent cannot change value *unless* it is occluded. This is taken care of by the *no change premise*, NCH, formulated as

$$\forall f, t. \neg Occlude(t+1, f) \rightarrow (H(t, f) \equiv H(t+1, f)).$$

Action Scenario Description

An *action scenario description* (scenario) is a triple $\langle \text{OBS}, \text{ACT}, \text{SCD} \rangle$, where OBS is a set of observations, ACT a set of action laws, and SCD a set of schedule statements such that the actions are totally ordered. Furthermore, let SCD(ACT) denote a set of formulae such that for every schedule statement in SCD we instantiate the temporal variables of the corresponding action law (in ACT) with the two time points in the schedule statement. We will call such formulae *action instances*.

PMON Circumscription Policy

To minimize change, the action instances are circumscribed (by second-order circumscription as shown in [Doherty and Łukaszewicz, 1994, Doherty, 1994]), i.e. by minimizing the *Occlude* predicate, and then by filtering with the observations and nochange premises. For a scenario description $\Upsilon = \langle \text{OBS}, \text{ACT}, \text{SCD} \rangle$ we have

$$\begin{aligned}
&PMON(\Upsilon) = \\
&\quad Circ_{SO}(\text{SCD}(\text{ACT})(Occlude); \langle Occlude \rangle) \cup \\
&\quad \{\text{NCH}\} \cup \text{OBS} \cup \Gamma,
\end{aligned}$$

where $Circ_{SO}$ denotes the second-order circumscription operator, and Γ is the set of unique names axioms. Doherty [1994] shows that this second-order theory can always be reduced to a first-order theory that provides a *completion* (or, definition) of the *Occlude* predicate. For Example 3.1.1 the completion will be the following:

$$\begin{aligned}
&\forall t, f. Occlude(t, f) \equiv \\
&\quad H(3, over_board) \wedge \\
&\quad ((t = 4 \wedge f = over_board) \vee (t = 5 \wedge f = over_board) \vee \\
&\quad (t = 4 \wedge f = white) \vee (t = 5 \wedge f = white) \vee \\
&\quad (t = 4 \wedge f = black) \vee (t = 5 \wedge f = black))
\end{aligned} \tag{3.2}$$

A classical model for a circumscribed scenario will be called an *intended model*. A scenario is consistent if it has an intended model, otherwise it is inconsistent.

Let M_Υ be the class of intended models for a PMON scenario, Υ. For $t \in \mathcal{T}$, we let $M_\Upsilon(t)$ denote the set of propositional interpretations for the fluents in Υ at time point t, in the obvious way. A member s of $M_\Upsilon(t)$ is called a *state*.

We say that a scenario Υ *entails* a statement $[\mathbf{t}]\varphi$, and writes $\Upsilon \mid\!\sim [\mathbf{t}]\varphi$, iff $PMON(\Upsilon) \vdash [\mathbf{t}]\varphi$, where $\vdash$ denotes classical deductive inference. Furthermore, a schedule statement $[\mathbf{s}, \mathbf{t}]A$ is said to entail a statement $[\mathbf{t}]\varphi$, written $[\mathbf{s}, \mathbf{t}]A \mid\!\sim [\mathbf{t}]\varphi$, iff $\langle \emptyset, \textsc{act}, [\mathbf{s}, \mathbf{t}]A\rangle \mid\!\sim [\mathbf{t}]\varphi$.

The completion of *Occlude* has a useful ramification: It *conditionalizes* the branches of the actions in the scenario, i.e. when the completion axiom is added to the theory, it is no longer possible for a precondition to be false while its postconditions are true. For Example 3.1.1 we see that the definition of *Occlude* (formula 3.2) ensures that the precondition of the action *Drop*, $H(3, \mathit{over\_board})$, must be true for $\mathit{Occlude}(t, f)$ to be true, for any t and f. This behavior is similar to the behavior of if-then statements in programming languages, and makes it possible to use Dijkstra's *wlp* formula transformer.

3.2 Knowledge Representation Issues

PMON originated as a theoretical construction in [Sandewall, 1994], and has been analysed and extended as such in a number of papers ([Doherty, 1994], [Doherty and Łukaszewicz, 1994], [Gustafsson and Doherty, 1996], [Karlsson, 1997]). However, much work is currently being undertaken to implement PMON in pragmatically efficient ways (the results in Chapter 4 are steps in this direction). The primary ontological commitment imposed by PMON is the notion of a scenario, where we have to explicitly model action occurences. However, as argued in [Ghallab and Laruelle, 1994], the possibility of being able to explicitly model time and to reason with time points and intervals is crucial for automatic control applications.

Chapter 4

Regression

In this chapter we explore two tracks of interest: the development of a computational mechanism for reasoning with PMON, and the relationship of PMON to imperative programming languages. For the first track we define a regression operator which is the key to the results in the second track.

Basically, regression is the technique of generating a state description, α, of what holds before some action, A, is executed, given some state description, β, presumably holding after A is executed. Also, we want α to contain all possible states such that, if A is executed in any of them the result will belong to β. Without this last constraint, the empty set would always be a sound regression result.

In Section 2.1.1 we showed how Reiter [1991] generates a biconditional relationship between the states holding before and after (that is α and β as above) the execution of an action for deterministic actions. It is easy to see that such a relation does not hold generally if we allow nondeterministic actions. Take, for example, the action of flipping a coin, which either results in $tails$ or $\neg tails$, but not both. If we observe that $tails$ holds after flipping, what are the sufficient conditions before the actions, so that we are guaranteed that $tails$ will be true? Well, there are no such conditions, since the result of the action is nondeterministic and not dependent on the state in which the action was invoked. On the other hand, what states would make it possible for the action to have the effect $tails$? The answer is "all of them", since no state would prohibit the possibility.

This distinction between *sufficient* and *necessary* conditions will be formalized below using a classical computer science approach, the weakest liberal precondition operator, wlp.

In Proposition 4.1.10 we establish that *wlp* computations are equivalent to PMON entailment.

4.1 *Wlp*

We will introduce Dijkstra's semantics for programming languages (see for example [Dijkstra, 1976, Dijkstra and Scholten, 1990, Gries, 1981]), but in the setting of PMON. We also present theorems which connect the states before and after the execution of actions. Originally, the theory was developed to provide semantics for small programming languages with a higher ordered function. This function was a predicate transformer that, given a command, S, in the language and a machine state, s_i, returned a set, W, of machine states such that all states in which S could be executed and terminate in s_i, if it at all terminated, belonged to W. This predicate transformer was called the *weakest liberal precondition*, *wlp*. For the purpose of programming languages, a stronger variant of *wlp*, namely the weakest precondition operator *wp*, was developed[1] as well. It was defined only for terminating actions. Since all the actions used here are terminating (we do not have any indefinite loops), *wp* and *wlp* coincide. For historical reasons, we will call the operator *wlp*.

Let φ be an atomic H formula, f a fluent and $V \in \{\mathbf{T}, \mathbf{F}\}$. We write $[\mathbf{s}]\varphi[f \leftarrow V]$ to denote that all H statements[2] mentioning the fluent f are simultaneously replaced by V, and where the time argument of the H statements is changed to $\mathbf{s}$ ($[\mathbf{s}]$ will be called the *time mark* of the formula). As a consequence, $[\mathbf{s}]\varphi[]$ denotes the replacement of all time arguments in φ with $\mathbf{s}$. The notation $[\mathbf{s}]\varphi[f_1 \leftarrow V_1, \ldots, f_n \leftarrow V_n]$ denotes the simultaneous substitution of the H formulae for the truth values. We will let formulae with nested time marks, such as $[\mathbf{s}][\mathbf{t}]\alpha \wedge [\mathbf{t}']\beta$, be equivalent to formulae where all internal time marks are removed, that is, only the outmost time mark will remain. So,

$$[\mathbf{s}][\mathbf{t}]\alpha \wedge [\mathbf{t}']\beta \equiv [\mathbf{s}]\alpha \wedge \beta$$

Since the operations below are syntactical, we will have to assume that the effect formulae (see Definition 3.1.2) ψ are extended with a conjunction of $H(t, \mathbf{f}) \vee \neg H(t, \mathbf{f})$ for every occluded fluent not already mentioned in ψ[3].

[1]In the literature *wlp* is generally defined in terms of *wp*.
[2]*Not* the possible negation in front of them.
[3]Semantically, this has no effect, since we are adding tautologies to the formula.

These effect formulae will be called *extended* effect formulae. For example, the action of flipping a coin could be formalized as follows:

$$[s,t]Flip \Rrightarrow \forall t'.s < t' \leq t \rightarrow Occlude(t', \mathbf{tails}),$$

where the the effect formula is $\mathbf{T}$. The same action with an extended effect formula appears as follows:

$$\begin{aligned}&[s,t]Flip \Rrightarrow \\ &\quad (H(t,\mathbf{tails}) \vee \neg H(t,\mathbf{tails})) \wedge \\ &\quad \forall t'.s < t' \leq t \rightarrow Occlude(t', \mathbf{tails})\end{aligned}$$

Definition 4.1.1 (Weakest liberal precondition, *wlp*)
Let φ be an H formula. We define wlp inductively over extended effect formulae of the action $[\mathbf{s},\mathbf{t}]A$, so that, for all fluents $f \in \mathcal{F}$

$$wlp(H(\mathbf{t}, f), \varphi) \stackrel{\text{def}}{=} [\mathbf{s}]\varphi[f \leftarrow \mathbf{T}] \tag{4.1}$$

$$wlp(\neg H(\mathbf{t}, f), \varphi) \stackrel{\text{def}}{=} [\mathbf{s}]\varphi[f \leftarrow \mathbf{F}] \tag{4.2}$$

$$wlp(\alpha \wedge \beta, \varphi) \stackrel{\text{def}}{=} [\mathbf{s}]wlp(\beta, [\mathbf{t}]wlp(\alpha, \varphi)) \tag{4.3}$$

$$wlp(\alpha \vee \beta, \varphi) \stackrel{\text{def}}{=} [\mathbf{s}]wlp(\alpha, \varphi) \wedge wlp(\beta, \varphi) \tag{4.4}$$

Furthermore, let $[\mathbf{s},\mathbf{t}]A$ be a schedule statement for which the corresponding law has n branches. Then

$$\begin{aligned}&wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) \stackrel{\text{def}}{=} \\ &\quad [\mathbf{s}](\textstyle\bigwedge_{i=1}^{n}[precond_i^A \rightarrow wlp(postcond_i^A, \varphi)]) \\ &\qquad\qquad\qquad \wedge \\ &\qquad\quad (\textstyle\bigwedge_{i=1}^{n}[\neg precond_i^A] \rightarrow [\mathbf{s}]\varphi[])\end{aligned} \tag{4.5}$$

The second conjunct encodes that if none of the preconditions was true then φ had to be true at the beginning of the action. We define the *conjugate* of wlp, wlp^*, as $wlp^*(S, \alpha) = \neg wlp(S, \neg\alpha)$.□

Note that wlp is applied to the possible *effects* of an action in (4.1) – (4.4), and is generalized to complete action instances in (4.5).

Definition (4.3) is based on a sequential underlying computational model. If we implement wlp, we will have to perform the reassignments in some order, and here we choose to first apply α and then β. Since we have assumed that the actions are admissible, the order does not matter.

Definition (4.4) handles nondeterminism. It guarantees that no matter which of α or β is executed, we will end up in a state satisfying φ. The only states that can guarantee this are those that can start in α *and* β and end in φ.

For (4.5) we can note that *wlp* should generate every state, σ, such that if a precondition is true in σ, *wlp* applied to the corresponding postcondition and some φ should be true in σ too. This excludes every action branch that would not terminate in a state satisfying φ. The second conjunct of (4.5) makes *wlp* work exhaustively, that is if none of the preconditions are true then φ is true, before the action.

The conjugate $wlp^*([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi)$ should, analogously to *wlp*, be interpreted as: "The set of all states at $\mathbf{s}$ such that the execution of A in any of them does *not* end in a state at $\mathbf{t}$ satisfying $\neg\varphi$."

4.1.1 Example 3.1.1 Revisited

We illustrate *wlp* by computing the weakest liberal precondition for *white* or *black* to hold after the *Drop* action. The scenario is formalized as follows:

OBS1 $H(0, over\_board) \wedge \neg H(0, white) \wedge \neg H(0, black)$
ACT $[s,t]Drop \Rrightarrow H(s, over\_board) \rightarrow [s,t]\gamma$
SCD $[3,5]Drop$
OBS2 $H(6, white) \wedge \neg H(6, black)$,

where $[s,t]\gamma$ is the formula

$$\begin{aligned}
&\neg H(t, over_board) \wedge (H(t, white) \vee H(t, black)) \wedge \\
&\forall t'.s < t' \leq t \rightarrow Occlude(t', holding) \wedge \\
&Occlude(t', white) \wedge Occlude(t', black),
\end{aligned}$$

We are only interested in the *Drop* action, which is instantiated as follows:

$$\begin{aligned}
&H(3, over_board) \rightarrow \\
&\quad \neg H(5, over_board) \wedge (H(5, white) \vee H(5, black)) \wedge \\
&\quad Occlude(4, over_board) \wedge Occlude(5, over_board) \\
&\quad Occlude(4, white) \wedge Occlude(5, white) \\
&\quad Occlude(4, black) \wedge Occlude(5, black).
\end{aligned}$$

The effect formula is in this case $\neg H(5, over\_board) \wedge (H(5, white) \vee H(5, black))$, and since all occluded fluents already occur in the effect formula ψ, the extended effect formula coincides with ψ.

It should be clear that the only way in which we can be guaranteed that *white* or *black* is true after *Drop* is if *over_board*, or *white* or *black*, was true before the action.

$$\begin{aligned}
&wlp([3,5]Drop, H(5, white) \vee H(5, black)) = \\
&(H(3, over_board) \rightarrow \\
&\quad [3]wlp(\neg H(5, over_board) \wedge (H(5, white) \vee H(5, black)), \\
&\qquad H(5, white) \vee H(5, black))) \wedge \\
&(\neg H(3, over_board) \rightarrow \\
&\quad H(3, white) \vee H(3, black))
\end{aligned} \tag{4.6}$$

Now, we focus on the right-hand side of the first implication:

$$\begin{aligned}
&[3]wlp(\neg H(5, over_board) \wedge (H(5, white) \vee H(5, black)), \\
&\quad H(5, white) \vee H(5, black))) = \\
&[3]wlp(H(5, white) \vee H(5, black), \\
&\quad [5]wlp(\neg H(5, over_board), H(5, white) \vee H(5, black))) = \\
&[3]wlp(H(5, white) \vee H(5, black), \\
&\quad H(5, white) \vee H(5, black)) = \\
&[3](wlp(H(5, white), H(5, white) \vee H(5, black)) \wedge \\
&\quad wlp(H(5, black), H(5, white) \vee H(5, black))) = \\
&(\mathbf{T} \vee H(3, black)) \wedge (H(5, white) \vee \mathbf{T}) = \\
&\mathbf{T}
\end{aligned}$$

We computed $wlp([3,5]\gamma, H(5, white) \vee H(5, black))$, and the first conjunct of $[3,5]\gamma$, $[3,5]over\_board := \mathbf{F}$, did not not have any effect on $H(5, white) \vee H(5, black)$. When *wlp* was applied to the three disjuncts of $[3,5]\gamma$ they were all computed to $\mathbf{T}$. Thus,

$$wlp([3,5]\gamma, H(5, white) \vee H(5, black)) = \mathbf{T},$$

and (4.6) will be equivalent to

$$\begin{aligned}
&(H(3, over_board) \rightarrow \mathbf{T}) \wedge \\
&(\neg H(3, over_board) \rightarrow H(3, white) \vee H(3, black)) \\
&\equiv \\
&\neg H(3, over_board) \rightarrow H(3, white) \vee H(3, black),
\end{aligned}$$

which is what we wanted.

4.1.2 Weakness Theorem and Logical Properties of *wlp*

The correctness of wlp is in Dijkstra's original setting quite obvious. But since the semantics of PMON is defined independently of wlp, we will have to prove that our version actually is weakest. This fact will imply most of the forthcoming results. We start with some helpful lemmas.

Lemma 4.1.2 Let s be a state and α a formula over some vocabulary. Let $\mathbf{f}$ be a member of that vocabulary. Define $s' = s/\mathbf{f}$ to be the state that maps all fluents to the same truth value as s, except for $\mathbf{f}$ where $s'(\mathbf{f}) = \mathbf{T}$ regardless of what the value of $s(\mathbf{f})$ is. Analogously we define $s/\neg\mathbf{f}$.

If s does not satisfy $\alpha[\mathbf{f} \leftarrow \mathbf{T}]$ ($\alpha[\mathbf{f} \leftarrow \mathbf{F}]$), then $s/\mathbf{f}$ ($s/\neg\mathbf{f}$) does not satisfy α.
Proof: Structural induction over α.□

Lemma 4.1.2 can easily be generalized to handle multiple substitutions of the form $\alpha[\mathbf{f}_1 \leftarrow V_1, \ldots, \mathbf{f}_n \leftarrow V_n]$.

Lemma 4.1.3

$$wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi \wedge \varphi') \equiv wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) \wedge wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi'),$$

that is, wlp distributes over conjunction in the second argument.
Proof: Structural induction over the definition of wlp.□

Now we prove the weakness theorem. The basic idea is to prove that there can be no weaker formula than the result of the wlp computation that is true before the execution of the action, if the scenario should be consistent. Thus, we show that for an arbitrary formula σ which is true before action A, the formula σ must imply the result of the wlp computation. We will use contraposition of the implication for all of the cases.

Theorem 4.1.4 Let $[\mathbf{s},\mathbf{t}]A$ be an admissible action and $[\mathbf{t}]\varphi$ an H formula. If $\langle [\mathbf{s}]\sigma, law, [\mathbf{s},\mathbf{t}]A\rangle \mathrel{|\!\sim} [\mathbf{t}]\varphi$ then $[\mathbf{s}]\sigma \to wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi)$, for any H formula $[\mathbf{s}]\alpha$.
Proof: We assume that $[\mathbf{s},\mathbf{t}]A \cup \{[\mathbf{s}]\alpha\} \mathrel{|\!\sim} [\mathbf{t}]\varphi$, and prove the theorem with structural induction over actions. We also assume that the extended effect formulae ψ are on CNF. First, we can note that if $\alpha \equiv \mathbf{F}$, the theorem holds trivially, so we assume that α is consistent.

1. $\psi = H(\mathbf{t}, f)$.
   We know that $wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) = [\mathbf{s}]\varphi[f \leftarrow \mathbf{T}]$ and we assume that $[\mathbf{s}]\sigma \wedge \neg\varphi[f \leftarrow \mathbf{T}]$ is consistent. Thus, there is state, s, which satisfies σ but not $\varphi[f \leftarrow \mathbf{T}]$. By Lemma 4.1.2 we know that the values in s cannot be flipped by psi so that the resulting state may satisfy φ. Thus, s cannot be an initial state, and the result follows.

2. $\psi = \neg H(\mathbf{t}, f)$.
   This case is proven analogously to the case above.

3. $\psi = \beta_1 \vee \ldots \vee \beta_m$, where β_i is an H literal.
   $wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) = [\mathbf{s}] \bigwedge_{i=1}^{m} wlp(\beta_i, \varphi)$, by definition. We assume that $[\mathbf{s}]\sigma \wedge \bigvee_{i=1}^{m} \neg wlp(\beta_i, \varphi)$ is consistent. This implies that there exists a state s that satisfies σ and, for at least one i, does not satisfy $wlp(\beta_i, \varphi)$. Now, set $V_i = \mathbf{T}$ if $\beta_i = H(\mathbf{t}, f_i)$, else $V_i = \mathbf{F}$. Then $wlp(\beta_i, \varphi) = [s]\varphi[f_i \leftarrow V_i]$. Since s does not satisfy $\varphi[f_i \leftarrow V_i]$, the flipped version does not satisfy φ (according to Lemma 4.1.2). If the execution of A is invoked in s, it does not matter how the value of f_i is changed, the execution will not terminate in a state satisfying φ, and we have a contradiction.

4. $\psi = \alpha_1 \wedge \ldots \wedge \alpha_n$, where α_i is a disjunction of H literals.
   By definition $wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) = wlp(\alpha_n, \ldots wlp(\alpha_1, \varphi)\ldots)$. Let s be state satisfying $\sigma \wedge \neg wlp(\alpha_1, \ldots wlp(\alpha_n, \varphi)\ldots)$. If $\alpha_i = \beta_1^i \vee \ldots \vee \beta_{k_i}^i$, where β_j^i is an H literal, we know that $wlp(\alpha_n, \varphi) = \bigwedge_{i=1}^{k_n} \beta_{k_i}^n$, which means that

   $$wlp(\alpha_{n-1}, wlp(\alpha_n, \varphi) = \bigwedge_{i=1}^{k_n} wlp(\alpha_{n-1}, wlp(\beta_{k_i}^n, \varphi)),$$

   by Lemma 4.1.3.
   By induction we get a conjunction of nested wlp terms, where the first argument is an H formula. If we let all the H formulas in the conjuncts have effect on φ, the result will be a conjunction of formulas of the form $\varphi[f_{\beta_i}^{\alpha_1} \leftarrow V_{\beta_j}^{\alpha_1}, \ldots, f_{\beta_j}^{\alpha_n} \leftarrow V_{\beta_j}^{\alpha_n}]$, that is every combination of disjuncts in the α_i will take effect in some conjunct. We know that s does not satisfy one of these conjuncts, which implies that it cannot be an initial state.

5. $A = \bigwedge_{i=1}^{n} [[s]precond_i \rightarrow [s,t]postcond_i]$.

$$
\begin{aligned}
&wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) = \\
&\quad (\textstyle\bigwedge_{i=1}^{n}[precond_i^A \to wlp(postcond_i^A, \varphi)]) \\
&\qquad\qquad \wedge \\
&\quad (\textstyle\bigwedge_{i=1}^{n}[\neg precond_i^A] \to [\mathbf{s}]\varphi[])
\end{aligned}
$$

Again, we examine a state s satisfying $[s]\sigma \wedge \neg wlp([s,t]A, [t]\varphi)$, where $\neg wlp([s,t]A, [t]\varphi)$ is equivalent to

$$
\begin{aligned}
&(\textstyle\bigvee_{i=1}^{n}[precond_i^A \wedge \neg wlp(postcond_i^A, \varphi)]) \\
&\qquad\qquad \vee \\
&(\textstyle\bigwedge_{i=1}^{n}[\neg precond_i^A] \wedge \neg[\mathbf{s}]\varphi[])
\end{aligned}
$$

It easy to verify that s cannot be an initial state.

□

We have proved that *wlp* in fact provides the *weakest* precondition. Now we will establish relationships between PMON entailment and *wlp*, and investigate when successor state axioms can be generated.

For all the theorems below we assume that the actions are admissible.

Proposition 4.1.5 $[\mathbf{s},\mathbf{t}]A \mathrel{|\!\sim} wlp([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi) \to [\mathbf{t}]\varphi$.
Proof (sketch): Follows from Theorem 4.1.4.□

Corollary 4.1.6 $[\mathbf{s},\mathbf{t}]A \mathrel{|\!\sim} [\mathbf{t}]\varphi \to wlp^*([\mathbf{s},\mathbf{t}]A, [\mathbf{t}]\varphi)$.
Proof: Contraposition of the implication yields a formula which holds according to proposition 4.1.5.□

Proposition 4.1.7 $wlp(S,\varphi) \equiv wlp^*(S,\varphi)$ for deterministic actions.
Proof: The cases correspond to the definition of *wlp*.

1.

$$
\begin{aligned}
&wlp(f := V, \varphi) = \\
&\quad \varphi[f \leftarrow V] \equiv \\
&\quad \neg\neg\varphi[f \leftarrow V] \equiv \\
&\quad \neg wlp(f := V, \neg\varphi) = \\
&\quad wlp^*(f := V, \varphi),
\end{aligned}
$$

where $V \in \{\mathbf{T}, \mathbf{F}\}$.

2. Let $S = f_1 := V_1 \wedge \ldots \wedge f_n := V_n$. Since S is consistent we know that if $f_i = f_j$, then $V_i = V_j$, for $1 \leq i, j \leq n$. We can, therefore, safely say

$$wlp(S, \varphi) = \varphi[f_1 \leftarrow V_1, \ldots, f_n \leftarrow V_n],$$

which brings us back to case 1.

3. Let $[\mathbf{s}, \mathbf{t}]A$ be a deterministic action. By definition we know that

$$\begin{gathered} wlp([\mathbf{s}, \mathbf{t}]A, [\mathbf{t}]\varphi) = \\ [\mathbf{s}]((\textstyle\bigwedge_{i=1}^{n}[precond_i^A \rightarrow wlp(postcond_i^A, \varphi)])) \\ \wedge \\ (\textstyle\bigwedge_{i=1}^{n}[\neg precond_i^A] \rightarrow \varphi)) \end{gathered}$$

and for the conjugate, that

$$\begin{gathered} wlp^*([\mathbf{s}, \mathbf{t}]A, [\mathbf{t}]\varphi) = \\ \neg wlp([\mathbf{s}, \mathbf{t}]A, [\mathbf{t}]\neg\varphi) = \\ \\ \neg([\mathbf{s}](\textstyle\bigwedge_{i=1}^{n}[precond_i^A \rightarrow wlp(postcond_i^A, \neg\varphi)] \\ \wedge \\ ((\textstyle\bigwedge_{i=1}^{n}[\neg precond_i^A]) \rightarrow \neg\varphi))) = \\ \\ [\mathbf{s}](\textstyle\bigvee_{i=1}^{n}[precond_i^A \wedge \neg wlp(postcond_i^A, \neg\varphi)] \\ \vee \\ ((\textstyle\bigwedge_{i=1}^{n}[\neg precond_i^A]) \wedge \varphi)) \end{gathered}$$

We know that $postcond_i^A$ is a consistent conjunction of reassignments which implies (by case (2)) that

$$\neg wlp(postcond_i^A, \neg\varphi) \equiv wlp(postcond_i^A, \varphi).$$

The result follows from the fact that either exactly one of the preconditions is true, or none of them is.

□

Proposition 4.1.8 Let $[\mathbf{s},\mathbf{t}]A$ be a deterministic action. Then

$$[\mathbf{s},\mathbf{t}]A \mathrel|\!\sim wlp([\mathbf{s},\mathbf{t}]A,[\mathbf{t}]\varphi) \equiv [\mathbf{t}]\varphi,$$

for arbitrary formulae φ.
Proof: Follows immediately from propositions 4.1.5, 4.1.7, and corollary 4.1.6.□

Proposition 4.1.7 provide means for an elegant and straightforward definition of successor state axioms in PMON.

Definition 4.1.9 (PMON Successor State Axioms)
Let SCD $= \{[\mathbf{s}_1,\mathbf{t}_1]A_1,\ldots,[\mathbf{s}_n,\mathbf{t}_n]A_n\}$ be a set of schedule statements in a deterministic PMON scenario description. For each fluent, $\mathbf{f}$, the *PMON Successor State Axiom* is defined as

$$\forall t. \bigwedge_{i=1}^{n} t = \mathbf{t}_i \rightarrow (H(t,\mathbf{f}) \equiv wlp([\mathbf{s}_i,\mathbf{t}_i]A_i, H(\mathbf{t}_i,\mathbf{f})))$$

□

However, this does not solve the problem of regression for nondeterministic actions. Now we prove that PMON entailment and *wlp* computations coincide.

Proposition 4.1.10 Let $[\mathbf{s},\mathbf{t}]A$ be an admissible, possibly nondeterministic action, and $[\mathbf{t}]\varphi$ an H formula. Then

$$[\mathbf{s},\mathbf{t}]A \mathrel|\!\sim [\mathbf{t}]\varphi \qquad \text{iff} \qquad wlp([\mathbf{s},\mathbf{t}]A,[\mathbf{t}]\varphi) = \mathbf{T}$$

Proof: $\Leftarrow$) Follows immediately from Proposition 4.1.5.

$\Rightarrow$) If $wlp([\mathbf{s},\mathbf{t}]A,[\mathbf{t}]\varphi) \neq \mathbf{T}$ held, *wlp* would not yield the weakest precondition which contradicts Theorem 4.1.4. □

Proposition 4.1.10 says that *wlp* is sound and complete with respect to PMON entailment.

4.2 Regression-based Pre- and Postdiction Procedures

Using our definition fo *wlp* we can now construct regression procedures for pre- and postdiction in PMON. One single procedure for both pre- and postdiction does not exist, and we will argue that *wlp* is applicable for prediction, while *wlp** should be used for postdiction. The way in which observations are handled also differs between the two.

By prediction we mean that, given a scenario, Υ, we want to know if a formula, φ, holds *after* the actions of the scenario have been executed. For postdiction we want to know if φ holds at the initial time point of the scenario. Without loss of generality, we only consider the cases where no observations occur after the query for prediction and before the query for postdiction. We can assume that such observations occur at the same time point as the query, due to the inertia assumption.

Algorithm 4.2.1 (Prediction)
Input is a formula, $[\mathbf{t}]\varphi$, a scenario, Υ, and a time point $\mathbf{s}$, such that no observation, or starting point of an action, in Υ occurs at a time point $< \mathbf{s}$. Output is a tuple $\langle\beta, \delta\rangle$ such that $\Upsilon \mathrel{|\!\sim} [0]\beta \wedge \delta \rightarrow [\mathbf{t}]\varphi$ holds. $\beta \wedge \delta$ describes all initial states such that if the action sequence is started in either of them, the sequence terminates in a state satisfying φ.

1. $\tau := \mathbf{t}$ and $\alpha := \varphi$.
2. Repeat the following steps until a tuple is returned.
   (a) If $\tau = \mathbf{s}$ then return $\langle\alpha, [\mathbf{s}]\delta\rangle$ if such initial observation δ exists, else return $\langle\alpha, \mathbf{T}\rangle$.
   (b) If there is an observation $[\tau]\delta$ in Υ, then $\alpha := (\delta \wedge \alpha)$.
   (c) If there is an schedule statement $[\mathbf{t}, \tau]A$ in Υ, then $\alpha := wlp([\mathbf{t}, \tau]A, [\tau]\alpha)$ and $\tau := \mathbf{t}$, else $\tau := \tau - 1$ and $\alpha := [\tau - 1]\alpha[]$.

$\square$

If Υ is deterministic, then $\Upsilon \mathrel{|\!\sim} [0]\beta \equiv [\mathbf{t}]\varphi$ holds for any $\langle\beta, \delta\rangle$ returned by the algorithm.
Correctness follows from proposition 4.1.5.

Example 4.2.2 We illustrate algorithm 4.2.1 with the scenario in Example 3.1.1 slightly changed: We do not include OBS2. Our goal is to prove that $H(6, white)$ is a consequent of the scenario.

Input: $\varphi \equiv H(6, white)$, $\Upsilon =$ the changed scenario, and $\mathbf{s} = 0$.

First Iteration: $\tau = 6$ and $\alpha \equiv H(6, white)$.

Second Iteration: $\tau = 5$ and $\alpha \equiv H(5, white)$.

Third Iteration: The schedule statement $[3, \tau]Drop$ is found, and $wlp([3,5]Drop, Holds(5, white))$ is computed. After the iteration $\tau = 3$ and $\alpha \equiv \neg H(3, over\_board) \wedge H(3, white)$.

Sixth Iteration: $\tau = 0$ and $\alpha \equiv \neg H(0, over\_board) \wedge H(0, white)$.

In the sixth iteration the observation at time point 0 is considered, so

$$\langle H(0, over\_board) \wedge \neg H(0, white) \wedge \neg H(0, black), \\ \neg H(0, over\_board) \wedge H(0, white)\rangle$$

is returned by the algorithm.□

The algorithm returns a tuple $\langle \alpha, \delta \rangle$, where α is the regression result and δ is the initial observation. The conjunction of the regression result and the initial observation, $\alpha \wedge \delta$ produce, in Example 4.2.2, a contradiction, which tells us that the goal, H(6,white), is not a consequence of the scenario, since there are no initial states allowed by the initial observation that would terminate the sequence of actions in a state satisfying the query. If the conjunction is consistent and the implication $\delta \rightarrow \alpha$ is a tautology, the goal is a consequent, since this means that every state described by the initial observation will terminate the sequence of actions in a state satisfying the query. The third case, when the conjunction is consistent and the implication is neither a contradiction nor a tautology, implies that the conjunction is a *condition* at the initial time point for the scenario to entail the goal. This is of interest from a planning perspective since the algorithm generates the conditions under which a certain plan may succeed. This was recently examined in [Łukaszewicz and Madalińska-Bugaj, 1997]. Now, we summarize the possible interpretations of the returned tuple of the algorithm:

Let Υ be a scenario, φ a query, and $\langle \alpha, \delta \rangle$ the result returned from Algorithm 4.2.1, then

if $\alpha \wedge \delta$ is inconsistent, $\Upsilon \not\models \varphi$,

if $\alpha \wedge \delta$ is consistent, and

- if $\delta \rightarrow \alpha$ is a tautology, $\Upsilon \mid\!\sim \varphi$
- if $\delta \rightarrow \alpha$ is not a tautology, $\alpha \wedge \delta$ describes all initial states that would terminate Υ in a state satisfying φ.

For Example 4.2.2 we have

$$\alpha \equiv H(0, over\_board) \wedge \neg H(0, white) \wedge \neg H(0, black),$$

and

$$\delta \equiv \neg H(0, over\_board) \wedge H(0, white).$$

In this case we can note that $\alpha \wedge \delta$ is inconsistent, which implies that $\Upsilon \not\mid\!\sim \varphi$. This does not imply that $\Upsilon \mid\!\sim \not\varphi$, due to the nondeterministic actions.

For the postdiction case, we start by taking the last observation and regress back to the initial time point. We are now interested in regression results that are implied by the conditions *after* the action is performed, since the observations are parts of the axiomatization. We will therefore make use of corollary 4.1.6 for this algorithm.

Algorithm 4.2.3 (Postdiction)
Input is a formula, $[0]\varphi$, a consistent action scenario description, Υ, and a time point $\mathbf{s}$, where an observation, $[\mathbf{s}]\delta$, occurs, and no observation occurs at any time point $> \mathbf{s}$. Output is a formula β such that every initial state that is consistent with the scenario, and no other state, is a model of β.

1. $\tau := \mathbf{s}$ and $\alpha := \mathbf{T}$.
2. Repeat the following steps until a formula is returned.
   (a) If there is an observation $[\tau]\delta$ in Υ, then $\alpha := (\delta \wedge \alpha)$.
   (b) If $\tau = \mathbf{0}$ then return α.
   (c) If there is a schedule statement $[\mathbf{t}, \tau]A$ in Υ, then $\alpha := wlp^*([\mathbf{t}, \tau]A, [\tau]\alpha)$ and $\tau := \mathbf{t}$, else $\tau := \tau - 1$ and $\alpha := [\tau - 1]\alpha[]$.

□

When a formula β has been generated we can choose if we want to prove that the scenario is consistent with the initial states φ by proving that $\varphi \rightarrow \beta$ holds. If we want to prove that $\Upsilon \mid\!\sim [0]\varphi$, we prove that $\beta \rightarrow \varphi$ holds.
Correctness follows from corollary 4.1.6.
To illustrate algorithm 4.2.3, we again look at example 3.1.1. This time we

remove OBS1, and input OBS2, $H(6, white) \wedge \neg H(6, black)$, to the algorithm. Without going into details, we can note that

$$wlp^*([3,5]Drop, H(5, white) \wedge \neg H(5, black)) = \\ \neg H(3, over\_board) \wedge H(3, white) \wedge \neg H(3, black),$$

which coincide with our intuition.

The intuition behind the difference of how observations are handled by the algorithms is that for prediction we want observations to *verify* the computation results, that is to describe state sets that are subsets of the regression result, and for postdiction, the observation should filter out all states not consistent with it.

4.3 Planning in PMON

Haslum [1997] investigates planning in PMON with *wlp*. He provides a sound and complete planning algorithm with an implementation for linear planning, and number of planning heuristics for non-linear planning. Not surprisingly, it turns out that the planning problem, given the version of PMON used in this thesis, is PSPACE-hard. More interestingly, if we allow nested actions (that is implications inside postconditions) we have EXPSPACE-completeness. We conjecture that even the formalism in this thesis provide EXPSPACE-completeness for the planning problem, but we have no such proof yet.

Chapter 5

Tractability

The lack of complexity results for reasoning about action and change will now be remedied. We will develop an expressive logic that is more suited for the analysis than for example PMON or SitCalc. Thus, the results in this chapter include not only the complexity results, but also the logic, in which we address continuous time, concurrency, and memory of actions. The construction, and intuitiveness, of the proposed logic rest on two assumptions:

1. Actions always succeed. This is the *action omniscience assumption.*

2. Feature values change if and only if an action explicitly changes them. This is the *inertia assumption.*

We argue that for scenarios where these two assumptions hold, the frame problem is solved.

To facilitate the reading of the somewhat technical material, we begin with an informal overview of the results of the chapter.

5.1 Overview

In Section 5.2 we develop a temporal logic, Λ, which is syntactically related to the propositional temporal logic TPTL [Alur and Henzinger, 1994], but without the tense modal operators (the dynamic behaviors will be handled by *action expressions*).

The temporal domain is the set of real numbers and temporal expressions are based on relations $=$, $\leq$, $<$, $\geq$ and $>$ between linear polynomials with rational coefficients over a set of temporal variables. The semantics of this temporal logic is standard. It can be viewed as propositional interpretations (states) on a time line.

The formalism for reasoning about action is *narrative* based, which means that *scenario descriptions* (similar to those of PMON) are used to model the real world. Scenario descriptions consist of formulae in the temporal logic (*observations*) and *action expressions* which are constructs that state that certain changes in values of the *features* (propositions, fluents) may occur. We write action expressions as $\pi \Rightarrow [\alpha]\epsilon\,\texttt{Infl}$, where π is the precondition for the action, ϵ the effects, α a temporal expression denoting *when* the effects are taking place, and `Infl` is the set of all features that are influenced by the action. The influenced features are not subject to the assumption of inertia, i.e. we allow them, and only them, to change during the execution of the action.

It turns out that deciding satisfiability is NP-complete, both for the temporal logic and the scenario descriptions. Interestingly, the problem is NP-complete for scenario descriptions that only include Horn clause observations, unconditional and unary action expressions (this terminology will be explained later), and no stated disjunctive relations between temporal expressions.

To extract a tractable subset from our formalism we rely on a recent result in *temporal constraint reasoning* by Jonsson and Bäckström in [1996] (also discovered independently by Koubarakis [1996]). They have identified a large tractable class of temporal constraint reasoning, using *Horn Disjunctive Linear Relations* (Horn DLRs) which are relations between linear polynomials with rational coefficients. We make use of their result by restricting formulae in our scenario descriptions to be Horn and then by encoding scenario descriptions into Horn DLRs. For the temporal logic this is fairly straightforward. For the scenario descriptions, it turns out that we have to put some constraints on the temporal relations and actions in the scenario descriptions.

We will use the following two examples: Jump into a Lake with a Hat [Giunchiglia and Lifschitz, 1995] and Soup Bowl Lifting [Gelfond *et al.*, 1991]. Below we informally describe the examples.

Example 5.1.1 (Jump into a Lake with a Hat, JLH)
If you jump into the lake you will get wet. If you have been in the water at some time point it is unclear if you still have your hat on. This is an example of *nondeterminism* and of *memory of actions.* □

Example 5.1.2 (Soup Bowl Lifting, SBL)

If we lift either side of a soup bowl at some time points, the content will be spilled, unless we lift both sides at the same time point. This is an example of *concurrency.* □

The first example stated above can be handled by the tractable subset of our formalism, while the other cannot.

5.2 Scenario Descriptions

We introduce a semantics that is a simpler variant of Sandewall's Features and Fluents Framework [Sandewall, 1994], in that the effects of an action can only occur at one and the same time point for a given action, and we use only propositional values of features (similar to the work of Doherty [1994]) However, in some respects this formalism is more flexible than Sandewall's: we use a *continuous* time domain, we allow *concurrently* executing actions, and effects of actions can depend on other states in the history than the state at the *starting time point* of the action (this implies *memory* of actions, in Sandewall's [1994] terminology). One example of a formalism having memory is that of Gustafsson and Doherty [1996].

Initially, a basic temporal logic is defined. The computational properties of this logic will be exploited by the scenario description logic, i.e. ultimately (in Section 5.3) the scenario descriptions will be transformed into formulae of the basic temporal logic.

5.2.1 Syntax

We begin by defining the basic temporal logic.

We assume that we have a set $\mathcal{T}$ of time point variables intended to take real values, and a set $\mathcal{F}$ of *features* intended to take propositional values.

Definition 5.2.1 A *signature* is a tuple $\sigma = \langle \mathcal{T}, \mathcal{F} \rangle$, where $\mathcal{T}$ is a finite set of time point variables and $\mathcal{F}$ is a finite set of propositional features. A *time point expression* is a linear polynomial over $\mathcal{T}$ with rational coefficients. We denote the set of time point expressions over $\mathcal{T}$ by $\mathcal{T}^*$. □

We could, for example, use the signature $\langle \mathcal{T}, \mathcal{F} \rangle$, where $\mathcal{T} = \{\mathbf{c}_1, \mathbf{c}_2\}$ and $\mathcal{F} = \{hat\_on, dry, on\_land\}$, to represent JLH. Then $\mathbf{c}_1$ would be the time point when the person jumped into the lake, and $\mathbf{c}_2$ the time point when the status of the hat is examined, with the assumption $\mathbf{c}_1 \leq \mathbf{c}_2$.

Definition 5.2.2 Let $\sigma = \langle \mathcal{T}, \mathcal{F} \rangle$ be a signature, let $\alpha, \beta \in \mathcal{T}^*$, $f \in \mathcal{F}$, $R \in \{=, \leq, <, \geq, >\}$, $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, and define the *scenario description language* Λ over σ by

$$\Lambda ::= \mathbf{T} \mid \mathbf{F} \mid f \mid \alpha R \beta \mid \neg\Lambda \mid \Lambda_1 \oplus \Lambda_2 \mid [\alpha]\Lambda.$$

A formula of the form $\alpha R \beta$ is a *linear relation*, and one that does not contain any connectives (any of the constructs $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$ and $[\cdot]$) is *atomic.* If γ is atomic and $\alpha \in \mathcal{T}^*$, then the formulae γ, $[\alpha]\gamma$, $\neg\gamma$, $[\alpha]\neg\gamma$ and $\neg[\alpha]\gamma$ are *literals.* (A formula $[\alpha]\gamma$ expresses that at time α, γ is true.) A literal l is *negative* iff it contains $\neg$ and its corresponding atomic formula γ is not of the form $\alpha R \beta$ for $R \in \{<, \leq, >, \geq\}$. A literal that is not negative is *positive.* Disjunctions of literals are *clauses.* A formula $\gamma \in \Lambda$ is in *conjunctive normal form*, CNF, iff it is a conjunction of clauses. A formula γ is *Horn* iff it is a clause with at most one positive literal. A set Γ of formulae is *Horn* iff every $\gamma \in \Gamma$ is Horn. Syntactical identity between formulae is written $\equiv$, and when ambiguity is to be avoided, we denote formulae $\gamma \in \Lambda$ by $\ulcorner\gamma\urcorner$.

Let γ be a formula. A feature $f \in \mathcal{F}$ occurs *free* in γ iff it does not occur within the scope of a $[\alpha]$ expression in γ. $\alpha \in \mathcal{T}^*$ *binds* f in γ if a formula $[\alpha]\phi$ occurs as a subformula of γ, and f is free in ϕ. If no feature occurs free in γ, γ is *closed.* If γ does not contain any occurrence of $[\alpha]$ for any $\alpha \in \mathcal{T}^*$, then γ is *propositional.* □

For JHL we make some observations: $[0]\mathit{hat\_on} \wedge \mathit{dry} \wedge \mathit{on\_land}$ that denotes that, initially, the hat is on, the person is dry and not in the water, and $\mathbf{c}_1 \geq 0 \wedge \mathbf{c}_1 \leq \mathbf{c}_2$. Note that both the observations are Horn.
Using Λ, we can thus express propositions being true at time points, and express relations between time points. Next we define the extension of the basic temporal logic by introducing *action expressions*, i.e. constructs that enable modelling of change. This extension will be referred to as Λ'

Definition 5.2.3 Let $\sigma = \langle \mathcal{T}, \mathcal{F} \rangle$ be a signature. An *action expression* over σ is a tuple $A = \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle$, $\alpha \in \mathcal{T}^*$, π a closed formula in Λ, $\mathtt{Infl} \subseteq \mathcal{F}$, and ϵ a propositional formula, where all features occurring in ϵ are in $\mathtt{Infl}$. α is the *result time point* of A, π is the *precondition* of A, $\mathtt{Infl}$ is the set of *influenced features* of A, and ϵ is the *effects* of A. A is *unconditional* iff $\pi \equiv \mathbf{T}$, and *unary* iff $|\mathtt{Infl}| = 1$.

For convenience, we write action expressions as $\pi \Rrightarrow [\alpha]\epsilon\,\mathtt{Infl}$; for example we have $[3]\mathit{loaded} \Rrightarrow [4]\neg\mathit{alive}\{\mathit{alive}\}$ that the denotes that if a gun is loaded at time point 3, then a turkey will not be alive at time point 4, for an action shoot. If $\pi \equiv \mathbf{T}$, we remove it and the $\Rrightarrow$ symbol, and if $\epsilon \equiv \mathbf{T}$, we remove it. An example is an unconditional loading action $[2] loaded\{loaded\}$, and the action of spinning the chamber of a gun $[3]\{loaded\}$.

An *observation* over σ is a closed formula in Λ. □

It might be of interest to note that we do not confine the actions to be "Markovian", i.e. to depend only on the state in which they are performed. They may depend on multiple states in the past, or even on states in the future (even if it is quite unlikely that that would occur in a real-world scenario). Next, we combine the concepts defined so far into one.

Definition 5.2.4 A *scenario description* is a tuple $\Upsilon = \langle \sigma, \text{SCD}, \text{OBS} \rangle$, where $\sigma = \langle \mathcal{T}, \mathcal{F} \rangle$ is a signature, SCD (the *schedule*) is a finite set of action expressions over σ, and OBS is a finite set of observations over σ. The *size* of a scenario description is defined as the sum of lengths of all formulae in SCD and OBS. □

Now, we formalise the examples from Section 5.1.

Example 5.2.5 (JLH)
The intended conclusion of the following scenario is that the person is wet at time $\mathbf{c}_1$, and we do not know if the hat is on at time point $\mathbf{c}_2$, occurring after the person jumps.

OBS1	$[0] hat\_on \land dry \land on\_land$
SCD1	$[\mathbf{c}_1] \neg on\_land\{on\_land\}$
SCD2	$[\mathbf{c}_1] \neg on\_land \Rrightarrow [\mathbf{c}_1] \neg dry\{dry\}$
SCD3	$[\mathbf{c}_1] \neg on\_land \Rrightarrow [\mathbf{c}_2]\{hat\_on\}$
OBS4	$\mathbf{c}_1 \geq 0 \land \mathbf{c}_2 \geq \mathbf{c}_1$ □

Example 5.2.6 (SBL)
We have two actions: one for lifting the left side of the soup bowl and one for lifting the right side. If the actions are not executed simultaneously, the tablecloth will no longer be dry. The intended conclusion here, is $\mathbf{c}_2 = \mathbf{c}_1$.

OBS1	$[0] dry$
SCD1	$[\mathbf{c}_1] leftup\{leftup\}$
SCD2	$[\mathbf{c}_1] \neg rightup \Rrightarrow [\mathbf{c}_1] \neg dry\{dry\}$
SCD3	$[\mathbf{c}_2] rightup\{rightup\}$
SCD4	$[\mathbf{c}_2] \neg leftup \Rrightarrow [\mathbf{c}_2] \neg dry\{dry\}$
OBS2	$[\mathbf{c}_2] dry$
OBS3	$\mathbf{c}_2 > 0 \land \mathbf{c}_1 > 0$ □

5.2.2 Semantics

For the presentation of the semantics we proceed in a similar way to the presentation of the syntax. We begin by defining the semantics of the basic temporal logic.

Definition 5.2.7 Let $\sigma = \langle \mathcal{T}, \mathcal{F} \rangle$ be a signature. A *state* over σ is a function from $\mathcal{F}$ to the set $\{\mathbf{T}, \mathbf{F}\}$ of truth values. A *history* over σ is a function h from $\mathbb{R}$ to the set of states. A *valuation* ϕ is a function from $\mathcal{T}$ to $\mathbb{R}$. It is extended in a natural way (as a homomorphism from $\mathcal{T}^*$ to $\mathbb{R}$), giving for instance $\phi(3t + 4.3) = 3\phi(t) + 4.3$. A *development*, or *interpretation*, over σ is a tuple $\langle h, \phi \rangle$ where h is a history and ϕ is a valuation. □

We will now define the notion of *model* for closed formulae in Γ in a classical way.

Definition 5.2.8 Let $\gamma \in \Lambda$, and let $D = \langle h, \phi \rangle$ be a development. Define the *truth value* of γ in D for a time point $t \in \mathbb{R}$, denoted $D(\gamma, t)$, as follows (here we overload $\mathbf{T}$ and $\mathbf{F}$ to denote both formulae and truth values). Assume $f \in \mathcal{F}$, $R \in \{=, \leq, <, \geq, >\}$, $\alpha, \beta \in \mathcal{T}^*$, $\gamma, \delta \in \Lambda$, $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, and $\tau \in \{\mathbf{T}, \mathbf{F}\}$. Now define

$$
\begin{aligned}
&D(\tau, t) = \tau \\
&D(f, t) = h(t)(f) \\
&D(\alpha R \beta, t) = \phi(\alpha) R \phi(\beta) \\
&D(\neg\gamma, t) = \neg D(\gamma, t) \\
&D(\gamma \oplus \delta, t) = D(\gamma, t) \oplus D(\delta, t) \\
&D([\alpha]\gamma, t) = D(\gamma, \phi(\alpha)).
\end{aligned}
$$

Two formulae γ_1 and γ_2 are *equivalent* iff $D(\gamma_1, t) = D(\gamma_2, t)$ for all D and t. A set $\Gamma \subseteq \Lambda$ of formulae is *satisfiable* iff there exists a development D and a time point $t \in \mathbb{R}$ such that $D(\gamma, t)$ is true for every $\gamma \in \Gamma$. A development D is a *model* of a set $\Gamma \subseteq \Lambda$ of closed formulae iff $D(\gamma, t)$ is true for every $t \in \mathbb{R}$ and $\gamma \in \Gamma$. □

Fact 5.2.9 For $\gamma \in \Lambda$ and $\alpha \in \mathcal{T}^*$, $\neg[\alpha]\gamma$ is equivalent to $[\alpha]\neg\gamma$. For a closed formula γ, $D(\gamma, t) = D(\gamma, t')$ for any $t, t' \in \mathbb{R}$ and development D. □

Thus, if γ is closed, we can write $D(\gamma)$ instead of $D(\gamma, t)$.

Now we define the semantics of the action expressions based on models for the basic temporal logic. Inertia (the frame problem) is handled by identifying all time points where a feature f can possibly change its value.

Then during every interval where no such change time point exists, f has to have the same value throughout the interval.

Definition 5.2.10 Let $D = \langle h, \phi \rangle$ be a development. An action expression $A = \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle$ *has effects* in D iff $D(\pi) = \mathbf{T}$, that is, whenever the preconditions of A are true in the model. Let $f \in \mathcal{F}$, and define $Chg(D, \textsc{scd}, f, t)$ to be true for a time point $t \in \mathbb{R}$ iff $f \in \mathtt{Infl}$ for some action expression $A = \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd}$ that has effects in D, with $\phi(\alpha) = t$. Note that $Chg(D, \textsc{scd}, f, t)$ can only be true for a finite number of time points for fixed SCD and f. Now we have defined the time points where the fluents can change value.

Let $\Upsilon = \langle \sigma, \textsc{scd}, \textsc{obs} \rangle$ be a scenario description. An *intended model* of Υ is a development $D = \langle h, \phi \rangle$ where

- D is a model of OBS
- For each $A = \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd}$ that has effects in D, $D(\epsilon, \phi(\alpha)) = \mathbf{T}$
- For each $f \in \mathcal{F}$ and $s, t \in \mathbb{R}$ where $s < t$ such that for no $t' \in (s, t)$ (open interval), $Chg(D, \textsc{scd}, f, t')$ holds, we have $h(t')(f) = h(s)(f)$ for every $t' \in (s, t)$. Intuitively, this definition insures that no change in the value of a feature occurs in an interval if no action explicitly changes it.

Denote by $Mod(\Upsilon)$ the set of all intended models for a scenario description Υ.

A formula $\gamma \in \Lambda$ is *entailed* by a scenario description Υ, denoted $\Upsilon \models \gamma$, iff γ is true in all intended models of Υ. Υ is *satisfiable* iff $Mod(\Upsilon) \neq \emptyset$. □

Fact 5.2.11 If $\Upsilon = \langle \sigma, \textsc{scd}, \textsc{obs} \rangle$ is a scenario description and $\gamma \in \Lambda$ a formula, then $\Upsilon \models \gamma$ iff $\langle \sigma, \textsc{scd}, \textsc{obs} \cup \{\neg\gamma\} \rangle$ is unsatisfiable. □

We comment on how this is used in our two examples:

- For JHL we can note that every intended model $D = \langle h, \phi \rangle$ has the following properties:
    1. $h(\phi(0))(hat\_on) = \mathbf{T}$, $h(\phi(0))(dry) = \mathbf{T}$, and $h(\phi(0))(on\_land) = \mathbf{T}$, due to OBS1.
    2. $\phi(\mathbf{c}_1) \geq \phi(0)$ and $\phi(\mathbf{c}_2) \geq \phi(\mathbf{c}_1)$, due to OBS4.

3. Since all actions have effect we can note, for instance, that $D(dry, \phi(\mathbf{c}_1)) = \mathbf{F}$, since SCD1 sets dry to $\mathbf{F}$.
4. Since D is an intended model of JHL, we know that all three actions have effects in D. Thus, the following Chg extensions are true:
   - $Chg(D, \text{SCD}, on\_land, \phi(\mathbf{c}_1))$ due to SCD1,
   - $Chg(D, \text{SCD}, dry, \phi(\mathbf{c}_1))$ due to SCD2, and
   - $Chg(D, \text{SCD}, hat\_on, \phi(\mathbf{c}_2))$ due to SCD2.

   Intuitively, all features are allowed to change iff they are influenced by an action.

Property (3) ensures that if we add the observation $[\mathbf{c}_1]dry$ the scenario would be unsatisfiable. So JHL entails $[\mathbf{c}_1]\neg dry$. On the other hand, if instead we added $[\mathbf{c}_2]hat\_on$ or $[\mathbf{c}_2]\neg hat\_on$ to the scenario, we would not get unsatisfiability, since SCD3 split the set of models into those where $[\mathbf{c}_2]hat\_on$ is true, and those where it is not. Thus neither of the expressions are logical consequences of the scenario.

- For SBL, adding $\mathbf{c}_2 \neq \mathbf{c}_1$ as an observation will make the scenario unsatisfiable.

5.3 Complexity Results

5.3.1 Basic Results

It is no surprise that deciding satisfiability for the basic temporal logic is NP-hard. Proofs of NP-completeness, on the other hand, depend on the tractability results.

Proposition 5.3.1 Deciding satisfiability of a set $\Gamma \subseteq \Lambda$ is NP-hard.
Proof: Propositional logic is a subset of Λ. $\Box$

Corollary 5.3.2 Deciding whether a scenario description is satisfiable is NP-hard. $\Box$

That these problems are in NP, and thus are NP-complete, is proved in Theorem 5.3.9 and Theorem 5.3.10.

Interestingly, we can strengthen the result considerably.

Theorem 5.3.3 Deciding whether a scenario description is satisfiable is NP-hard, even if action expressions are unconditional and unary, only Horn observations are allowed, and no disjunctive relations between time points may be stated.

Proof: Reduction from 3SAT. Let $C_i = l_{i1} \vee l_{i2} \vee l_{i3}$ be the clauses of a 3CNF formula ϕ, and v the set of propositional symbols used in ϕ. We construct a scenario description Υ satisfying the required restrictions as follows. Set $\mathcal{F} = v \cup \{p' | p \in v\}$, $\mathcal{T} = \{s, t\} \cup \{t_p | p \in v\}$,

$$\textsc{scd} = \{\langle t, \mathbf{T}, \{p\}, \neg p\rangle | p \in v\} \cup \{\langle t, \mathbf{T}, \{p'\}, p'\rangle | p \in v\},$$

$$\begin{aligned} \textsc{obs} = \; & \{t > s\} \cup \{[s]p, [s]\neg p' | p \in v\} \cup \\ & \{\mathcal{C}(l_{i1}) \vee \mathcal{C}(l_{i2}) \vee \mathcal{C}(l_{i3})\}, \end{aligned}$$

where $\mathcal{C}(p) = \neg[t_p]p'$ and $\mathcal{C}(\neg p) = \neg[t_p]p$.

We shall use intended models D as models for ϕ, interpreting $D(t_p < t)$ as the truth value of p, and so show that Υ is satisfiable iff ϕ is satisfiable. First note the facts that in any $D \in \mathit{Mod}(\Upsilon)$, $D(\neg[t_p]p') = D([t_p]p)$, and that $D([t_p]p) = D(t_p < t)$.

$\Rightarrow$) Suppose $D \in \mathit{Mod}(\Upsilon)$, and consider a clause $C_i = l_{i1} \vee l_{i2} \vee l_{i3}$ in ϕ. By the construction of OBS, one of $\mathcal{C}(l_{i1})$, $\mathcal{C}(l_{i2})$ or $\mathcal{C}(l_{i3})$ has to be true in D, say $l = l_{i1}$. If $l = p$, then $\mathcal{C}(l) = \neg[t_p]p'$; thus $D([t_p]p)$ is true, and so is $D(t_p < t)$. If $l = \neg p$, then $\mathcal{C}(l) = \neg[t_p]p$; thus $D([t_p]p)$ is false, and so is $D(t_p < t)$.

$\Leftarrow$) Suppose ϕ has a propositional model M, and consider a clause $C_i = l_{i1} \vee l_{i2} \vee l_{i3}$ in ϕ. Construct an intended model D of Υ by letting features have values as forced by the scenario, t having the value 0 and s the value -1, and for each $p \in v$, if p is true in M, then set $t_p = -1$, and otherwise $t_p = 1$. It is clear that the expression $\mathcal{C}(l_{i1}) \vee \mathcal{C}(l_{i2}) \vee \mathcal{C}(l_{i3})$ is true in D. The result follows, since it is clear that the reduction is polynomial. □

We now present the key to tractability, which is a linear-programming approach to temporal constraint reasoning, by Jonsson and Bäckström [1996].

Definition 5.3.4 Let α and β be linear polynomials with rational coefficients over some set X of variables. Then a *disjunctive linear relation*, DLR, is a disjunction of one or more expressions of the form $\alpha = \beta$, $\alpha \neq \beta$, $\alpha \leq \beta$, $\alpha < \beta$. A DLR is *Horn* iff it contains at most one disjunct with the relation $=$, $<$ or $\leq$.

An assigment m of variables in X to real numbers is a *model* of a set Γ of DLRs iff all formulae in Γ are true when taking the values of variables in the DLRs. A set of DLRs is *satisfiable* iff it has a model. □

The following result is the main result of Jonsson and Bäckström [1996].

Proposition 5.3.5 Deciding satisfiability of a set of Horn DLRs is polynomial. □

Now we restrict the scenario description language and the form of actions. Furthermore, a structural restriction on scenario descriptions, verifiable in polynomial time, is imposed. We shall define an encoding function that takes a Horn scenario description Υ and returns a set Γ of Horn DLRs such that Γ is satisfiable iff Υ is satisfiable.

5.3.2 Satisfiability of Horn Formulae is Tractable

First, we code Horn formulae as Horn DLRs.

Definition 5.3.6 Let l be a closed literal, and assume the existence of fresh, unique time point variables t_f^α for each $f \in \mathcal{F}$ and $\alpha \in \mathcal{T}^*$. Then $C(l)$ is defined as follows, assuming $R \in \{=, \leq, <, \geq, >\}$ and $f \in \mathcal{F}$.

$C(\mathbf{T}) = \ulcorner 0 = 0 \urcorner$
$C(\mathbf{F}) = \ulcorner 0 \neq 0 \urcorner$
$C(\alpha R \beta) = \ulcorner \alpha R \beta \urcorner$
$C(\neg \alpha = \beta) = \ulcorner \alpha \neq \beta \urcorner$
$C(\neg \alpha \leq \beta) = \ulcorner \alpha > \beta \urcorner$
$C(\neg \alpha < \beta) = \ulcorner \alpha \geq \beta \urcorner$
$C(\neg \alpha \geq \beta) = \ulcorner \alpha < \beta \urcorner$
$C(\neg \alpha > \beta) = \ulcorner \alpha \leq \beta \urcorner$
$C([\eta]\alpha R \beta) = C(\alpha R \beta)$
$C(\neg[\alpha]\gamma) = C([\alpha]\neg\gamma)$
$C([\alpha]f) = \ulcorner t_f^\alpha = 0 \urcorner$
$C([\alpha]\neg f) = \ulcorner t_f^\alpha \neq 0 \urcorner$

The last two parts are the key to the transformation.
Let $\gamma \in \Lambda$ be a closed Horn formula, and let γ' be obtained from γ by simplifying away occurrences of $\mathbf{T}$ and $\mathbf{F}$. Now $\gamma' = \bigvee_i l_i$. Then define $C(\gamma)$ to be the DLR $\delta = \bigvee_i C(l_i)$. Note that δ is always a Horn DLR.

Let $\Gamma \subseteq \Lambda$ be a set of closed Horn formulae, and T the set of all time point expressions occurring in Γ. Then $C(\Gamma)$ is defined by

$$\begin{aligned} C(\Gamma) = \{&C(\gamma) | \gamma \in \Gamma\} \cup \\ &\{C(\neg[\alpha]f \vee \beta \neq \alpha \vee [\beta]f) | f \in \mathcal{F}, \alpha, \beta \in T\}. \end{aligned}$$

The second set is called the *correspondence equations*. Note that the argument of C in a correspondence equation is equivalent to $[\alpha]f \wedge \beta = \alpha \rightarrow [\beta]f$. $\Box$

The following result is crucial.

Theorem 5.3.7 Let $\Gamma \subseteq \Lambda$ be a set of closed Horn formulae. Then Γ is satisfiable iff $C(\Gamma)$ is satisfiable.
Proof: Let T be the set of time point expressions occurring in Γ.
$\Rightarrow$) Let $D = \langle h, \phi \rangle$ be a model of Γ. We shall construct a model m of $C(\Gamma)$. First set $m(t) = \phi(t)$ for all $t \in \mathcal{T}$. Now all temporal relations from Γ are satisfied in m, since they are directly transferred. For each $f \in \mathcal{F}$ and $\alpha \in T$, if $h(\phi(\alpha))(f)$ is true, then set $m(t_f^\alpha) = 0$, otherwise set $m(t_f^\alpha) = 1$. It is clear that the correspondence equations are satisfied by this definition, and so are the remaining elements of $C(\Gamma)$.
$\Leftarrow$) Let m be a model of $C(\Gamma)$, Construct an interpretation $D = \langle h, \phi \rangle$ as follows. First set $\phi(t) = m(t)$ for all $t \in \mathcal{T}$. It is enough to determine h for values $\phi(\alpha)$, $\alpha \in T$, since we have no restrictions on other values. Set $h(\phi(\alpha))(f)$ to be true iff $m(t_f^\alpha) = 0$. That h is well defined follows directly from the correspondence equations which hold in m. $\Box$

Corollary 5.3.8 Deciding satisfiability of sets of closed Horn formulae is polynomial.
Proof: It is clear that the transformation C is polynomial. The result follows from Proposition 5.3.5. $\Box$

Now we have the results for the proofs of membership in NP for the satisfiability problems of Λ and of scenario descriptions. Proofs (and auxillary definitions) of the following two theorems can be found in Appendix A.

Theorem 5.3.9 Deciding satisfiability of a set $\Gamma \subseteq \Lambda$ is NP-complete. $\Box$

Theorem 5.3.10 Deciding whether a scenario description is satisfiable is NP-complete. $\Box$

5.3.3 Tractable Scenario Descriptions

Using Corollary 5.3.8, we see that if we can code scenario descriptions into sets of Horn formulae, we will have a polynomial algorithm for reasoning with scenario descriptions. In order to obtain such a result, we need to restrict what scenario descriptions are allowed.

The strategy can briefly be described as follows: we identify all observation time points which bind a feature value and all time points where an action expression can possibly change a feature value. Then we connect bound literals with biconditionals between time points where the literal value should not change. For example, if some action expression changes the value of the feature f at time point α, there exists a $\gamma \in \textsc{obs}$ which binds f at a time point β, $\alpha < \beta$, and no changes of the value of f occurs between α and β, then $[\alpha]f \leftrightarrow [\beta]f$ should be added to the theory. This formula can be rewritten in Horn form. The example represents one of the six cases (case 3). The other cases are similar.

There are basically two restrictions: First we will have to represent action expressions as Horn formulae (*restricted* action expressions). Second, the scenario descriptions must be *ordered*, for example, we could not remove the restriction $\alpha < \beta$ in the example above.

Definition 5.3.11 Let $\Upsilon = \langle \sigma, \textsc{scd}, \textsc{obs} \rangle$ be a scenario description. For each $f \in \mathcal{F}$, define

$$E_f = \{\alpha | \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd} \wedge f \in \mathtt{Infl}\}$$

and

$$C_f = \{\alpha | \alpha \ \textit{binds} \ f \ \textit{in} \ \gamma \wedge \gamma \in O\},$$

for $O = \textsc{obs} \cup \{\pi | \langle \alpha', \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd}\}$.

E_f is *ordered* iff for $\alpha, \beta \in E_f$, exactly one of $\alpha < \beta$, $\alpha = \beta$ and $\alpha > \beta$ is consistent with[1] OBS. For $\alpha, \beta \in E_f$, E_f ordered, we define the following.

- $\alpha \prec_f \beta$ iff $\alpha < \beta$ is consistent with OBS, and for every $\gamma \in E_f$, $\alpha < \gamma < \beta$ is inconsistent with OBS,
- $-\infty \prec_f \alpha$ iff for no $\beta \in E_f$, $\beta < \alpha$ is consistent with OBS, and
- $\alpha \prec_f \infty$ iff for no $\beta \in E_f$, $\alpha < \beta$ is consistent with OBS.

Let $\alpha \in E_f$, $\beta \in \mathcal{T}^*$, and define the following.

- $\alpha \ll_f \beta$ iff $\alpha \leq \beta$ is consistent with OBS, $\alpha > \beta$ is inconsistent with OBS, and for every $\gamma \in E_f$, $\alpha < \gamma \leq \beta$ is inconsistent with OBS,
- $-\infty \ll_f \beta$ iff for every $\alpha \in E_f$ $\alpha \leq \beta$ is inconsistent with OBS.

[1]A formula γ is consistent with a set Γ iff $\Gamma \cup \{\gamma\}$ is satisfiable.

If for all $f \in \mathcal{F}$, E_f is ordered, and for all $\omega \in C_f$, $\alpha \ll_f \omega$ for some $\alpha \in E_f \cup \{-\infty\}$, then Υ is *ordered.* The last condition says that for each observation of a feature f, there is a unique change of f which sets its value, or it precedes all changes of f. $\Box$

For the JLH and SBL examples we have the following:

- For JLH we have

$$\begin{aligned} &E_{hat\_on} = \{\mathbf{c}_2\}, C_{hat\_on} = \{0\} \\ &E_{dry} = \{\mathbf{c}_1\}, C_{dry} = \{0\} \\ &E_{on\_land} = \{\mathbf{c}_1\}, C_{on\_land} = \{0, \mathbf{c}_1\} \end{aligned}$$

  We can easily verify that JLH is ordered.

- For SBL we have

$$\begin{aligned} &E_{dry} = \{\mathbf{c}_1, \mathbf{c}_2\}, C_{dry} = \{0, \mathbf{c}_2\} \\ &E_{leftup} = \{\mathbf{c}_1\}, C_{leftup} = \{\mathbf{c}_2\} \\ &E_{rightup} = \{\mathbf{c}_2\}, C_{rightup} = \{\mathbf{c}_1\} \end{aligned}$$

  SBL is not ordered since E_{dry} is not.

The orderedness of Υ will be required for us to be able to connect feature statements at different time points by biconditionals, which are required to get a Horn theory. If an action changes the value of a feature, f, at time point $\mathbf{c}$ and we have an observation at time point $\mathbf{c}+1$, e.g. $[\mathbf{c}+1]f$, and no actions have effects between $\mathbf{c}$ and $\mathbf{c}+1$ we can add the formulas $[\mathbf{c}+1]f \vee \neg[\mathbf{c}]f$ and $[\mathbf{c}]f \vee \neg[\mathbf{c}+1]f$ to the theory. Without the total ordering this would be impossible. This is the reason why the SBL example does not belong to the tractable class.

Proposition 5.3.12 Testing if a scenario description Υ is ordered is polynomial, if OBS is Horn.
Proof: For the orderedness of E_f, check for each feature that all pairs of result time points satisfy the condition. Since OBS is Horn, Corollary 5.3.8 guarantees that this can be done in polynomial time. The check for $\ll_f$ is polynomial in the same way. $\Box$

Definition 5.3.13 Let $A = \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle$ be an action expression. Then A is *restricted* iff either of the following holds:

- π is $\mathbf{T}$ and ϵ is a conjunction of propositional Horn formulae
- π is a disjunction of negative literals, and ϵ is either $\mathbf{T}$, or a conjunction of negative literals.

An ordered scenario description $\Upsilon = \langle \sigma, \text{SCD}, \text{OBS} \rangle$ is *restricted* iff every action expression in SCD is restricted and OBS is Horn. A restricted scenario description is *normal* iff it is ordered, and for every action expression $A \in \text{SCD}$, either of the following holds:

- π is $\mathbf{T}$ and ϵ is a propositional Horn formula
- π is a negative literal and ϵ is either $\mathbf{T}$ or a negative literal.

□

Both the examples previously stated (Example 5.2.5 and Example 5.2.6) are restricted, which is easy to verify.

The following result will make the forthcoming proofs easier.

Proposition 5.3.14 Let Υ be a restricted scenario description. Then we can in polynomial time construct an equivalent normal scenario description Υ'.

Proof: We note that a restricted action expression $A = \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle \in \text{SCD}$ where $\pi \equiv \bigvee_i \neg l_i$ is equivalent to replacing it by one action expression $\langle \alpha, \neg l_i, \mathtt{Infl}, \epsilon \rangle$ for each disjunct in π. Similarly, a restricted action expression $A = \langle \alpha, \pi, \mathtt{Infl}, \bigwedge_i \phi_i \rangle$ can be split into action expressions A_i by $A_i = \langle \alpha, \pi, \mathtt{Infl}, \phi_i \rangle$ (or even easier if $\epsilon \equiv \mathbf{T}$), and we obtain the same set of intended models. The transformation is clearly polynomial.

Orderedness is preserved, since we do not change the order in which features are changed. □

Thus, we can assume that our restricted scenario descriptions are normal. Next, we define the function Φ which transforms scenario descriptions into sets of Horn formulae.

Definition 5.3.15 First let $A = \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle$ be a normal action expression. There are three cases for Φ:

- If $\pi \equiv \mathbf{T}$ and $\epsilon = \bigvee_j l_j$ propositional Horn, then define $\Phi(A) = \{\bigvee_j [\alpha] l_j\}$

- If $\pi \equiv \neg l$ and $\epsilon \equiv \mathbf{T}$, then define $\Phi(A) = \emptyset$
- If $\pi \equiv \neg l$ and $\epsilon \equiv \neg m$, then define $\Phi(A) = \{l \vee [\alpha]\neg m\}$.

The restriction to normal action expressions should be clear; it implies that $\Phi(A)$ is Horn. For a set $\mathcal{S}$ of action expressions, define $\Phi(\mathcal{S}) = \bigcup_{A \in \mathcal{S}} \Phi(A)$.

Let $\Upsilon = \langle \sigma, \textsc{scd}, \textsc{obs} \rangle$ be a restricted scenario description with $\sigma = \langle \mathcal{T}, \mathcal{F} \rangle$, and without loss of generality, assume that each feature in $\mathcal{F}$ occurs in SCD or OBS. Also let b be a fresh time point variable (b standing for "beginning"), and for each $f \in \mathcal{F}$, add a new feature f'. A few construction steps (basically corresponding to the possible relations between time points in E_f and C_f) are necessary. We provide the intuitions behind the constructions as we present them.

1. $\Gamma_1 = \textsc{obs} \cup \Phi(\textsc{scd}) \cup \{b < \alpha | \alpha \in E_f \cup C_f, f \in \mathcal{F}\}$.
   The observations, the transformed action expressions, and an initial time point are added.

2. $\Gamma_2 = \{\neg[\alpha]f \vee [b]f, [\alpha]f \vee \neg[b]f | f \in \mathcal{F}, \alpha \in C_f, -\infty \ll_f \alpha\}$.
   No action expression influences f before α where it is bound by an observation; therefore f should have the same value at b as at α. Note that the members of Γ_2 are equivalent to $[\alpha]f \leftrightarrow [b]f$.

3. $\Gamma_3 = \{\neg[\alpha]f \vee [\beta]f, [\alpha]f \vee \neg[\beta]f | f \in \mathcal{F}, \alpha \in E_f, \beta \in C_f, \alpha \ll_f \beta\}$.
   f is influenced at α and bound by an observation at a later time point β. No actions have effects between α and β; therefore f should have the same value at β as it had at α.

4. $\Gamma_4 = \{\neg[\alpha]f' \vee [b]f, [\alpha]f' \vee \neg[b]f | f \in \mathcal{F}, \alpha \in E_f, -\infty \prec_f \alpha\}$.
   This case resembles case 2, with the difference that f is influenced at α. Therefore, we introduce a new feature symbol f' which has the same value at α as f has at b. The new symbols will be treated properly below, in case 6.

5. $\Gamma_5 = \{\neg[\alpha]f \vee [\beta]f', [\alpha]f \vee \neg[\beta]f' | f \in \mathcal{F}, \alpha, \beta \in E_f, \alpha \prec_f \beta\}$.
   This relates to case 3, as case 4 relates to case 2.

6. Since E_f is ordered, we can form equivalence classes T_f^α of time points for $\alpha \in E_f$ by

$$T_f^\alpha = \{\beta \in E_f | \alpha = \beta \text{ is consistent with } \textsc{obs}\}.$$

For each $\alpha \in E_f$, define

$$\mathcal{P}_f^\alpha = \{\pi | \langle \beta, \pi, \texttt{Infl}, \epsilon \rangle \in \textsc{scd} \wedge f \in \texttt{Infl} \wedge \beta \in T_f^\alpha\}.$$

Now set

$$\Gamma_6 = \{\bigvee \mathcal{P}_f^\alpha \vee \neg[\alpha]f \vee [\alpha]f', \bigvee \mathcal{P}_f^\alpha \vee \neg[\alpha]f' \vee [\alpha]f | \alpha \in E_f\}.$$

First note that this set is equivalent to the set

$$\{\neg \bigvee \mathcal{P}_f^\alpha \rightarrow ([\alpha]f \leftrightarrow [\alpha]f') | \alpha \in E_f\}.$$

Here we ensure that when actions do not have effects, f will have the same value as it had the last time it was changed. This value is held by the feature f'.

Now set $\Phi(\Upsilon) = \bigcup_i \Gamma_i$. It is clear that the transformation performed by Φ is polynomial. $\Box$

We look at the transformation of JLH. First we can note that the scenario is normal. For readability, we write the members of the sets as biconditionals and implications instead of as pairs of disjunctions. The application of Φ to the actions has the following results:

$$\begin{aligned}
\Phi(\textsc{scd}1) &= \{[\mathbf{c}_1]\neg on_land\} \\
\Phi(\textsc{scd}2) &= \{\neg[\mathbf{c}_1]on_land \rightarrow [\mathbf{c}_1]\neg dry\} \\
\Phi(\textsc{scd}3) &= \emptyset
\end{aligned}$$

This gives us

$$\begin{aligned}
\Gamma_1 = {} & \\
& \{[0]hat_on \wedge dry \wedge on_land, \mathbf{c}_1 \geq 0 \wedge \mathbf{c}_2 \geq \mathbf{c}_1\} \cup \\
& \{[\mathbf{c}_1]\neg on_land, \neg[\mathbf{c}_1]on_land \rightarrow [\mathbf{c}_1]\neg dry\} \cup \\
& \{b < 0, b < \mathbf{c}_1, b < \mathbf{c}_2\}
\end{aligned}$$

In Γ_2 we ensure that the feature values at time point b is the same as they are at the earliest time point (i.e. time point 0 for all features) in the scenario.

$$\begin{aligned}&\Gamma_2 = \\ &\quad \{[0]hat\_on \leftrightarrow [b]hat\_on\} \cup \\ &\quad \{[0]dry \leftrightarrow [b]dry\} \cup \\ &\quad \{[0]on\_land \leftrightarrow [b]on\_land\}\end{aligned}$$

In Γ_3 we connect feature values at time points where actions have effect with time points where the feature values are bound by observations, and no actions have effects between the time points. Since no such time points exist for any feature except for *on_land* where the time points are $\mathbf{c}_1$ and $\mathbf{c}_1$, we get a set consisting of one tautology.

$$\begin{aligned}&\Gamma_3 = \\ &\quad \{[\mathbf{c}_1]on\_land \leftrightarrow [\mathbf{c}_1]on\_land\}\end{aligned}$$

In Γ_4 we prepare for inertia. We connect feature values at b with the first time points where features may be affected by actions, so that if the preconditions of the actions are false, the feature values at b persist throughout the action.

$$\begin{aligned}&\Gamma_4 = \\ &\quad \{[\mathbf{c}_2]hat\_on' \leftrightarrow [b]hat\_on\} \cup \\ &\quad \{[\mathbf{c}_1]dry' \leftrightarrow [b]dry\} \cup \\ &\quad \{[\mathbf{c}_1]on\_land' \leftrightarrow [b]on\_land\}\end{aligned}$$

Since for all features f in JLH E_f are singleton sets, Γ_5 is empty.

For Γ_6 we note that there will only be three sets $\mathcal{T}_f^\alpha$ and three sets $\mathcal{P}_f^\alpha$ since all three sets E_f are singleton. Thus we get

$$\begin{aligned}&\mathcal{P}_{hat\_on}^{\mathbf{c}_2} = \{[\mathbf{c}_1]\neg on\_land\} \\ &\mathcal{P}_{dry}^{\mathbf{c}_1} = \{[\mathbf{c}_1]\neg on\_land\} \\ &\mathcal{P}_{on\_land}^{\mathbf{c}_1} = \{\mathbf{T}\}\end{aligned}$$

We can now compute Γ_6.

$$\begin{aligned}&\Gamma_6 = \\ &\quad \{\neg[\mathbf{c}_1]\neg on\_land \rightarrow ([\mathbf{c}_2]hat\_on \leftrightarrow [\mathbf{c}_2]hat\_on')\} \cup \\ &\quad \{\neg[\mathbf{c}_1]\neg on\_land \rightarrow ([\mathbf{c}_1]dry \leftrightarrow [\mathbf{c}_1]dry')\}\end{aligned}$$

It is clear that an encoding to Horn DLRs can be performed from this transformation. The following two theorems validate such an encoding.
The proof of the following theorem can be found in Appendix A.

Theorem 5.3.16 Let Υ be a restricted scenario description, and set

$$\Gamma = C(\Phi(\Upsilon)).$$

Then Υ is satisfiable iff Γ is satisfiable. $\square$

Theorem 5.3.17 Deciding satisfiability (and entailment) for restricted scenario descriptions is polynomial. $\square$

5.4 Discussion

In this chapter our concern has been computational complexity for reasoning about action. It is important to note that although we have provided polynomial algorithms for the reasoning tasks, these can hardly be considered efficient. The important results, however, are that there *exist* polynomial algorithms; the next obvious step is to also make them *fast*. For efficient implementation, there is one direction we are particularly interested in investigating: since the technique used for achieving tractability can be described as an encoding of our logic as temporal constraints for which there is a tractable algorithm for determining satisfiability, it should be possible to do something similar for other tractable temporal algebras, for example those identified in the papers by Drakengren and Jonsson [1996, 1997]. Also, an algorithm for a purely *qualitative* scenario description language (i.e. not involving metric time) would probably have a faster satisfiability-checker.

We have shown that satisfiability of scenario descriptions is NP-complete within our formalism. We feel that it would be a mistake to interpret this negatively. On the contrary, one could argue (in lines with [Gottlob, 1996]) that this would imply that many approximations, powerful heuristics and non-trivial tractable subsets of problems for reasoning about action remain to be found. Our work is a step on the way in this endeavour.

Chapter 6

Conclusions and Future Work

The two themes of this thesis have automation of logical reasoning about action and change in common. We have taken a careful look at regression, a technique we used for projecting a logic with explicit time to an atemporal logic. We have presented a computational strategy for prediction and postdiction for a subset of PMON. The strategy is based on Dijkstra's weakest liberal precondition operator. We have shown that wlp and its conjugate wlp^* have different properties for nondeterministic scenarios and that the former is applicable for prediction, and the latter for postdiction. For deterministic scenarios, the two operators coincide, thus providing a formal foundation for generating successor state axioms.

The second theme has been computational complexity of logics of action and change. We have presented a temporal logic and an extension for reasoning about action from which tractable subsets have been extracted. This has been done with an encoding of the logic to Horn DLRs. The formalism is narrative-based with continuous time, and the world is modelled using scenario descriptions consisting of action expressions and observations. It is possible to model nondeterminism, concurrency and memory of actions. Time is represented by linear polynomials with rational coefficients over real-valued variables.

6.1 Future Work

For both themes in this thesis there are a number of open questions and tracks that we would like to investigate.

For regression we, for example, we would like to extend the regression procedures to handle non-propositional fluents, partially-ordered scenarios, and continuous time. This does not seem to be too hard. To extend the regression procedures to handle PMON(RC) ([Gustafsson and Doherty, 1996]) seems, on the other hand, to be quite a challenge, since it would involve developing regression for non-Markovian transitions.

For the complexity theme we would like to see whether other restrictions of the formalism may produce new classes of tractable scenarios. But, the track we are most interested in is to find the limits of the NP-complete class of problems. For example, is it possible to express causal constraints and still have NP-completeness?

References

[AAAI '96, 1996] American Association for Artificial Intelligence. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, August 1996. AAAI Press/MIT Press.

[Alur and Henzinger, 1994] R. Alur and T.A. Henzinger. A Really Temporal Logic. *Journal of the ACM*, (41):181–204, 1994.

[Baral, 1995] C. Baral. Reasoning about actions: Non-deterministic effects, constraints, and qualifications. In IJCAI '95 [1995], pages 384–390.

[Bjäreland and Karlsson, 1997] M. Bjäreland and L. Karlsson. Reasoning by regression: Pre- and postdiction procedures for logics of action and change with nondeterminism. In IJCAI '97 [1997].

[Davis *et al.*, 1993] R. Davis, H. Shrobe, and P. Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[Dijkstra and Scholten, 1990] W. D. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, 1990.

[Dijkstra, 1976] E. W. Dijkstra. *A Discipline of Programming.* Prentice-Hall, 1976.

[Doherty and Łukaszewicz, 1994] P. Doherty and W. Łukaszewicz. Circumscribing features and fluents. In D. Gabbay and H.J. Ohlbach, editors, *Proceedings of the First International Conference on Temporal Logic*, volume 827 of *Lecture Notes in Artificial Intelligence*, 1994.

[Doherty, 1994] P. Doherty. Reasoning about action and change using occlusion. In A.G. Cohn, editor, *Proceedings of the Eleventh European Conference on Artificial Intelligence*, Amsterdam, Netherlands, August 1994.

European Coordinating Committee for Artficial Intelligence, John Wiley & Sons.

[Doherty, 1997] P. Doherty. Pmon+: A fluent logic for action and change: Formal specification, version 1.0. Linkoping Electronic Articles in Computer and Information Sciences 020, Department of Computer and Information Science, 1997. URL: `http://www.ep.liu.se/ea/cis/1997/020/`.

[Drakengren and Bjäreland, 1997] T. Drakengren and M. Bjäreland. Reasoning about action in polynomial time. In IJCAI '97 [1997].

[Drakengren and Jonsson, 1996] Thomas Drakengren and Peter Jonsson. Maximal tractable subclasses of Allen's interval algebra: Preliminary report. In AAAI '96 [1996], pages 389–394.

[Drakengren and Jonsson, 1997] Thomas Drakengren and Peter Jonsson. Eight maximal tractable subclasses of Allen's algebra with metric time. *Journal of Artificial Intelligence Research*, 7:25–45, 1997.

[Gelfond *et al.*, 1991] M. Gelfond, V. Lifschitz, and A. Rabinov. What are the Limitations of the Situation Calculus? In R. Boyer, editor, *Essays in Honor of Woody Bledsoe*, pages 167–179. Kluwer Academic, 1991.

[Ghallab and Laruelle, 1994] M. Ghallab and H. Laruelle. Representation and control in IxTeT, a temporal planner. In K. Hammond, editor, *Proceeding of the Second Conference on Artificial Intelligence Planning Systems (AIPS'94)*, Chicago, 1994. AAAI-Press.

[Ginsberg, 1996] M. Ginsberg. Do Computers Need Common Sense? In KR '96 [1996].

[Giunchiglia and Lifschitz, 1995] E. Giunchiglia and V. Lifschitz. Dependent Fluents. In IJCAI '95 [1995].

[Giunchiglia *et al.*, 1997] E. Giunchiglia, G. N. Kartha, and V. Lifschitz. Representing action: Indeterminancy and ramifications. *Artificial Intelligence*, 95, 1997.

[Gottlob, 1996] G. Gottlob. Complexity and Expressive Power of KR Formalism. In KR '96 [1996].

[Gries, 1981] D. Gries. *The Science of programming.* Springer-Verlag, 1981.

[Gustafsson and Doherty, 1996] J. Gustafsson and P. Doherty. Embracing Occlusion in Specifying the Indirect Effects of Actions. In KR '96 [1996].

[Haslum, 1997] P. Haslum. Regression planning techniques. Master's thesis, Dept of Comp and Info Science, Linköpings universitet, September 1997.

[IJCAI '95, 1995] C. Mellish, editor. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI '95)*, Montreal, Canada, August 1995. Morgan Kaufmann.

[IJCAI '97, 1997] M.E. Pollack, editor. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI '97)*, Nagoya, Japan, August 1997. Morgan Kaufmann.

[Israel, 1994] D. J. Israel. The role(s) of logic in artificial intelligence. In D. Gabbay, editor, *Handbook of logic in artificial intelligence and logic programming*, volume 1. Oxford University Press, 1994.

[Jonsson and Bäckström, 1996] Peter Jonsson and Christer Bäckström. A linear-programming approach to temporal reasoning. In AAAI '96 [1996], pages 1235–1240.

[Karlsson, 1997] L. Karlsson. Reasoning with incomplete initial information and nondeterminism in situation calculus. In IJCAI '97 [1997].

[Koubarakis, 1996] M. Koubarakis. Tractable disjunctions of linear constraints. In *Proceedings of the 2nd International Conference on Principles and Practice for Constraint Programming*, pages 297–307, Cambridge, MA, August 1996.

[Kowalski and Sergot, 1986] R. A. Kowalski and M. J. Sergot. A Logic-Based Calculus of Events. *New Generation Computing*, 4:67–95, 1986.

[KR '96, 1996] J. Doyle and L. Aiello, editors. *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*, Cambridge, Massachusetts, November 1996. Morgan Kaufmann.

[Kvarnström and Doherty, 1997] J. Kvarnström and P. Doherty. Visualization and implementation of temporal action logics. `http://anton.ida.liu.se/vital/vital.html`, 1997.

[Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92(1-2), May 1997.

[Lin, 1996] F. Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. In AAAI '96 [1996].

[Lukaszewicz and Madalinska-Bugaj, 1995] W. Lukaszewicz and E. Madalinska-Bugaj. Reasoning about action and change using Dijkstra's semantics for programming languages: Preliminary report. In IJCAI '95 [1995].

[Łukaszewicz and Madalińska-Bugaj, 1997] W. Łukaszewicz and E. Madalińska-Bugaj. Reasoning about plans. In IJCAI '97 [1997].

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

[McCarthy, 1958] J. McCarthy. Programs with common sense. In *Mechanisation of Thought Processes, Proceedings of the Symposium of the National Physics Laboratory*, 1958. Available at `http://www-formal.stanford.edu/jmc/mcc59.html`.

[McDermott, 1987] D. McDermott. A critique of pure reason. *Computational Intelligence*, 3(3):151–160, August 1987.

[Pednault, 1986] E.D.P. Pednault. ADL: Exploring the middle ground between STRIPS and the Situation Calculus. In *Proceedings of the First International Conference on Principles of Knowledge Representation and reasoning*, Stanford University, December 1986. Morgan Kaufmann.

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

[Sandewall and Shoham, 1994] E. Sandewall and Y. Shoham. Nonmonotonic temporal reasoning. In D. Gabbay, editor, *Handbook of logic in artificial intelligence and logic programming*, volume 2. Oxford University Press, 1994.

[Sandewall, 1994] E. Sandewall. *Features and Fluents. The Representation of Knowledge about Dynamical Systems, volume I.* Oxford University Press, 1994. ISBN 0-19-853845-6.

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

[Thielscher, 1995] M. Thielscher. The logic of dynamic systems. In IJCAI '95 [1995].

[Waldinger, 1977] R. Waldinger. *Achieving Several Goals Simultaneously*, volume 8 of *Machine Intelligence*, pages 94–136. Ellis Horwood, 1977.

Appendix A

Proofs of theorems

Here we present the more space-consuming proofs and necessary definitions of theorems in Chapter 5.

Definition A.0.1 ()
Let $\Gamma \subseteq \Lambda$. Then we define $\neg\Gamma = \{\neg\gamma | \gamma \in \Gamma\}$, *feat*$(\Gamma)$ to be the set of all features occurring in Γ, *linrel*(Γ) to be the set of all linear relations (which are atomic formulae) occurring in Γ, and *time*(Γ) to be the set of all time point expressions occurring in Γ. $\Box$

Theorem 5.3.9 Deciding satisfiability of a set $\Gamma \subseteq \Lambda$ is NP-complete.
Proof (sketch): By Proposition 5.3.1, it remains to prove that the problem is in NP.

Let $\Gamma \subseteq \Lambda$ be a set of formulae. A few auxiliary definitions will be needed for the proof.

Let $F = \mathit{feat}(\Gamma)$, $L = \mathit{linrel}(\Gamma)$, $T = \mathit{time}(\Gamma)$, let t be a fresh time point variable, set

$$A^{+} = \{[\alpha]f | \alpha \in T \cup \{\mathrm{t}\} \wedge f \in F\} \cup L,$$

and

$$A' = A^{+} \cup \neg A^{+}.$$

We say that a set $W \subseteq A'$ is a *syntactic* Λ*-interpretation* iff for any $\phi \in A^{+}$, exactly one of ϕ and $\neg\phi$ is a member of W.

Next, we define a way to evaluate a formula $\gamma \in \Lambda$ in a syntactic Λ-interpretation, following Definition 5.2.8. Let $\gamma \in \Lambda$. We define the *truth value* of γ in W for $\alpha' \in \mathcal{T}^*$, denoted $W(\gamma, \alpha')$, as follows. Assume $f \in \mathcal{F}$, $R \in \{=, \leq, <, \geq, >\}$, $\alpha, \beta \in \mathcal{T}^*$, $\gamma, \delta \in \Lambda$, and $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$.

- $W(\mathbf{T}, \alpha') = \mathbf{T}$

- $W(\mathbf{F}, \alpha') = \mathbf{F}$
- $W(f, \alpha') = (\ulcorner[\alpha']f\urcorner \in W)$
- $W(\alpha R\beta, \alpha') = (\ulcorner \alpha R\beta\urcorner \in W)$
- $W(\neg\gamma, \alpha') = \neg W(\gamma, \alpha')$
- $W(\gamma \oplus \delta, \alpha') = W(\gamma, \alpha') \oplus W(\delta, \alpha')$
- $W([\alpha]\gamma, \alpha') = W(\gamma, \alpha)$

It is clear that $W(\gamma, \alpha')$ is always defined for $\gamma \in \Lambda$. Thus we can say that a syntactic Λ-interpretation W is a *syntactic Λ-model* of a set Γ of formulae iff W is satisfiable, and for all $\gamma \in \Gamma$, $W(\gamma, \mathtt{t})$ is true.

If we can prove that whenever Γ is satisfiable, then there exists a syntactic Λ-model W of Γ, and vice versa, then NP-membership will follow, since

- the size of W is polynomial in the size of Γ
- it can be checked in polynomial time whether W is a syntactic Λ-interpretation or not
- $W(\gamma, \alpha')$ can easily be computed in polynomial time
- by Corollary 5.3.8, checking satisfiability of a set of closed Horn formulae is polynomial, and thus checking satisfiability of W is polynomial.

$\Rightarrow$) Suppose that Γ is satisfiable, i.e. that for some $t \in \mathbb{R}$, $I(\gamma, t) = \mathbf{T}$ for all $\gamma \in \Gamma$. Construct a syntactic Λ-interpretation W from $I = \langle h, \phi\rangle$ by first setting $I' = \langle h, \phi'\rangle$, where $\phi'(s) = \phi(s)$ for all $s \in \mathcal{T} - \{\mathtt{t}\}$, and $\phi'(\mathtt{t}) = t$, and then defining

$$W = \{\delta \in A' | I'(\delta) = \mathbf{T}\}.$$

It remains to check that W is satisfiable, and that $W(\gamma, \mathtt{t}) = \mathbf{T}$ for all $\gamma \in \Gamma$.

W is trivially satisfiable, since by construction I' is a model of W. We prove by induction on γ that $W(\gamma, \alpha') = I(\gamma, \phi'(\alpha'))$ for all $\gamma \in \Lambda$ and $\alpha' \in T \cup \{\mathtt{t}\}$; thus $W(\gamma, \mathtt{t}) = \mathbf{T}$ for all $\gamma \in \Gamma$.

First, the basis cases: If $\gamma \equiv \mathbf{T}$ or $\gamma \equiv \mathbf{F}$, the result is immediate. If $\gamma \equiv f$ for $f \in \mathcal{F}$, then

$$\begin{aligned} & W(\gamma, \alpha') = \\ &\quad = W(f, \alpha') \\ &\quad = (\ulcorner[\alpha']f\urcorner \in W) \\ &\quad = I'([\alpha']f) \\ &\quad = I(f, \phi'(\alpha')). \end{aligned}$$

If $\gamma \equiv \alpha R \beta$ for $\alpha, \beta \in \mathcal{T}^*$ and $R \in \{=, \leq, <, \geq, >\}$, then

$$\begin{aligned} & W(\gamma, \alpha') = \\ & \quad = \quad W(\alpha R \beta, \alpha') \\ & \quad = \quad (\ulcorner \alpha R \beta \urcorner \in W) \\ & \quad = \quad I'(\alpha R \beta, \phi'(\alpha')) \end{aligned}$$

Now, the induction: If $\gamma \equiv \neg \delta$, then

$$\begin{aligned} & W(\gamma, \alpha') = \\ & \quad = \quad W(\neg \delta, \alpha') \\ & \quad = \quad \neg W(\delta, \alpha') \\ & \quad = \quad \neg I(\delta, \phi'(\alpha')) \quad \text{induction} \\ & \quad = \quad I(\neg \delta, \phi'(\alpha')). \end{aligned}$$

If $\gamma \equiv \delta_1 \oplus \delta_2$ for $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, then

$$\begin{aligned} & W(\gamma, \alpha') = \\ & \quad = \quad W(\delta_1 \oplus \delta_2, \alpha') \\ & \quad = \quad W(\delta_1, \alpha') \oplus W(\delta_2, \alpha') \\ & \quad = \quad I(\delta_1, \phi'(\alpha')) \oplus I(\delta_2, \phi'(\alpha')) \quad \text{induction} \\ & \quad = \quad I(\delta_1 \oplus \delta_2, \phi'(\alpha')). \end{aligned}$$

If $\gamma \equiv [\alpha]\delta$, then

$$\begin{aligned} & W(\gamma, \alpha') = \\ & \quad = \quad W([\alpha]\delta, \alpha') \\ & \quad = \quad W(\delta, \alpha) \\ & \quad = \quad I(\delta, \phi'(\alpha)) \qquad \text{induction} \\ & \quad = \quad I([\alpha]\delta, \phi'(\alpha')). \end{aligned}$$

$\Leftarrow$) Suppose that W is a syntactic Λ-model of Γ, i.e. that W is a satisfiable syntactic Λ-interpretation, and $W(\gamma, \mathtt{t}) = \mathbf{T}$ for all $\gamma \in \Gamma$. Let $I = \langle h, \phi \rangle$ be a model of W (note that W contains only closed formulae). We need to show that $I(\gamma, \phi(\mathtt{t})) = \mathbf{T}$ for all $\gamma \in \Gamma$. By induction we prove the stronger result that $I(\gamma, \phi(\alpha')) = W(\gamma, \alpha')$ for all $\gamma \in \Lambda$ and $\alpha' \in T \cup \{\mathtt{t}\}$.

First, the basis cases: If $\gamma \equiv \mathbf{T}$ or $\gamma \equiv \mathbf{F}$, the result is immediate. If $\gamma \equiv f$ for $f \in \mathcal{F}$, then

$$\begin{aligned} & I(\gamma, \phi(\alpha')) = \\ & \quad = \quad I(f, \phi(\alpha')) \\ & \quad = \quad I([\alpha']f) \\ & \quad = \quad (\ulcorner [\alpha']f \urcorner \in W) \\ & \quad = \quad W(f, \alpha'). \end{aligned}$$

If $\gamma \equiv \alpha R \beta$ for $\alpha, \beta \in \mathcal{T}^*$ and $R \in \{=, \leq, <, \geq, >\}$, then

$$\begin{aligned} & I(\gamma, \phi(\alpha')) = \\ & \quad = I(\alpha R \beta, \phi(\alpha')) \\ & \quad = (\alpha R \beta \in W) \\ & \quad = W(\alpha R \beta, \alpha'). \end{aligned}$$

Now, the induction: If $\gamma \equiv \neg\delta$, then

$$\begin{aligned} & I(\gamma, \phi(\alpha')) = \\ & \quad = I(\neg\delta, \phi(\alpha')) \\ & \quad = \neg I(\delta, \phi(\alpha')) \\ & \quad = \neg W(\delta, \alpha') \\ & \quad = W(\neg\delta, \alpha'). \end{aligned}$$

If $\gamma \equiv \delta_1 \oplus \delta_2$ for $\oplus \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, then

$$\begin{aligned} & I(\gamma, \phi(\alpha')) = \\ & \quad = I(\delta_1 \oplus \delta_2, \phi(\alpha')) \\ & \quad = I(\delta_1, \phi(\alpha')) \oplus I(\delta_2, \phi(\alpha')) \\ & \quad = W(\delta_1, \alpha') \oplus W(\delta_2, \alpha') \\ & \quad = W(\delta_1 \oplus \delta_2, \alpha'). \end{aligned}$$

If $\gamma \equiv [\alpha]\delta$, then

$$\begin{aligned} & I(\gamma, \phi(\alpha')) = \\ & \quad = I([\alpha]\delta, \phi(\alpha') \\ & \quad = I(\delta, \phi(\alpha)) \\ & \quad = W(\delta, \alpha) \\ & \quad = W([\alpha]\delta, \alpha'). \end{aligned}$$

Thus the result follows. □

Theorem 5.3.10 Deciding whether a scenario description is satisfiable is NP-complete.

Proof (sketch): By Corollary 5.3.2, it remains to prove that the problem is in NP.

Let $\Upsilon = \langle \sigma, \textsc{scd}, \textsc{obs} \rangle$ be a scenario description. A few auxiliary definitions will be needed for the proof.

Define the set Δ of formulae by

$$\Delta = \textsc{obs} \cup \{\ulcorner \pi \rightarrow [\alpha]\epsilon \urcorner | \langle \alpha, \pi, \texttt{Infl}, \epsilon \rangle \in \textsc{scd}\},$$

let $T = time(\Delta)$, $F = feat(\Delta)$, $L = linrel(\Delta)$, and define

$$\Delta' = \{\alpha R \beta | \alpha, \beta \in T \wedge R \in \{=, <, >\}\},$$

$$A^+ = \{[\alpha] f | \alpha \in T \wedge f \in F\} \cup L \cup \Delta',$$

and

$$A' = A^+ \cup \neg A^+.$$

We say that a set $W \subseteq A'$ is a *syntactic scenario interpretation* iff for any $\phi \in A^+$, exactly one of ϕ and $\neg\phi$ is a member of W.

By employing exactly the same method as in Theorem 5.3.9, we can define the *truth value* in W of a formula $\gamma \in \Delta \cup \Delta'$, denoted $W(\gamma)$ (no temporal parameter is needed, since all formulae in $\Delta \cup \Delta'$ are closed). Exactly as in Theorem 5.3.9, W satisfies the following:

- If I is a model of $\Delta \cup \Delta'$, and we set

  $$W = \{\delta \in A' | I(\delta) = \mathbf{T}\},$$

  then W is a syntactic Λ-model of $\Delta \cup \Delta'$, and I is a model of W.

- If W is a syntactic Λ-model of $\Delta \cup \Delta'$ which is satisfiable by an interpretation I, then I is a model of $\Delta \cup \Delta'$.

Two more auxiliary definitions are needed, for W being a syntactic scenario interpretation.

First, let $\alpha, \beta \in W$. Then we say that α *precedes* β in W, written $\alpha \lhd \beta$, iff $\alpha < \beta \in W$, and for no $\beta' \in T$, $\alpha < \beta' \in W$ and $\beta' < \beta \in W$ holds. If for no α', $\alpha' \lhd \beta$, we write $-\infty \lhd \beta$, and if for no β', $\alpha \lhd \beta'$, we write $\alpha \lhd \infty$.

Second, W is a *syntactic scenario model* of the scenario description Υ iff the following holds:

- W is a syntactic Λ-model of Δ

- For each $f \in \mathcal{F}$ and $\alpha, \beta \in T$ such that $\alpha \lhd \beta$ in W, if there is no action expression $\langle \beta, \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd}$ with $f \in \mathtt{Infl}$ for some π, $\mathtt{Infl}$ and ϵ, then $[\alpha] f \in W$ iff $[\beta] f \in W$.

Now, the following remains in order to prove the result of the theorem:

- If Υ is satisfiable, then there exists a syntactic scenario model W for Υ

- If there exists a syntactic scenario model W for Υ, then Υ is satisfiable.

This suffices for proving NP-membership, due to the following:

- The size of W is polynomial in Υ
- Whether W is a syntactic Λ-model of $\Delta \cup \Delta'$ or not can be checked in polynomial time by the same argument as in Theorem 5.3.9
- Checking the second condition of the definition of syntactic scenario model is polynomial, since we can start by sorting elements of T by the order imposed on T by W (α comes before β iff $\alpha < \beta \in W$), and then proceed with the simple checks of SCD, which is obviously polynomial.

$\Rightarrow$) Suppose that Υ is satisfiable with an intended model $D = \langle h, \phi \rangle$. Now D is a clearly a model of Δ, and we can set

$$W = \{\delta \in A' | D(\delta) = \mathbf{T}\}.$$

Clearly W is a syntactic Λ-model of Δ. It remains to check the second condition in the definition of syntactic scenario model. Take $f \in \mathcal{F}$ and $\alpha, \beta \in T$ satisfying the required conditions, and suppose there is no action expression $\langle \beta, \pi, \mathtt{Infl}, \epsilon \rangle \in \text{SCD}$ such that $f \in \mathtt{Infl}$. By the definition of T and our assumption, we cannot have $Chg(D, \text{SCD}, f, t')$ to be true for any t' with $D(\alpha) < t' \leq D(\beta)$. Since there are only finitely many points where Chg is true, we can choose a time point $t'' > D(\beta)$ such that for no $s \in [D(\beta), t'')$, $Chg(D, \text{SCD}, f, s)$ is true. Now, the definition of intended model yields that $[\alpha]f$ is true iff $[\beta]f$ in D, and by the definition of W, that $[\alpha]f \in W$ iff $[\beta]f \in W$.
$\Leftarrow$) Suppose that W is a syntactic scenario model for Υ, which is satisfied by a model $I = \langle h, \phi \rangle$. We construct an intended model $D = \langle h', \phi \rangle$ from I as follows. For each feature $f \in F$ do the following. First for each α such that $-\infty \lhd \alpha$ in W, set $h'(t)(f) = h(\phi(\alpha))(f)$ for every $t \leq \phi(\alpha)$. Then for each α, β with $\alpha \lhd \beta$ in W, set $h'(t)(f) = h(\phi(\alpha))(f)$ for every t such that $\phi(\alpha) \leq t < \phi(\beta)$. Finally, for each α such that $\alpha \lhd \infty$, set $h'(t)(f) = h(\phi(\alpha))(f)$ for every $t \geq \phi(\alpha)$. It is clear that h' is defined for every f and t. It remains to verify that D is an intended model for Υ.

Since W is a syntactic scenario model for Υ, it is also a syntactic Λ-model of Δ. By the fact that I is a model of W, and since D and I agree on the values of all time point expressions in T, D is also a model of W, and by what we know about syntactic Λ-interpretations, D is a model of Δ. Thus, what is left to verify is the second condition in the definition of intended model.

For this purpose, let f be a feature and $s, t \in \mathbb{R}$ with $s < t$, such that for no $t' \in (s, t)$, $Chg(D, \textsc{scd}, f, t')$ holds. We want to prove that $h'(t')(f) = h'(s)(f)$ for every $t' \in (s, t)$. Suppose to the contrary that for some t' with $s < t' < t$, $h'(t')(f) \neq h'(s)(f)$. It is easy to see that by the construction of D from I, this cannot hold, and we have a contradiction. Thus D is an intended model of Υ. $\Box$

Theorem 5.3.16 Let Υ be a restricted scenario description, and set $\Gamma = C(\Phi(\Upsilon))$. Then Υ is satisfiable iff Γ is satisfiable.
Proof (sketch): We start by defining a set Γ' which is satisfiable iff Γ is, by first defining sets Γ'_i for $i \in \{1, \ldots, 6\}$, giving $\Gamma' = \bigcup_{1 \leq i \leq 6} \Gamma'_i$:

1. $\Gamma'_1 = \Gamma_1$
2. $\Gamma'_2 = \{[\alpha]f \leftrightarrow [b]f | f \in \mathcal{F}, \alpha \in C_f, -\infty \ll_f \alpha\}$
3. $\Gamma'_3 = \{[\alpha]f \leftrightarrow [\beta]f | f \in \mathcal{F}, \alpha \in E_f, \beta \in C_f, \alpha \ll_f \beta\}$
4. $\Gamma'_4 = \{[\alpha]f' \leftrightarrow [b]f | f \in \mathcal{F}, \alpha \in E_f, -\infty \prec_f \alpha\}$
5. $\Gamma'_5 = \{[\alpha]f \leftrightarrow [\beta]f' | f \in \mathcal{F}, \alpha, \beta \in E_f, \alpha \prec_f \beta\}$
6. $\Gamma'_6 = \{\neg(\bigvee_i \pi_i) \rightarrow ([\alpha]f \leftrightarrow [\alpha]f') | \mathbf{T} \not\in \{\pi_i\} = \mathcal{P}^{\alpha}_f, \alpha \in E_f\}$.

It is easy to see that each Γ'_i is satisfiable iff the corresponding Γ_i is.
$\Rightarrow$) Suppose Υ is satisfiable by an intended model $D = \langle h, \phi \rangle$. We first construct a new interpretation $D' = \langle h', \phi' \rangle$ as follows.

- Define ϕ' by setting $\phi'(s) = \phi(s)$ for all $s \in \mathcal{T} - \{b\}$, and $\phi'(b) = \min\{\phi(\alpha) | \alpha \in T\} - 1$.
- For each feature f, set $h'(t)(f) = h(t)(f)$.
- For each feature f, introduce a new feature f', and define $h'(t)(f')$ so that the formulae in Γ'_4 and Γ'_5 are all true. It is clear that this is possible, since these are essentially *definitions*.

We want to prove that D' is a model of Γ. This is done by checking that for every $i \in \{1, \ldots, 6\}$, D' is a model of Γ'_i.

1. For Γ'_1, all formulae in OBS are true in D', by definition. Furthermore, all formulae in $\Phi(\text{SCD})$ will be true, since these code exactly the conditions required by an intended model, in terms of actions. Furthermore, we have set $\phi'(b)$ to be strictly smaller than the value of every time point expression used, so everything in the last part of Γ'_1 will be true in D'.

2. Γ'_2 says that for each observation or action precondition involving a feature f that is not preceded by any effect on the feature f, the value will be the same as the value at b. Since the value of b in D' is less than every time point expression used, this amounts to strict inertia in each f at all time points before the first change in f. This is clearly satisfied in D', since it inherits this property from the intended model D.

3. Γ'_3 says that for any feature f, observations and action precondition involving f coming directly after changes of f with no change in between, f should retain its value from the change. This is the same as inertia, which is clearly satisfied in any intended model and is inherited from D.

4. Γ'_4 is satisfied by construction.

5. Γ'_5 is satisfied by construction.

6. Γ'_6 says that if none of the actions which can cause a change in f at α has its precondition true, then f and f' coincide at $\alpha \in E_f$. Since by the definitions of the f' formulae, if $\beta \in E_f$, then f' is always forced to have the value at β that f had just before it changed at β, which just amounts to inertia, which holds for f by construction.

$\Leftarrow$) Suppose Γ is satisfiable by a model $I = \langle h, \phi \rangle$. We construct an intended model $D = \langle h', \phi \rangle$ of Υ from I as follows. Start by removing all features f' for features f. Then for each feature f, do the following. First, for each $\alpha \in E_f$ such that $-\infty \prec_f \alpha$, set $h'(t)(f) = h(\phi(b))(f)$ for each t such that $t < \phi(\alpha)$. Then for each $\alpha, \beta \in E_f$ such that $\alpha \prec_f \beta$, set $h'(t)(f) = h(\phi(\alpha))(f)$ for each t such that $\phi(\alpha) \leq t < \phi(\beta)$. Finally, for each $\alpha \in E_f$ such that $\alpha \prec_f \infty$, set $h'(t)(f) = h(\phi(\alpha))(f)$ for each $t \geq \phi(\alpha)$. It is clear that h' is defined for every t and f, and that by construction, for each f and $\alpha \in E_f$, $h(\phi(\alpha))(f) = h'(\phi(\alpha))(f)$. It remains to verify that D is an intended model of Υ.

First note that D is still a model of $\Gamma'_1 \cup \Gamma'_2 \cup \Gamma'_3$, since the truth of all these formulae are preserved by the transformation from I to D. Thus, D is a model of OBS, since all values used to evaluate the truth of formulae in OBS are identical in I and D, due to D being a model of Γ'_2 and Γ'_3.

For the second condition of the definition of intended model, note that the set

$$\Delta = \{\pi \rightarrow [\alpha]\epsilon | \langle \alpha, \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd}\}$$

is satisfiable iff $\Phi(\textsc{scd})$ is, by construction, and that this is equivalent to the second condition being satisfied. Thus, since I is a model of $\Phi(\textsc{scd})$, I is also a model of Δ. It remains to check that D is also a model of Δ. Exactly as for the previous condition, the truth values of the π component will not change from I to D. Furthermore, by construction, D will have the same values as I on the $[\alpha]\epsilon$ expressions, since values of features f at time points $\alpha \in E_f$ are the same in D and I. Thus D is also a model of Δ.

Now the proof for the third condition. Suppose that for a feature f and time points $s, t \in \mathbb{R}$ with $s < t$, we have that for no $t' \in (s, t)$, $\mathit{Chg}(D, \textsc{scd}, f, t')$ holds, but $h'(t')(f) \neq h'(s)(f)$ for some $t' \in (s, t)$. Now, it has to hold that $t' = \phi(\alpha')$ for some $\alpha' \in E_f$, by the construction of D.

First suppose $-\infty \prec_f \alpha'$. By the construction of D, $h'(s)(f) = h'(\phi(b))(f)$, and $h'(\phi(b))(f) = h(\phi(b))(f)$, and thus $h'(s)(f) = h(\phi(b))(f)$. Since I is a model of Γ'_4, $h(\phi(\alpha'))(f') = h'(s)(f)$; so by assumption, $h(\phi(\alpha'))(f') \neq h'(\phi(\alpha'))(f)$, and by the construction of D, $h(\phi(\alpha'))(f') \neq h(\phi(\alpha'))(f)$. I is a model of Γ'_6; thus $\neg \bigvee \mathcal{P}_f^{\alpha'}$ must be false in I, and so π is true in I for some $\langle \alpha', \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd}$ with $f \in \mathtt{Infl}$. But this means that $\mathit{Chg}(D, \textsc{scd}, f, t')$, a contradiction.

Then suppose $\alpha \prec_f \alpha'$ for some $\alpha \in E_f$. It will suffice to find a contradiction for $s = \phi(\alpha)$, so we make that assumption. Now, by the construction of D, $h'(\phi(\alpha))(f) = h(\phi(\alpha))(f)$, and thus $h'(s)(f) = h(\phi(\alpha))(f)$. Since I is a model of Γ'_5, $h(\phi(\alpha'))(f') = h'(s)(f)$; so by assumption, $h(\phi(\alpha'))(f') \neq h'(\phi(\alpha'))(f)$, and by the construction of D, $h(\phi(\alpha'))(f') \neq h(\phi(\alpha'))(f)$. I is a model of Γ'_6; thus $\neg \bigvee \mathcal{P}_f^{\alpha'}$ must be false in I, and so π is true in I for some $\langle \alpha', \pi, \mathtt{Infl}, \epsilon \rangle \in \textsc{scd}$ with $f \in \mathtt{Infl}$. But this means that $\mathit{Chg}(D, \textsc{scd}, f, t')$, a contradiction. The result follows. □