FORMALISMS FOR AGENTS REASONING WITH STOCHASTIC ACTIONS AND PERCEPTIONS

by

GAVIN B. RENS

Submitted in fulfilment of the academic requirements for the degree of

DOCTOR OF PHILOSOPHY

in the

School of Mathematics, Statistics and Computer Science,

University of KwaZulu-Natal,
Durban,
South Africa

SUPERVISOR : **PROF. T. MEYER**
JOINT SUPERVISOR : **PROF. G. LAKEMEYER**

November 2014

SUMMARY

The thesis reports on the development of a sequence of logics (formal languages based on mathematical logic) to deal with a class of uncertainty that agents may encounter. More accurately, the logics are meant to be used for allowing robots or software agents to reason about the uncertainty they have about the effects of their actions and the noisiness of their observations. The approach is to take the well-established formalism called the partially observable Markov decision process (POMDP) as an underlying formalism and then design a modal logic based on POMDP theory to allow an agent to reason with a knowledge-base (including knowledge about the uncertainties).

First, three logics are designed, each one adding one or more important features for reasoning in the class of domains of interest (i.e., domains where stochastic action and sensing are considered). The final logic, called the Stochastic Decision Logic (SDL) combines the three logics into a coherent formalism, adding three important notions for reasoning about stochastic decision-theoretic domains: (i) representation of and reasoning about degrees of belief in a statement, given stochastic knowledge, (ii) representation of and reasoning about the expected future rewards of a sequence of actions and (iii) the progression or update of an agent's epistemic, stochastic knowledge.

For all the logics developed in this thesis, *entailment* is defined, that is, whether a sentence logically follows from a knowledge-base. Decision procedures for determining entailment are developed, and they are all proved sound, complete and terminating. The decision procedures all employ tableau calculi to deal with the traditional logical aspects, and systems of equations and inequalities to deal with the probabilistic aspects.

Besides promoting the compact representation of POMDP models, and the power that logic brings to the automation of reasoning, the Stochastic Decision Logic is novel and significant in that it allows the agent to determine whether or not a set of sentences is entailed by an arbitrarily precise specification of a POMDP model, where this is not possible with standard POMDPs.

The research conducted for this thesis has resulted in several publications and has been presented at several workshops, symposia and conferences.

Key terms: cognitive robotics, situated agents, reasoning, decision-making, uncertainty, entailment, modal logic, probability, stochastic, POMDP, action, sensing, observation, perception, epistemic, update

PREFACE

The research described in this thesis was carried out at the Centre for Artificial Intelligence Research and CSIR Meraka in Pretoria from May 2010 to April 2014, under the supervision of Professor Thomas Meyer, except for the period from October 2011 to September 2012, when research was conducted at the Knowledge-Based Systems Group of the department of Computer Science at the RWTH Technical University in Aachen, Germany, under the supervision of Professor Gerhard Lakemeyer.

These studies represent original work by the author and have not otherwise been submitted in any form for any degree or diploma to any tertiary institution. Whenever other researchers' work is used, they are duly acknowledged in the text.

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE
DECLARATION 1 - PLAGIARISM

I, _____, declare that

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other university.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    a Their words have been re-written but the general information attributed to them has been referenced.
    b Where their exact words have been used, then their writing has been placed inside quotation marks or indented, and referenced.
5. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the Bibliography.

Signed: _____

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE
DECLARATION 2 - PUBLICATIONS

Below follows a list of publications that form part of or include research presented in this thesis, whether submitted, accepted or published, with a remark indicating, for each publication, which case applies. For each publication, I am the main author, with second, third, etc. authors contributing by providing guidance and valuable feedback and advice with respect to technical content.

- Published:
  G. Rens, I. Varzinczak, T. Meyer, and A. Ferrein. A logic for reasoning about actions and explicit observations. In Jiuyong Li, editor, AI 2010: Advances in Artificial Intelligence, Proceedings of the Twenty-third Australasian Joint Conference, volume 6464 of LNAI, pages 395-404, Berlin/Heidelberg, December 2010. Springer Verlag. ISBN 978-3-642-17431-5.

- Published:
  G. Rens, G. Lakemeyer, and T. Meyer. A logic for specifying agent actions and observations with probability. In K. Kersting and M. Toussaint, editors, Proceedings of the Sixth Starting AI Researchers' Symposium (STAIRS 2012), volume 241 of Frontiers in Articial Intelligence and Applications, pages 252-263. IOS Press, 2012.
  url: http://www.booksonline.iospress.nl/Content/View.aspx?piid=31509.

- Published:
  G. Rens, T. Meyer, and G. Lakemeyer. On the logical specification of probabilistic transition models. In Proceedings of the Eleventh International Symposium on Logical Formalizations of Commonsense Reasoning (COMMONSENSE 2013), May 2013.
  url: http://www.commonsense2013.cs.ucy.ac.cy/docs/commonsense2013_submission_9.pdf

- Published:
  G. Rens and A. Ferrein. Belief-node condensation for online POMDP algorithms. In Proceedings of IEEE AFRICON 2013, pages 1270-1274, Red Hook, NY 12571 USA, September 2013. Institute of Electrical and Electronics Engineers, Inc.

- Published:
  G. Rens, T. Meyer, and G. Lakemeyer. SLAP: Specification logic of actions with probability. Journal of Applied Logic, 12(2), pages 128-150, April 2014.
  url: http://www.sciencedirect.com/science/article/pii/S157086831300075X

- Published:
  G. Rens, T. Meyer, and G. Lakemeyer. A logic for specifying stochastic actions and observations. In C. Beierle and C. Meghini, editors, Proceedings of the Eighth International Symposium on Foundations of Information and Knowledge Systems (FoIKS), Lecture Notes in Computer Science, pages 305-323. Springer-Verlag, 2014.

- To appear:
  G. Rens, T. Meyer, and G. Lakemeyer. A Modal Logic for the Decision-Theoretic Pro-

jection Problem. Seventh International Conference on Agents and Artificial Intelligence (ICAART), Lisbon, Portugal, 10-12 January, 2015.

Signed: _____

ACKNOWLEDGEMENTS

I would like to thank especially two people for their support during my doctoral studies: Tommie Meyer supervised my research in all respects and he gave patient advice whenever needed. Tommie always treated me as a peer, yet he would advise with authority. I enjoyed our meetings. My wife, Esmarie, supported me in domestic matters, helping me when i stressed out, and congratulating me heartily on successes. I thank her for the four years of moral and loving support, not to mention the years leading up to the doctoral studies.

Thanks to Gerhard Lakemeyer for his insights and advice. And thanks to Gerhard for letting me study under his supervision for one year in Aachen, Germany. Thanks to Alexander (Sascha) Ferrein for his collaborations, and for his support in the early stages of my PhD studies. I also remember Ivan Varzinczak's enthusiastic collaboration and tuition, where he co-authored the very first paper during my PhD. Thanks to Arina Britz for believing in me all those years ago.

I appreciate the moral support of my parents and parents-in-law too. It is nice to be able to share the highs and lows of the last four years with close family.

Administrative work is usually thankless. However, i very much appreciate the administrative support of Laila Gurudas (always via email) situated at the Westville campus of the University of KwaZulu-Natal, and Marlene Jivan at the CAIR office in Pretoria. You were both stellar.

Of course, i would not have been able to do the research full-time in relative comfort, if it were not for the financial support of the University of KwaZulu-Natal, and the Meraka Institute of the Council for Scientific and Industrial Research (CSIR). My scholarship was funded by both these organizations. And the one year i could spend in Germany as part of my studies was made possible by the German Academic Exchange Programme (DAAD) - it was a great experience, both academically and culturally.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION AND MOTIVATION

The world is full of uncertainty and noise. Due to physiological limitations or due to limiting factors in the environment, our actions are not always executed as we wish. For instance, one may not always catch a ball thrown to one, simply because one does not have the motor skills to react fast and accurate enough. The ball being slippery or the wind blowing may also contribute to the difficulty of catching the ball. Walking straight also becomes more difficult for the elderly and the intoxicated. Due to limitations in our sensing organs or due to limiting factors in the environment, our perceptions are also not perfect. For instance, reading words at a distance can be difficult due to being near-sighted. And lighting conditions also affect one's ability to read words at a distance. Smelling whether an aroma is of beef or bread is easier for some people, but becomes more difficult, in general, to the degree that other aromas are present. Engineered systems and machines have limitations analogous to physical human bodies.

In this thesis, when an *agent* is referred to, what is meant is a *situated agent* defined as follows.

Definition: A situated agent is an human engineered entity (machine, robot or software system) in an environment, which has a degree of autonomy, can affect the environment and can be affected by it.

We are concerned here with supplying artificial (human engineered) agents with one of the modules of its intelligence.

We call an actuator and sensor which has significant noise or produces significant uncertainty or is, in general, imperfect and unpredictable to some significant degree, a *stochastic* actuator and a *stochastic* sensor. Similarly, an environment which causes uncertainty in the effects of actions or which causes noise in the data gathered during perception is referred to as a *stochastic* environment. We shall investigate how stochasticity in domains containing a stochastic agent or which have a stochastic environment can be modeled with probability theory and formal languages.

When reasoning about the effects of actions (to decide what to do next), it is rational to incorporate any knowledge about domain stochasticity rather than to simply ignore it. For instance, if you have bad balance or if you are exhausted, your walking behavior will be unstable; you should thus not walk near the edge of a cliff. That is, given knowledge about the uncertainty in the effect of walking, one can make better decisions about where to walk (i.e., what to do). Hence, it is useful to have a means of reasoning about the effects of actions in stochastic domains.

For non-trivial tasks, an agent keeps some form of representation of its knowledge. It is this representation over which the agent reasons for decision-making. The work in this thesis is concerned with defining languages to model situated agents who have to deal with the uncertainty and noise

inherent in their muscles/actuators, and uncertainty and noise caused by the environment in which the agent normally operates.

Due to its autonomy, a situated agent must be able to reason in order to make decisions. Reasoning involves weighing up alternatives. In order to compare alternatives, an agent must have representations or models of them. Hence, we shall assume that the agents we consider in this thesis maintain models of their environment, even though their knowledge is incomplete and hazy. And as an agent or robot executes actions and makes observations, it must update its beliefs (subjective knowledge-base) accordingly.

Formal logics are generally accepted as being ideal for knowledge representation and reasoning [Levesque and Pirri, 1999, Brachman and Levesque, 2004, Darwiche, 2008, Kowalski, 2011]. But classical logics do not deal well with fine-grained uncertainty:

> There are many normative arguments for the use of logic in AI (Nilsson 1991, Poole et al. 1998). These arguments are usually based on reasoning with symbols with an explicit denotation, allowing relations amongst individuals, and quantification over individuals. This is often translated as needing (at least) the first-order predicate calculus. Unfortunately, the first-order predicate calculus has very primitive mechanisms for handling uncertainty, namely the use of disjunction and existential quantification. [Poole, 1998, § 1.3]

We shall use probability theory to capture notions of stochasticity. One formalism, with its basis in probability theory, for modeling and reasoning about stochastic domains, is the well-established partially observable Markov decision process (POMDP) [Monahan, 1982]. POMDP theory has proven to be a good general framework for formalizing *dynamic, stochastic* systems, with the required fine granularity. Moreover, POMDPs add a notion of preference for informing agent behavior. That is, preferences assist agents to compare alternative options. A drawback of traditional POMDP models though, is that they do not and cannot include information about general facts and laws. Moreover, *axioms* describing the dynamics of a domain cannot be written in POMDP theory. This is what logics are good at.

Several frameworks exist for reasoning about probabilistic inference in static domains [Bacchus, 1990, Fagin and Halpern, 1994, Halpern, 2003, Kooi, 2003, Shirazi and Amir, 2007, Van Benthem et al., 2009]. Here, a "static domain" is a domain in which the physical state of the system does not change, although the state of *information* of various agents in the system may change. A distinction between the two kinds of domains is not always clear, but we take action as primary and action effects on belief as secondary. Some of these frameworks are concerned with how knowledge changes as new information is gained. However, the information received is not considered to be a primitive *object* in the language of discourse. Moreover, they do not express the probability with which the received information was expected in the current situation. That is, they take the new information as *certain*. The focus of the logics presented in this thesis (our logics) is more on how stochastic actions change the *physical* state of a system—than on the dynamics of agents' epistemic state (belief revision [Gärdenfors, 1988]). With the main logic presented in this thesis, the dynamics of an agent's epistemic state is accounted for with the belief update approach of Bayes' Rule in probability theory (see, e.g., Russell and Norvig [2003] or Poole and Mackworth [2010]).

Many popular frameworks for reasoning about action employ or are based on the situation calculus [Reiter, 2001]. Reified situations make the meaning of formulae perspicuous. For instance, the framework proposed by Bacchus et al. [1999] (BHL) and the logic $\mathcal{ESP}$ by Gabaldon and Lakemeyer [2007] are based on the situation calculus and are for reasoning with probabilities of actions and observations. These logics are also able to express notions of (stochastic) belief. However, the situation calculus seems too rich and expressive for our purposes, and it would be desirable to remain decidable, hence the restriction in our logics to a propositional modal framework. Furthermore our logics are narrower than those based on first-order predicate calculus in the sense that our logics have finite vocabularies and they do not allow function symbols. Nor do our logics allow predicates in general, only some very specialized predicates.

The entailment problem is decidable for all our logics, which set them apart from first-order logics for reasoning about action (including the situation calculus) or reasoning with probabilities (including BHL's approach and $\mathcal{ESP}$). In other words, having a decidable formalism to reason about POMDPs is considered an asset and sets us apart from other more expressive logical formalisms addressing action and sensing under uncertainty. Moreover, BHL's approach and $\mathcal{ESP}$ cannot deal with nondeterministic actions (see § 8.4 and § 8.8 in Chap. 8).

There are some other logics which also come close to what we desire [De Weerdt et al., 1999, Van Diggelen, 2002, Van Benthem et al., 2009]. They incorporate notions of probability, but they were not created with POMDPs in mind and typically do not take observations as first-class objects. On the other hand, there are formalisms for specifying POMDPs that employ logic-based representation [Boutilier and Poole, 1996, Wang and Schmolze, 2005, Sanner and Kersting, 2010], but they are not defined entirely as logics. Our work is to bring the representation of and reasoning about POMDPs *totally* into the logical arena. One is then in very familiar territory and new opportunities for the advancement in reasoning about POMDPs may be opened up.

A logic-based language such as Golog [Levesque et al., 1997] has proven effective for robot programming, especially because of its ability to constrain the action search space by axiomatic/logical statements. DTGolog [Boutilier et al., 2000] has the benefits of Golog, and furthermore can deal with domains modeled as Markov decision processes (MDPs). But it is a programming language, rather than a logic, and it does not deal with stochastic observations. PODTGolog [Rens, 2010] is a Golog dialect attempting to deal with partially observable MDP (POMDP) environments, but it does not have a well defined semantics.

The main contribution of this thesis is thus to provide a logic for specifying and reasoning over nondeterministic, stochastic actions and perceptions, taking the semantics of POMDPs to do so. We have called the proposed logic the Stochastic Decision Logic (SDL). SDL allows the user to determine whether or not a set of sentences is entailed by an arbitrarily precise specification of a POMDP model. As far as we know, this is a novel property of SDL. Moreover, the procedure for deciding entailment is proved sound, complete and terminating. As a corollary, the entailment question for SDL is decidable.

In total, four *logics* are developed in this thesis. The first two, the Logic of Actions and Observations (LAO) and the Specification Logic of Actions with Probability (SLAP) lay down a solid foundation for what we want to achieve. A POMDP model includes an explicit set of observations. To get closer to the semantics of POMDPs, we developed LAO, which is a logic with explicit observation constants. LAO excludes all probabilistic notions. LAO will also introduce the flavor of

modal logic we use as the basis for the other logics. LAO could also be used independently in domains where one wants to give observations a more important status, or to separate observations (events) from how an agent represents its knowledge. To sort out issues with the specification of stochastic actions using our particular choice of logical language, we developed SLAP. With SLAP, one can specify probabilistic transition models. SLAP could also be used independently when there is a need to reason about stochastic actions, but where perception is needs not be modeled. These two logics might be used for relatively simple investigations; for answering particular philosophical or technical questions.

SLAP excludes any notion of perception/observation. Issues related to the specification of stochastic observations are left for the development of the third logic presented: The Specification Logic of Actions and Observations with Probability (SLAOP), can be thought of as a bridge from the foundation logics to the fourth logic. It includes notions of stochastic actions and stochastic perceptions, but it does not have a notion for (uncertain) epistemic states. SLAOP adds not only a notion of stochastic perception to SLAP, but also notions of reward and cost, bringing the logic into the realm of stochastic decision theory. Further elements are added to SLAOP so that the jump from SLAOP to the SDL is manageable.

The fourth logic is SDL, the 'main' logic of this thesis. SDL includes all the features of the other logics and also allows for the specification of and reasoning over epistemic stated and decision-theoretic projections. SDL takes SLAOP as basis and adds three important notions for reasoning about stochastic decision-theoretic domains: (i) representation of and reasoning about degrees of belief in a statement, given stochastic knowledge, (ii) representation of and reasoning about the utility (i.e., the expected future reward) of a sequence of actions and (iii) the progression or update of an agent's epistemic, stochastic knowledge. The task of the logic is to check whether a query (stated in the language of the logic) follows from a knowledge-base (KB), which is typically a POMDP model specification (also stated in the language of the logic). It will be seen that SDL has a relatively close relationship with POMDPs in meaning and application. In Chapter 7, a theorem relating POMDPs to the SDL is stated. The main contribution of SDL is that a POMDP model specification is allowed to be partial or incomplete with no restriction on the lack of information specified for the model. The KB may even contain information about non-initial beliefs. Essentially, entailment of arbitrary queries (expressible in the language) can be answered.

Every one of the four logics defined in this thesis has something in common with POMDPs—LAO and SLAP to a lesser extent and SLAOP and the SDL to a greater extent. Reviewing POMDP theory in Chapter 3 should help prepare the reader for the rest of the thesis. It is most needed for the understanding of SLAOP and the SDL. Readers not familiar with POMDP theory may wish to refresh their memory after Chapter 5 by consulting Chapter 3 again.

Each of our four logics employs a tableau calculus in the procedure to decide entailment. In the logics containing probabilities, the tableaux are extended with sub-procedures for deciding the feasibility of systems of equations and inequalities. These systems are used to check whether the probabilistic information provided is satisfiable. Systems of linear inequalities are at the heart of Nilsson's probabilistic logic [Nilsson, 1986], which has been extended with stochastic actions by Thiébaux et al. [1995]. Fagin et al. [1990] use a similar idea to prove that the axiomatization of their logic for reasoning about probabilities is complete. For SLAOP and the SDL, systems of equations must be set up to deal with stochastic *perception* information too. The logics of Nilsson

[1986], Thiébaux et al. [1995], Fagin et al. [1990] do not deal with observations.

The reason why we developed a sequence of four logics is so that the development could be kept modular and thus manageable, and so that the reader would be introduced to SDL in easier steps. If the reader feels that the introduction of SDL is too tedious, s/he may skip directly to Chapter 7 after reading Chapters 2 through 3.

The search space of sequences of optimal or even close to optimal actions in POMDPs is known to be intractable [Pineau et al., 2003]. One way to deal with the search space explosion in POMDPs is to use a particular class of algorithms which do not try to compute optimal policies (action strategies) offline. Instead, a policy is computed online by local look-ahead from the current belief-state up to a limited depth.

When the POMDP domain includes stochastic actions, nodes (representing belief-states) deeper in the search tree increase in the number of non-trivial states mentioned per node. Successor nodes in a search space require the application of a belief update procedure. Computation time of updating a belief-state is exponential in the number of states contained in the belief. We investigate methods to reduce the size of belief-nodes in the search tree, hence improving the running-time of online POMDP algorithms. This study of *belief-node condensation* is somewhat tangential to the presentation of the logics in the thesis. However, it is directly related to POMDPs, which is a thread running through the whole thesis.

To recapitulate, this thesis presents new formalisms for agents reasoning with stochastic actions and perceptions. In particular, the Stochastic Decision Logic (SDL) is developed to specify and reason about decision-theoretic domains where noisy observations lead to epistemic states represented as probability distributions over possible worlds. The semantics is based on the well-established partially observable Markov decision process (POMDP) theory. SDL allows for answering projection queries over incompletely specified POMDP models (equivalently, precisely specified classes of POMDP models), which, to our knowledge, is the first logic with this ability.

There are three more reasons why a logic like the SDL may be desirable. (i) People familiar with POMDP theory will find the thinking behind the SDL relatively obvious. (ii) POMDPs are well studied and accepted as a general purpose formalism for reasoning about stochastic domains. (iii) Having a logic based on POMDPs brings these benefits, plus the benefits of logical representation and reasoning, including the compactness of logical representation and the power of reasoning with semantic consequence.

The thesis is organized as follows. In each chapter defining a logic, a procedure is described for deciding what statements (expressed as sentences in the logic) formally follow or can be inferred from given information (also expressed as sentences of the logic). And for each logic, soundness, completeness and termination of the decision procedure is proved. And each chapter defining a logic, a framework for how the logic can be used in practice is presented. The research conducted for this thesis has resulted in several publications. Where applicable, a publication is cited in the introduction of the corresponding chapter.

In Chapter 2, we present a broad overview of notions and concepts often encountered when studying logics and formalisms for situated agents. The relevant POMDP theory is reviewed in Chapter 3. Chapter 4 presents the Logic of Actions and Observations (LAO). Chapter 5 present the

Specification Logic of Actions with Probability (SLAP). Chapter 6 present the Specification Logic of Actions and Observations with Probability (SLAOP), and Chapter 7 present the Stochastic Decision Logic (SDL). Then we review several logics and formalisms closely related to our work about formalisms for reasoning about stochastic agents and environments in Chapter 8. Here we highlighting what the related work lacks and how our work might attempt to address these deficiencies. We summarize what has been achieved in this thesis in the final chapter, and we point to directions in which the research can be continued and improved in future.

2. PRELIMINARY CONCEPTS

It is assumed that the reader has a good basic understanding of propositional logic and first-order logic. See, for instance, the books by Barwise and Etchemendy [1992] and Ben-Ari [2012].

In this chapter, we review several concepts and frameworks applicable to this thesis. These concepts and frameworks also serve as a primer for the next chapter, which reviews some related work and further motivates our work. We start with the basics of the *situation calculus* as a reference point for discussing logics for reasoning about action and change. Then, the famous frame problem [McCarthy and Hayes, 1969] in reasoning about action and change or commonsense reasoning is reviewed. Section 2.3 covers the basics of *modal* logic, in particular, *multi*-modal logic. Section 2.4 gives a brief reminder of decision processes relating to our work. Finally, we discuss some issues around the notions of uncertainty and nondeterministic action.

2.1 The Situation Calculus

The situation calculus was invented by McCarthy [1963]; however, we discuss the later formalism developed at the University of Toronto by Reiter, Levesque and others [Reiter, 2001, e.g.]. The situation calculus is a first-order logic (FOL) dialect for reasoning about dynamical systems based on agent actions. Two logics for reasoning about agent actions comparable with the situation calculus are the event calculus [Kowalski and Sergot, 1986] and the fluent calculus [Thielscher, 1999]. The situation calculus was chosen as a primer of action-based logics because of its popularity and because it is representative of the work in this area.

Actions and situations are *reified* to be objects in the language. The outcomes of a bout of reasoning in the situation calculus are meant to have an effect on the environment outside the agent. Predicates describe situations, in any situation, a predicate $F(\cdots, s)$ is either true of false, where s (a constant or a variable) is the name of some situation. The truth-value of $F(\cdots, s)$ depends on what situation s refers to.

When an agent or robot performs an action, the truth-value of certain predicates may change. A special function symbol do is defined in the situation calculus. $do(a, s)$ is the name of the situation that results from doing action a in situation s. Note that $do(a_2, do(a_1, s))$ is also a situation term, where a_2 and a_1 are actions.

Predicates and functions whose values can change due to actions are called *fluents*. Fluents have the *situation term* (like s or $do(\cdot, s)$, etc.) as the last argument. For example, if a robot is holding something, the fluent $Holding(r, x, s)$ (robot r is holding some x thing in situation s) is true, but when the robot does action $drop(r)$, $Holding(r, x, do(drop(r), s))$ should become false.

To reason in the situation calculus, one needs to define an initial knowledge-base (KB). The only situation term allowed in the initial KB is the special *initial situation* S_0. Constant S_0 is the situation before any action has been done.

There are three special types of formulae:[1]

1. *Precondition axioms* are formulae of the form $Poss(a, s)$, which means action a is possible in situation s ($\neg Poss(a, s)$ means it is not possible). Precondition axioms need to be defined for each action. For example, if a robot r_{33} has only one arm and gripper, and the gripper is already holding something (in situation s'), that is, if $Holding(r_{33}, stone, s')$ is true, then r_{33} cannot pick up something else; and then $\neg\exists x.Poss(pick\_up(r_{33}, x), s')$ is true. The precondition axiom for the action $pick\_up(r, x)$ could be defined as

$$Poss(pick\_up(r, x), s) \leftrightarrow \neg Holding(r, x', s) \wedge Near(r, x, s).$$

   That is, it is possible for robot r to pick x up in situation s if and only if r is not holding anything and the robot is near to x in s.

2. *Effect axioms* are formulae which describe how the world changes due to actions, that is, they specify the effects of actions. For instance,

$$Poss(pick\_up(r, x), s) \rightarrow Holding(r, x, do(pick\_up(r, x), s))$$

   means that if it is possible for robot r to pick up some x thing, then r will be holding it after performing the $pick\_up$ action. Another example is

$$Holding(r, x, s) \wedge Big(x) \wedge TileFloor \rightarrow FloorCracked(do(drop(r), s)). \qquad (2.1)$$

   Predicate symbols Big and $TileFloor$ are not fluents because their truth values are independent of situations. They are referred to as *rigid* in situation calculus parlance. Effect axiom 2.1 describes the effect of action $drop$ (a cracked floor) under the conditions that the robot who is doing the dropping is holding something big and is standing on a tile floor.

3. *Frame axioms*: Suppose that the robot is holding something which is small or that the floor is not tiled. Axiom 2.1 does not tell us the effect of action $pick\_up$ on fluent $FloorCracked$. To constrain or 'frame' the action's effects, one must provide a set of *frame axioms*. For every action a, for every fluent F, one axiom must be provided to specify: if F is true, the conditions under which F remains true when action a is executed, and another axiom must be provided to specify: if F is false, the conditions under which F remains false when action a is executed. The point is that if the truth value of a fluent doesn't change due to some action, then that action has no effect on the fluent. In that case, the unaffected fluent will not be mentioned in the effect axioms of the action. However, the knowledge that an action has no effect on a fluent must be stated in an agent's KB for the agent to know it. This is the role of frame axioms. Picking something up has no effect on a floor:

$$\begin{aligned} FloorCracked(s) &\rightarrow FloorCracked(do(pick\_up(r, x), s)) \\ \neg FloorCracked(s) &\rightarrow \neg FloorCracked(do(pick\_up(r, x), s)). \end{aligned}$$

[1] As is convention in the situation calculus, free variables are assumed universally quantified.

A negative frame axiom for $drop$ with respect to $FloorCracked$ is

$$\neg FloorCracked(s) \wedge (\neg Big(x) \vee \neg TileFloor) \rightarrow \neg FloorCracked(do(drop(r), s))$$

and a positive axiom is

$$FloorCracked(s)) \rightarrow FloorCracked(do(drop(r), s)).$$

2.2 The Frame Problem

The three most well known issues in knowledge representation in physically dynamic domains are (i) the qualification problem, (ii) the ramification problem and (iii) the frame problem.

The qualification problem appears when specifying when an action is possible. There seems to be an infinite number of features in the real world which one must assign a value to before the action can be said to be possible or impossible to execute. Fortunately, this problem is not so big in synthetic, structured or finite domains.

The ramification problem occurs when it is hard or impossible to specify all effects—direct or indirect—an action may have on the environment. Ramifications are dealt with implicitly for each of the logics presented in this thesis, when we explain how the respective logics are used.

Of the three, the frame problem [McCarthy and Hayes, 1969] seems to be the hardest to solve or has the worst effect on the amount of computation involved when reasoning with knowledge about actions: Typically, any one action has a limited effect on the world. One has to somehow state which features are not affected (i.e., stating that the rest of the world is not affected) to determine what has changed and what not, due to the execution of the action. Having to write a statement for every feature that is invariant with an action, for every action, seems counter-intuitive and there is potentially a huge number of such statements. In terms of the situation calculus, approximately $2 \times \mathcal{A} \times \mathcal{F}$ frame axioms are required, where $\mathcal{A}$ is the number of actions, $\mathcal{F}$ is the number of fluents and the 2 is for positive and negative axioms.

Reiter [1991] famously provided a solution to the frame problem for the situation calculus. He suggested *successor-state axioms* (SSAs)—formulae with a special form—to solve the problem, which is now sketched.

An SSA for action a combines all the effect axioms of a and the information captured by all the frame axioms involving a. In fact, frame axioms for an action are another way of stating under which conditions a fluent changes value [Brachman and Levesque, 2004, p. 292]. In other words, instead of writing frame axioms, one can make the following completeness assumption about the effect axioms. All the conditions under which fluent F can change value according to the effect axioms for action a, are the only conditions under which F can change value due to a. This is also known as the *explanation closure* of the effects of a on F.

There needs to be a successor-state axiom for each fluent, and each such successor-state axiom mentions only the actions that have an effect on the particular fluent. Suppose there is one more action in the vocabulary introduced in the previous section: $step\_forward(r, s)$, meaning that robot r steps one step forward. We could define the following successor-state axiom for the fluent

$Holding(r, x, s)$:

$$Holding(r, x, do(a, s)) \leftrightarrow Poss(a, s) \wedge \exists x.a = pick\_up(r, x) \vee Holding(r, x, s) \wedge a \neq drop(r, x).$$

In other words, the robot is holding something if and only if it picks up something (if it is not already holding something) or it does not drop what it is already holding. Note that $step\_forward(r)$ is not mentioned in the above formula because it does not have an effect on the value of $Holding$. Moreover, instead of approximately $\mathcal{A}$ (effect axioms) plus $2 \times \mathcal{A} \times \mathcal{F}$ frame axioms, there are only approximately $\mathcal{F}$ SSAs.

Some non-situation-based logics have solutions to the frame problem, sometimes quite different to Reiter's approach. The event and fluent calculi, for instance, have their own solutions. And the logic $\mathcal{LAP}_{\rightsquigarrow}$ [Castilho et al., 1999] reviewed in Chapter 8 makes use of so-called semantic *dependence relations*. Most logics extending or employing the situation calculus, rely on Reiter's solution or a variant thereof. An adequate 'frame solution' for our logic SLAP is presented in Chapter 5, and it is applicable to SLAOP and the SDL too. The solution does not use SSAs nor dependence relations.

Let $\mathcal{D} = \Sigma \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$, where

- Σ is the four foundational axioms for situations [Reiter, 2001, Section 4.2];
- $\mathcal{D}_{ss}$ is all successor state axioms;
- $\mathcal{D}_{ap}$ is all action precondition axioms,
- $\mathcal{D}_{una}$ is the set of unique names axioms for actions;
- $\mathcal{D}_{S_0}$ is the set of formulae specifying the initial situation.

Then $\mathcal{D}$ together with a formula that captures the *functional fluent consistency property* (refer to [Reiter, 2001, p. 60] for details) is a *basic action theory*. Reasoning with a basic action theory has certain desirable properties. Please refer to Reiter [2001] for a detailed explication of his version of the situation calculus. Alternatively, refer to Brachman and Levesque [2004] for a one-chapter coverage of the situation calculus.

2.3 Multi-modal Logics

Modal logic [Hughes and Cresswell, 1996, Chagrov and Zakharyaschev, 1997, Blackburn et al., 2001, 2007] is considered to be well suited to reasoning about beliefs and changing situations. Modal logic was originally developed in philosophy to talk about notions with a 'modal' character such as "...time, space, obligation, conditionality, knowledge, computation, and action..." [Blackburn et al., 2007, Preface]. The semantics of modal logic can be interpreted as a graph representing "... flows of time, relations between epistemic alternatives, transitions between computational states, networks of possible worlds ..." [Blackburn et al., 2007, Chap. 1]. Modal logic allows for the relatively compact and intuitive expression of a graph-like system where edges represent actions and vertices represent epistemic alternatives (belief-states). Our aim in this thesis

to develop a formalism with action transitions between epistemic or belief states makes modal logic a good candidate.

The basic modal logic is classical propositional logic with the addition of a *monadic* or *unary* operator $\Box$. The grammar of the basic modal logic, as defined in Backus-Naur Form (BNF), is then

$$\phi ::= p \mid \neg\phi \mid \phi \wedge \phi' \mid \Box\phi,$$

where p is a propositional atom from a set of propositional atoms $\mathcal{P} = \{p_1, p_2, \ldots\}$, $\neg$ is read 'not' and $\wedge$ is read 'and'. Abbreviation $\vee$ (for 'or') and $\rightarrow$ (for 'implies') are defined as usual. The *verum* ($\top$, truth) and *falsum* ($\bot$, falsehood) symbols may also be defined. Let $\mathcal{L}$ be the language of basic modal logic, such that $\phi \in \mathcal{L}$ for all formations of ϕ.

Depending on the purpose for which the modal logic will be used, the sentence $\Box\phi$ may have different interpretations in English, but the general 'feeling' or 'mode' of the sentence is the same. $\Box\phi$ may be interpreted as 'It is necessarily the case that ϕ', 'ϕ is obliged to be the case', 'It is inevitable that ϕ', and 'ϕ has to be true'.[2]

'It is not the case that ϕ is necessarily false' can be written as $\neg\Box\neg\phi$. But this is the same as saying 'It is possibly the case that ϕ'. It is thus convention to define a unary operator $\Diamond$ as follows: $\Diamond\phi$ abbreviates $\neg\Box\neg\phi$, usually read 'possibly ϕ'. $\Box$ and $\Diamond$ are called modal operators.

To provide a semantics for $\mathcal{L}$, one can define a *structure* $\mathcal{M} = \langle W, R, I\rangle$, where W is a nonempty set of *worlds*, R is an *accessibility* relation between worlds and $I : \mathcal{P} \rightarrow 2^W$ an *interpretation* function (also called a *valuation* function) that determines for each propositional atom, in which worlds it is true.[3] Formally, any sentence $\phi \in \mathcal{L}$ is *satisfied* in structure $\mathcal{M}$ at worlds $w \in W$ (denoted $\mathcal{M}, w \models \phi$) when:

- $\mathcal{M}, w \models p \iff w \in I(p)$;
- $\mathcal{M}, w \models \neg\phi \iff \mathcal{M}, w \not\models \phi$;
- $\mathcal{M}, w \models \phi \wedge \phi' \iff \mathcal{M}, w \models \phi$ and $\mathcal{M}, w \models \phi'$;
- $\mathcal{M}, w \models \Box\phi \iff$ for all w, if $R(w, w')$, then $\mathcal{M}, w' \models \phi$.

A formula ϕ is said to be *valid* in a structure (denoted $\mathcal{M} \models \phi$) if $\mathcal{M}, w \models \phi$ for every $w \in W$.

This is the so-called *possible worlds* semantics of modal logics. The intuition behind this semantics is that each world is an alternate state that the world can be in. If an agent maintains that it is in world w (the state of the world), the agent may, according to its knowledge, also be in world w' (a different state of the world) because of some information that the agent is missing.

The above is an *epistemic* view of possible worlds semantics. The connection between worlds is formally captured with the relation R. $R(w, w')$ in the epistemic view can be read 'Mental state w' is (epistemologically) accessible from mental state w.' Therefore, if an agent is said to be in a world w, then all worlds accessible from w are also possible, according to the agent.

We formally define when one sentence logically follows a set of sentences with the notion of *semantic consequence*. We shall employ two kinds of semantic consequence with respect to modal

[2] $\Box$ is one conventional symbol, but theoretically, any symbol can be used, for instance, K.

[3] Kripke [1959] first suggested and subsequently developed *possible world semantics* via such structures.

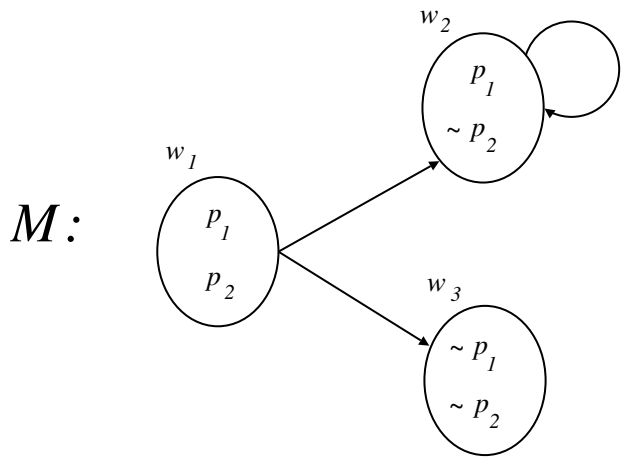


Fig. 2.1: Example structure in modal logic. (In this figure, $\sim$ denotes $\neg$.)

logic: local and global. Let $\mathcal{K} \subseteq \mathcal{L}$ be a set of sentences.

Local semantic consequence: ϕ is a local semantic consequence of $\mathcal{K}$ if and only if, for every structure $\mathcal{M}$, for every world $w \in W$ of $\mathcal{M}$, if $\mathcal{M}, w \models \kappa$ for every $\kappa \in \mathcal{K}$, then $\mathcal{M}, w \models \phi$.

Global semantic consequence: ϕ is a global semantic consequence of $\mathcal{K}$ if and only if, for every structure $\mathcal{M}$, if $\mathcal{M} \models \kappa$ for every $\kappa \in \mathcal{K}$, then $\mathcal{M} \models \phi$.

To the reader unfamiliar with the two kinds of semantic consequence, the two definitions might look the same, only with different wording. They are, in fact, different; it could happen that ϕ' is a global semantic consequence of $\mathcal{K}$, while it is not a local semantic consequence of $\mathcal{K}$. (But if ϕ' is a local semantic consequence of $\mathcal{K}$, then it must be a global semantic consequence of $\mathcal{K}$.) Intuitively, local consequence centres around truth at worlds, while global consequence centres around truth in structures.

In the field of logic-based knowledge representation and reasoning, the term *logical entailment* is also used. In this thesis, the two terms are used interchangeably. And if we say that $\mathcal{K}$ *entails* ϕ, we mean that $\mathcal{K}$ has ϕ as a semantic consequence.

Suppose we have propositions p_1, p_2, and a structure M with worlds w_1, w_2, w_3, the relation $R = \{(w_1, w_2), (w_1, w_3), (w_2, w_2)\}$ and the interpretation $I(p_1) = \{w_1, w_2\}$, $I(p_2) = \{w_1\}$. Then Figure 2.1 depicts M graphically. One can determine that in this structure

- $M, w_1 \models p_2$
- $M, w_1 \models \Box\neg p_2$
- $M, w_1 \not\models \Box\neg p_1$
- $M, w_2 \not\models \Diamond p_2$
- $M, w_2 \models \neg\Diamond p_2$
- $M, w_1 \models \Diamond\Diamond p_1$

An *ontic* view of the possible worlds semantics interprets the accessibility relation as the worlds that are physically accessible through *action*.[4] For example, in the ontic view, a world $w_{stepped}$ where a robot is one step farther on than it was, is accessible from a world $w_{current}$ where the robot has not yet taken a step forward. Moreover, in this example, $(w_{current}, w_{stepped})$ should be an element of R only if the robot has a 'step-forward' action available to it and it is possible for the robot to perform that action in the current world.

Any agent or dynamical system typically has more than a single action that it can perform. The specification of such a 'multi-action' system using a modal logic, requires a *multi-modal* logic.

Besides actions, a knowledge engineer may want to talk about several subsystems at once, for example several agents in a system, or the engineer may want to capture different epistemic modes of one system. In general, if a knowledge engineer wants to talk about k modes, s/he will need k different modal operators $\Box_1, \ldots, \Box_k$. There is a dual $\Diamond_\ell$ for each $\Box_\ell$. In multi-modal logics about change, that is, in logics about dynamical systems with multiple modes, it is convention to write $[\gamma_1], \ldots, [\gamma_k]$ and $\langle\gamma_1\rangle, \ldots, \langle\gamma_k\rangle$ for the box and diamond operators respectively, where the γ_ℓ are the different actions or events or whatever is required for the system being reasoned about.

One multi-modal logic originally developed for program verification in computer science is *propositional dynamic logic* (PDL) [Harel et al., 2000]. Informally, PDL is defined as follows. Suppose p is an atomic proposition and a is an atomic program, then ϕ is a proposition, α is a compound program and Φ is a formula:

$$\begin{aligned}
\phi &::= p \mid \neg\phi \mid \phi \wedge \phi' \\
\alpha &::= a \mid \alpha; \alpha' \mid \alpha \cup \alpha' \mid \alpha^* \mid \psi? \\
\psi &::= p \mid \neg\psi \mid \psi \wedge \psi' \mid [\alpha]\psi.
\end{aligned}$$

$[\alpha;\alpha']\psi$ is an abbreviation for $[\alpha][\alpha']\psi$ and $\alpha;\alpha'$ intuitively means 'execute α, then execute α''. $[\alpha \cup \alpha']\psi$ is an abbreviation for $[\alpha]\psi \wedge [\alpha']\psi$ and $\alpha \cup \alpha'$ intuitively means 'choose either α or α' nondeterministically, and execute it'. $[\psi?]\psi'$ abbreviates $\psi \rightarrow \psi'$ and $\psi?$ intuitively means 'Test the truth of ψ; proceed if true, fail if false'. $[\alpha^*]\psi$ does not have a straight-forward abbreviation (see, e.g., Harel et al. [2000])—its intuitive meaning is 'ψ is true after executing program α zero or an arbitrary number of times'.

De Giacomo and Lenzerini [1996], and Prendinger and Schurz [1996], for instance, have presented PDL-based frameworks for specifically reasoning about action and change in the context of *agents* and not programs. In these frameworks, programs are viewed as complex agent actions. Meyer [2000] discusses the uses of dynamic logic for reasoning about actions and agents.

For good introductory textbooks on modal logic, please refer to Hughes and Cresswell [1996] and Chellas [1980]. Books by Chagrov and Zakharyaschev [1997], Blackburn et al. [2001] and Blackburn et al. [2007] are also standard references. Most of the logics reviewed in Chapter 8 have a modal component.

[4] Informally, something is *ontic* when it is physical and *epistemic* when it is mental.

2.4 Decision Procedures

Ultimately, the logics we are interested in in this thesis are to be used by agents to decide what actions to take next. Such decisions typically hinge on whether or not a particular conjectured statement is true, given the agents' background knowledge. Suppose $\Psi \in \mathcal{L}$ is a sentence representing a particular conjecture, where $\mathcal{L}$ is some logical language. Suppose further that $BK \subset \mathcal{L}$ is a set of sentences representing an agent's background knowledge. Then we would like to determine whether Ψ logically 'follows' from BK. In this thesis, we use local or global semantic consequence to determine this.

To actually determine whether some Ψ is a semantic consequence of BK, some method, algorithm or procedure must be employed. Such procedures usually assume the language/vocabulary as known. A procedure should take Ψ and BK as input and produce 'yes' or 'no' as output, depending on whether BK does or does not entail Ψ.

There are various decision procedures for entailment in formal logic. Different procedures are better suited to different purposes, for instance, for making inferences in logic-based databases, for making inferences with knowledge-bases written in particular logics, or for proving certain properties of various logics. When it comes to logics for reasoning about action and change, decision procedures with particular characteristics are designed and used. Next, we discuss one popular procedure for the situation calculus. Then we outline the general procedure used to decide entailment in the logics presented in this thesis, that is, the *tableau calculus* approach.

Since Reiter [2001] invented successor-state axioms for the situation calculus, the *regression* method proved more efficient than classical decision procedures. Informally, regression works as follows.

Assume there is a successor-state axiom in the theory Θ for each distinct fluent in the language $\mathcal{L}$. Part of the definition for the *regression operator* $\mathcal{R}_\Theta$ for Θ is: When F is a fluent whose successor-state axiom in Θ is

$$(\forall a, s)(\forall x_1, \ldots, x_n) Poss(a, s) \rightarrow F(x_1, \ldots, x_n, do(a, s)) \leftrightarrow \Phi_F,$$

then

$$\mathcal{R}_\Theta[F(t_1, \ldots, t_n, do(\alpha, \sigma))] = \Phi_F|^{x_1,\ldots,x_n,a,s}_{t_1,\ldots,t_n,\alpha,\sigma},$$

where $t_1, \ldots, t_n$ are terms in $\mathcal{L}$ and α and σ are respectively an action term and a situation term in $\mathcal{L}$. "$\mathcal{R}_\Theta[G]$ is simply that formula obtained from G by substituting suitable instances of Φ_F in F's successor-state axiom for each occurrence in G of a fluent atom of the form $F(t_1, \ldots, t_n, do(\alpha, \sigma))$," [Reiter, 1991, p. 16]. The idea behind the regression operator is to reduce the nesting of the do symbol in the fluents appearing in G—if s appears in $do(a, s)$ on the left-hand side a successor-state axiom, the it always appears un-nested on the right-hand side of the axiom. Regression continues until do no longer appears in $\mathcal{R}_\Theta[\mathcal{R}_\Theta[\cdots \mathcal{R}_\Theta[G] \cdots]]$ and the only situation term is S_0. It is then sufficient to consult only facts in the initial knowledge-base to determine whether $\Theta \models G$, broadly speaking.

But successor-state axioms are not definable for all kinds of logics for reasoning about dynamical systems. This is especially the case when actions or events in the system have nondeterministic

effects.

The *tableaux* approach [Fitting, 1999] as a deductive system has proven well suited to modal logics in general [Goré, 1999]. It can also be applied in classical logics [Ben-Ari, 2012]. Informally, a tableau is a tree-like structure that is 'grown' by expanding the tree by adding branches for each application of a rule. The 'trunk' of a tableau tree is some tuple involving a logical formula. If a proof of validity for formula Φ is sought, then Φ in the trunk should be the negated or labeled FALSE.[5] Each path through a tree represents a (potential; partial) model (i.e., satisfying structure). If the conjunction of the formulae on one path form a contradiction, that path is not a model. If on some path, no contradiction can be generated by the rules of expansion, that path represents a model. A path which supports a contradiction, is called *closed*. A path which is not closed is called *open*. If all paths of a tree are closed, it means that no model for the trunk exists; the trunk is *unsatisfiable*. In this case, the negation of the trunk is valid. If no more rules can be applied to any formula on a path, and the path is open, then this path represents a model for the trunk, or, the trunk is *satisfiable*.

Terminology differs when talking about tableaux. Actually, the word "path" is not used; instead "branch" is used. In the explanation above, path was used because branch carries a meaning more difficult to use informally. Some people refer to *trees* instead of *branches*, and sometimes one may encounter a *forest* when one expects a *tree*.

A tableau *calculus* is a definition of the kinds of structures, rules and processes allowed. Different logics may employ the same tableau calculus but implement these calculi differently. In this thesis, we say that a particular instance of a tableau calculus is a tableau *method*. The different calculi used for the different logics presented in this thesis all use a *labeled formula* notation, that is, sets of formulae labeled with some meta-information are maintained. The calculi of the four logics are almost identical; in fact, the tableau calculi used with our SLAP and SLAOP are identical (although their tableau methods are slightly different). The calculus used with our LAO is closest to the one used with the logic $\mathcal{LAP}$ [Castilho et al., 1999] (cf.. § 8.2). Both LAO and $\mathcal{LAP}$ use a 'skeleton' structure in their calculi. The SLAP/SLAOP calculus does not use skeletons, because they need not keep track of nested operators. Our SDL makes use of a kind of skeleton called an *activity sequence*.

We now provide the SLAP/SLAOP tableau calculus. We shall indicate when some terminology or elements is common to all calculi used in this thesis. The foundation calculus is adapted from Castilho et al. [1999]. It is based on labeled formulae.

Definition 2.4.1: A *labeled formula* is a pair (x, Ψ), where $\Psi \in \mathcal{L}_{SLAP}$ or $\Psi \in \mathcal{L}_{SLAOP}$ is a formula and x is an integer called the *label* of Ψ.

Definition 2.4.2: A *node* Γ_k^j with superscript j (the *branch* index) and subscript k (the *node* index), is a set of labeled formulae.

Definition 2.4.3: For all calculi, the initial node, that is, Γ_0^0, to which the tableau rules must be applied, is called the *trunk*.[6]

[5] Formulae may be labeled with some semantic information depending on the particular tableau calculus, or formulae may be unlabeled.

[6] A trunk is also the root of a graph which is a tree.

Definition 2.4.4: For all calculi, a *tree* T is a set of nodes. A tree must include Γ_0^0 and only nodes resulting from the application of *tableau rules* to the trunk and subsequent nodes.

Definition 2.4.5: If one has a tree with trunk $\Gamma_0^0 = \langle\{(0, \Psi)\}, \emptyset\rangle$, we shall say one has *a tree for* Ψ.

For all calculi, when we say '...where x is a fresh integer', we mean that x is the smallest positive integer of the right sort (formula label or branch index) not yet used in the node to which the incumbent tableau rule will be applied.

For all calculi, a tableau rule applied to node Γ_k^j creates one or more new nodes; its child(ren). If it creates one child, then it is identified as Γ_{k+1}^j. If Γ_k^j creates a second child, it is identified as $\Gamma_0^{j'}$, where j' is a fresh integer. That is, for every child created beyond the first, a new branch is started.

Definition 2.4.6: For all calculi, a node Γ is a *leaf* node of tree T if no tableau rule has been applied to Γ in T.

Definition 2.4.7: For all calculi, a *branch* is the set of nodes on a path from the trunk to a leaf node.

Note that nodes with different branch indexes may be on the same branch.

Definition 2.4.8: For all calculi, Γ is *higher* on a branch than Γ' if and only if Γ is an ancestor of Γ'.

We assume that it is implicit in all calculi that (i) a tableau rule may only be applied to an open leaf node and (ii) a tableau rule may not be applied to a formula if it has been applied to that formula higher in the tree, as defined in Definition 2.4.8. The second constraint prevents trivial re-applications of rules. For example, if rule $\Box$ were applied to $(0, \Box f_1) \in \Gamma_3^2$, then it may not be applied to $(0, \Box f_1) \in \Gamma_4^2$.

Definition 2.4.9: A node Γ is *closed* if $(x, \bot) \in \Gamma$ for any $x \geq 0$. It is *open* if it is not closed. A branch is closed if and only if its leaf node is closed. A tree is closed if all of its leaf nodes are closed, else it is open.

Definition 2.4.10: For all calculi, a branch is *saturated* if and only if any rule that can be applied to its leaf node has been applied. A tree is *saturated* if and only if all its branches are saturated.

Part of a tableau tree discussed in Chapter 5 is reproduced here (Fig. 2.2) to give the reader an idea of the structure of a tableau tree. Please ignore the details in the figure for now.

2.5 Uncertainty and Nondeterministic Action

There are at least four kinds of uncertainty an agent may have:

1. about its state, that is, the agent may not know the status of every attribute of its world;
2. about the effect or outcome an action may have, that is, the agent cannot predict the effect of some actions with certainty (*nondeterministic/stochastic effect* of an action);
3. about the accuracy of exogenous events, that is, the agent may perceive some events (observations) in nature incorrectly or partially;

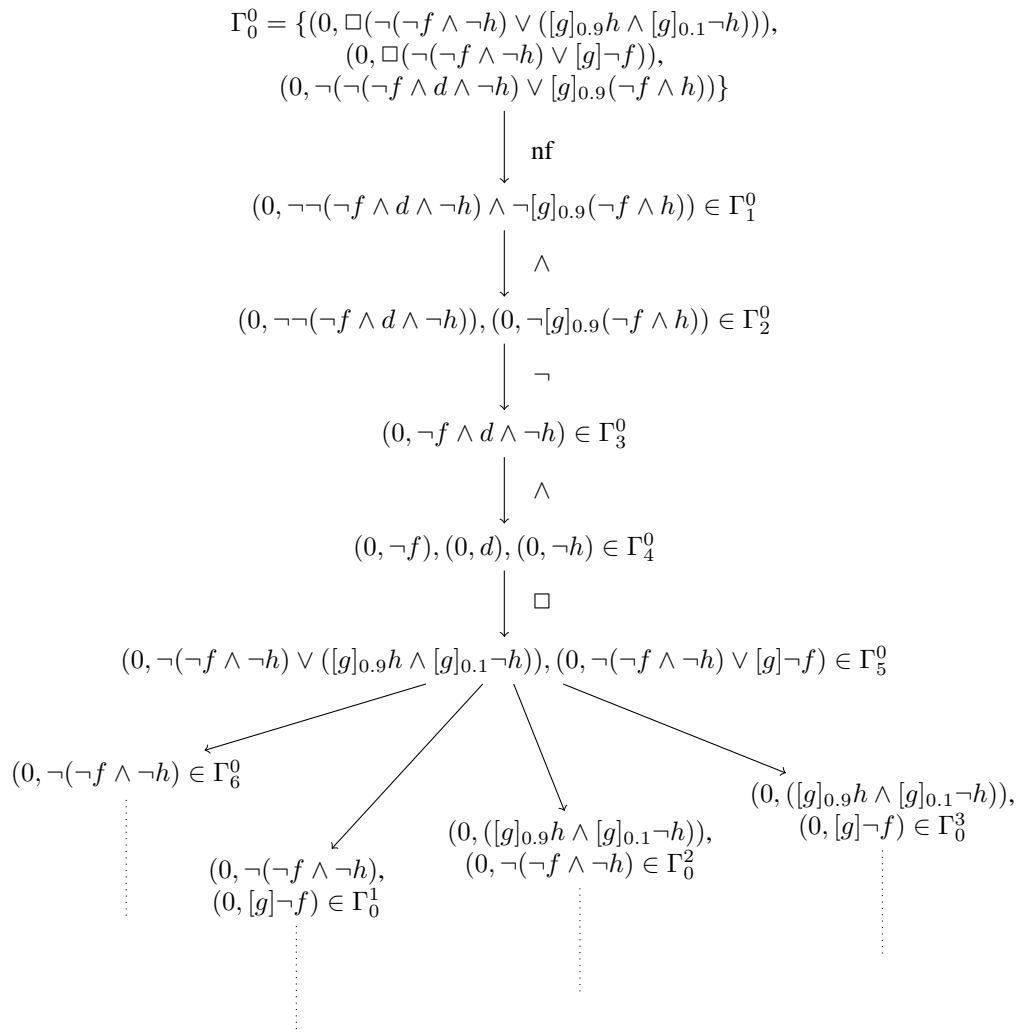


Fig. 2.2: Example of the first part of a tableau tree for $\Box(\neg f \wedge \neg h \rightarrow ([g]_{0.9} h \wedge [g]_{0.1} \neg h)) \wedge \Box(\neg f \wedge \neg h \rightarrow [g]\neg f) \wedge \neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h))$.

4. about which action the agent will perform, when some action is chosen to execute (*nondeterministic/stochastic choice* between actions).

One way to express these uncertainties is with disjunction:

1. The agent may know that the door is open, but all that it knows about the light is that it is on or off, but not which.
2. When the agent attempts to close the door, it might close completely or almost or only slightly.
3. Seeing that the light is off might mean that it is off or it is on (but just seems off because it is a low-wattage bulb and the sun is shining directly on the bulb).
4. When the agent intends to perform $move\_in\_direction\_10^\circ$, it will actually perform $move\_in\_direction\_12^\circ$ or $move\_in\_direction\_11^\circ$ or $move\_in\_direction\_10^\circ$ or $move\_in\_direction\_9^\circ$ or $move\_in\_direction\_8^\circ$.

Suppose that the uncertainty the agent has about which action it will perform (case 4) concerns closing the door rather than the direction of its movement. That is, suppose that when the agent wants to close the door, it might perform the action $completely\_close\_door$ or $almost\_close\_door$ or $only\_slightly\_close\_door$. Now we see that cases 2 and 4 are very closely related. In fact, cases like 2 can be modeled with the approach of case 4: Assuming that actions $completely\_close\_door$, $almost\_close\_door$ and $only\_slightly\_close\_door$ are deterministic, whenever the agent wants to $close\_door$, let it perform $completely\_close\_door$ or $almost\_close\_door$ or $only\_slightly\_close\_door$. Because the latter three actions are deterministic, the effects of performing $close\_door$ are 'guaranteed' to be that it closes completely or almost or only slightly. When uncertainty between actions (case 4) is used to simulate uncertainty in effects (case 2), it is said that *action decomposition* is being employed [Bacchus et al., 1999], [Reiter, 2001, Sec. 12.1].

The new logics presented in this thesis only model uncertainty in effect directly, not via action decomposition.

We use the convention that uncertainty expressed by disjunction is called *nondeterministic* uncertainty (also called *qualitative* uncertainty). A more sophisticated way to express uncertainty is by probability. Alternatives can be expressed more accurately; likelihoods of different possibilities can be expressed exactly to reflect the known facts. However, confidence in the known facts (about likelihoods) may be low, in which case it may be prudent to express uncertainty by disjunction or even by fuzzy set theory. In this work, we shall always assume that the known probabilities are known with high enough confidence that they can be used for modeling purposes. We use the convention that uncertainty expressed by probability is called *stochastic* uncertainty (also called *quantitative* uncertainty), although we use the word *probabilistic* too.

The four cases above could be expressed with probability as follows.

1. The agent may know that the door is open, but all that it knows about the light is that it is on with a probability of 0.6 and off with a probability of 0.4.
2. When the agent attempts to close the door, it might close completely with a probability of 0.33, almost with a probability of 0.33 and only slightly with a probability of 0.34.

3. Seeing that the light is off might mean that it is off with a probability of 0.9, but it is on with a probability of 0.1.

4. When the agent intends to perform $move\_in\_direction\_10^\circ$, it will actually perform $move\_in\_direction\_12^\circ$ with probability 0.1, $move\_in\_direction\_11^\circ$ with probability 0.2, $move\_in\_direction\_10^\circ$ with probability 0.5, $move\_in\_direction\_9^\circ$ with probability 0.2 and $move\_in\_direction\_8^\circ$ with probability 0.1.

Stochastic effect can also be simulated by stochastic choice with deterministic effect. However, we take the direct approach in this work.

In the next chapter, we review the relevant theory of partially observable Markov decision processes (POMDPs). This will help the reader to understand the motivation for the semantics of our logics, especially the definitions of the Specification Logic of Actions and Observations (SLAOP) and the Stochastic Decision Logic (SDL), presented in the latter half of the thesis.

3. PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES

A *partially observable Markov decision process* (POMDP) [Aström, 1965, Smallwood and Sondik, 1973, Monahan, 1982, Lovejoy, 1991] is a generalization of the Markov decision process (MDP) [Bellman, 1957, Howard, 1960, Puterman, 1994]. The semantics of SLAP (Chap. 5) are based on MDPs, but the semantics of SLAOP (Chap. 6) and the SDL (Chap. 7) are based on POMDPs. The POMDP formalism is well suited to representing a class of domains that occur frequently in practice, including robotics [Kaelbling et al., 1998, Boutilier et al., 1999, Russell and Norvig, 2003]. Our work focuses on providing high-level decision-making capabilities for situated agents who live in dynamic environments with time constraints for planning. One solution is to employ a *continuous planning* strategy, or *agent-centered search* Koenig [2001]. Aligned with agent-centered search is the *forward-search* approach or *online* plannning approach in POMDPs [Ross et al., 2008]. This is the approach we use.

In partially observable Markov decision processes, actions have nondeterministic results as in (fully observable) MDPs, but observations are uncertain. In other words, the effect of some chosen action is somewhat unpredictable, yet may be predicted with a probability of occurrence. However, in POMDPs, the world is not directly observable: some data are observable and the agent infers how likely it is that the world is in some particular state. The agent thus believes to some degree—for each possible state—that it is in that state, but it is never certain exactly which state it is in. In fact, the agent maintains a probability distribution over the states reflecting its conviction for being in each state.

Deciding which actions to take depends on the utility of the actions, conditioned on the states in which they are performed. But the decision process is complicated due to the agent's uncertainty about its state.

The theory in this chapter can be found in the papers of Kaelbling, Cassandra and Littman [Cassandra et al., 1994, Kaelbling et al., 1998], for example.

Section 3.1 covers the basic theory of POMDPs required for this thesis. Planning with POMDPs is discussed in Section 3.2. In Section 3.3, we present a graphical representation of the planning problem, which is an aid in understanding the process. Section 3.4 details two particular tasks that can be performed on a POMDP. These two tasks are directly related to two operators in our logic (the SDL) defined in Chapter 7.

3.1 Basic Theory

Formally, a POMDP is a tuple $\langle S, A, T, R, Z, O, b^0 \rangle$ where

- $S = \{s_1, s_2, \ldots, s_n\}$ is a finite set of states of the world (which the agent can be in);
- $A = \{a_1, a_2, \ldots, a_k\}$ is a finite set of actions (which the agent can choose to execute);
- $T : \mathcal{S} \times A \times S \rightarrow (\mathbb{R} \cap [0, 1])$ is the *state-transition function*; $T(s, a, s')$ denotes the probability of being in s' after performing action a in state s;
- $R : A \times S \rightarrow \mathbb{R}$ is the *reward function*; $R(a, s)$ is the immediate reward gained for executing a while in state s;
- $Z = \{z_1, z_2, \ldots, z_m\}$ is a finite set of observations the agent can perceive in its world;
- $O : S \times A \times Z \rightarrow (\mathbb{R} \cap [0, 1])$ is the *observation function*; $O(s', a, z)$ denotes the probability of observing z in state s' resulting from performing action a in some other state;
- b^0 is the initial probability distribution over all states in S.

Let b be a total function from S into $\mathbb{R}$. Each state s is associated with a probability $b(s) = p \in \mathbb{R}$, such that b is a probability distribution over the set $\mathcal{S}$ of all states. b can be called a *belief-state*.

An important function in POMDP theory, is the function that updates the agent's belief-state, or the *state estimation* function SE. $SE(a, z, b) = b_n$ is defined as

$$b_n(s') = \frac{O(s', a, z) \sum_{s \in S} T(s, a, s') b(s)}{Pr(z \mid a, b)}, \tag{3.1}$$

where $b_n(s')$ is the probability of the agent being in state s' in the 'new' belief-state b_n, relative to a, z and the 'old' belief-state b.

$$Pr(z \mid a, b) = \sum_{s' \in \mathcal{S}} O(s', a, z) \sum_{s \in \mathcal{S}} T(s, a, s') b(s) \tag{3.2}$$

in the denominator acts as a normalizer here. Equation (3.1) is derived from the Bayes Rule. Notice that $SE(\cdot)$ requires a belief-state, an action and an observation as inputs to determine the new belief-state.

When the states an agent can be in are *belief*-states (as opposed to objective, single states in S), the reward function R must be lifted to operate over belief-states. The *expected* reward $\rho(a, b)$ for performing an action a in a belief-state b is defined as

$$\rho(a, b) \stackrel{def}{=} \sum_{s \in S} R(a, s) b(s). \tag{3.3}$$

The notion of a *policy* captures what an agent should do or is instructed to do. There are several definitions for what a policy is; we mention first the traditional definition and then a definition more suited to our work. Formally, a *policy* π, in traditional POMDP theory, is a function from a set of belief-states $\mathcal{B}$ (all those the agent can be in) to a set of actions:

$$\pi : \mathcal{B} \mapsto A. \tag{3.4}$$

That is, actions are *conditioned* on beliefs. Therefore, for any belief-state the agent might find itself in, it needs only consult its policy and it will know what to do next. The belief-state space

is typically huge and determining a 'good' policy of this kind is impractical for agents who must plan online. Our work is focused on agents who plan online.

Our approach is that of *forward-search* or *online* POMDP plannning [Ross et al., 2008]. An agent looks ahead a few steps (typically less than ten) and determines a policy only for those steps and only for the current belief-state. Actions are thus conditioned on the steps to go and not on belief-states. This strategy is much more tractable than finding the best action for every possible belief-state, for all future actions. At the end of this section, we further motivate our choice of the forward-search approach.

Let the *planning horizon* h be the number of steps into the future that the agent will consider each time it selects its next few actions; h can also be called the *look-ahead depth*; in the recursive equations below, h can be thought of as the number of *steps to go*.

A policy with a planning horizon h is defined as a set of h pairs (h', a) where h' is the number of steps to go and a is the action the agent will/should take. In other words,

$$\pi = \{(h, a^1), (h-1, a^2), (h-2, a^3), \ldots, (1, a^h)\}.^1$$

Theoretically, the agent could determine a policy for h actions and perform any of the first $g \leq h$ actions recommended by the policy. The agent could then determine a new policy for h steps/actions, perform the first g, and so on.

Suppose the agent has performed $k-1$ steps. Then from step k onwards, the agent gets the sequence of rewards

$$r_k, r_{k+1}, r_{k+2}, \ldots$$

for steps $k, k+1, k+2, \ldots$, respectively. It is convention to deem a reward that can be obtained farther into the future as less valuable than a reward obtained earlier. Later rewards are thus 'discounted' by a *discount factor* denoted by γ $(0 < \gamma \leq 1)$.[2]

The discounted rewards V_k from step k onwards, is defined as

$$\begin{aligned} V_k &= r_k + \gamma(r_{k+1} + \gamma(r_{k+2} + \cdots)) \\ &= r_k + \gamma V_{k+1}. \end{aligned} \tag{3.5}$$

That is, the agent receives the immediate reward plus the sum of discounted future rewards.

The notion of the *value* of a belief-state, given a policy, brings together all the theory discussed so far: The *value* $V^\pi(b, h)$ of a belief-state b is the expected value of future states, given the actions selected at each step using policy π, until the horizon h is reached. In the equations below, $Pr(z \mid a, b)$ can be viewed as the probability of reaching the next belief-state $b_n = SE(a, z, b)$.

$$V^\pi(b, h) \stackrel{def}{=} \rho(\pi(h), b) + \gamma \sum_{z \in Z} Pr(z \mid \pi(h), b) V^\pi(SE(\pi(h), z, b), h-1)$$

$$V^\pi(b, 1) \stackrel{def}{=} \rho(\pi(1), b).$$

[1] Superscripted action and observation symbols indicate their chronological position in a sequence, whereas subscripted symbols refer to different members of A and O, respectively.

[2] The discount factor is especially convenient for mathematical analysis of infinite sequences of actions.

The rewards r_h in (3.5) become the believed or expected rewards $\rho(\pi(h), b)$ in the definition of the value function above.

It is convenient to know the value of belief-state b with respect to policy π, starting with a certain action a:

$$Q^{\pi}(a, b, h) \stackrel{def}{=} \rho(a, b) + \gamma \sum_{z \in Z} Pr(z \mid a, b) V^{\pi}(SE(a, z, b), h - 1)$$

$$Q^{\pi}(a, b, 1) \stackrel{def}{=} \rho(a, b).$$

Note that $V^{\pi}(b, h) = Q^{\pi}(\pi(h), b, h)$.

The forward-search approach results in relatively fast planning, compared to determining a full policy as defined in (3.4). During the calculation of the value of an action, that is, how much an action contributes to the expected future rewards, only belief-states which can be reached are considered. Put differently, only belief-states relevant in the short-term are considered for determining which action to perform next. The agents we have in mind live in changing environments. The longer it takes to generate a plan, the less likely it is that the current state is the state which the agent was in when it started planning. That is, an agent should be reactive in dynamic environments. However, there is a trade-off between reactivity and accuracy: Shorter policies result in more reactive agents, but then their reward estimates are less likely to result in the correct action selection (with respect to optimal selection). Hence, as time passes beyond time-point t when planning commenced, the policy found will be less applicable at the time-point t' when planning ended. There is thus a trade-off between accuracy and reactivity, proportional to the dynamism of the domain. This trade-off must be managed per domain.

3.2 Finite-Horizon Planning in POMDPs

We now turn to the actual topic of planning in partially observable Markov decision processes (POMDPs). Planning in the POMDP framework can be reduced to the problem of determining the 'best' actions to perform given the current belief-state.

Essentially, the forward-search approach looks ahead for h steps, calculates the total (discounted) reward that can be obtained for various possible execution sequences and then selects the next action as the first action of the sequence which provides the highest total reward. The idea is that rewards for executing sequences of actions of length h are estimates for infinitely long sequences. The larger h is, the more accurate is the estimate.

An *optimal policy* recommends the best actions. The optimal policy π^* is simply the policy which causes the agent to receive the maximum rewards over all future steps, till the horizon, starting in a given belief-state:

Definition 3.2.1: A policy π is optimal with respect to a fixed belief-state b and planning horizon h if there is no π' such that $V^{\pi'}(b, h) > V^{\pi}(b, h)$. In other words,

$$\pi^* = \arg\max_{\pi \in \Pi^h} V^{\pi}(b, h),$$

where Π^h is the set of all policies of length h.

The *optimal* value $V^*(b,h)$ of a belief-state b assumes that at each step the action that will maximize the state's value will be selected. The policy is thus implicit and needs not be provided. The optimal value of belief-state b for horizon h is

$$V^*(b,h) = \max_{a \in A} \left[\rho(a,b) + \gamma \sum_{z \in Z} Pr(z \mid a,b) V^*(SE(a,z,b), h-1) \right], \tag{3.6}$$

where $V^*(b,1) = \max_{a \in A} \rho(a,b)$. It assumes that at each step the action that will maximize the state's value will be selected.

The optimal value of belief-state b for horizon h, starting with a certain action a is

$$Q^*(a,b,h) = \rho(a,b) + \gamma \sum_{z \in Z} Pr(z \mid a,b) V^*(SE(a,z,b), h-1). \tag{3.7}$$

$V^*(b,h)$ is called the state value function and $Q^*(a,b,h)$ is called the action value function.

If an agent is not in possession of a policy, it can select its next action a^* using

$$a^* = \arg\max_{a \in A} Q^*(a,b,h) \tag{3.8}$$

where b is the current belief-state and h is the number of steps to go. The following relationship holds: $\pi^*(h) = a^*$. This relationship supports the statement that the optimal policy is implicit in the value function.

3.3 Belief-Decision-Trees

Online POMDP planning methods consist of two phases, a planning phase where a finite sequence of actions is computed, and an execution phase where the actions are executed in the real environment. After executing the actions, the agent switches back to the planning phase.

In the planning phase, Equations (3.6) and (3.7) have a graphical representation as a tree with belief-states and sensing events as nodes, and actions and observations as arcs, with the current belief-state as the root node. Finding the optimal policy of length h is analogous to finding an optimal path in the tree expanded to depth h. At each node (belief-state), certain action executions are considered, and a decision can be made about which action the agent would execute if it were in the projected belief-state. Such a tree for planning with belief-states is called a *belief-decision-tree*. Figure 3.1 depicts a belief-decision-tree of depth $h = 1$; triangles are belief-states and circles are sensing events. The actions considered in Figure 3.1 are `left` and `right`, the observations the agent can make are o_1 to o_4, leading to four different new nodes.

Figure 3.2 is a two-tier (i.e., $h = 2$) belief-decision-tree with two actions and two observations.

Planning continues until (i) the time for planning runs out, (ii) the value of the best action so far is satisfactory (ϵ-*optimal*) or (iii) the decision process has completed. If cases (i) or (ii) are not satisfied, a node from the fringe of the current tree is set as the root of a new tree which will be searched to depth h again, and so on. In case (iii), search will proceed to depth h, independent

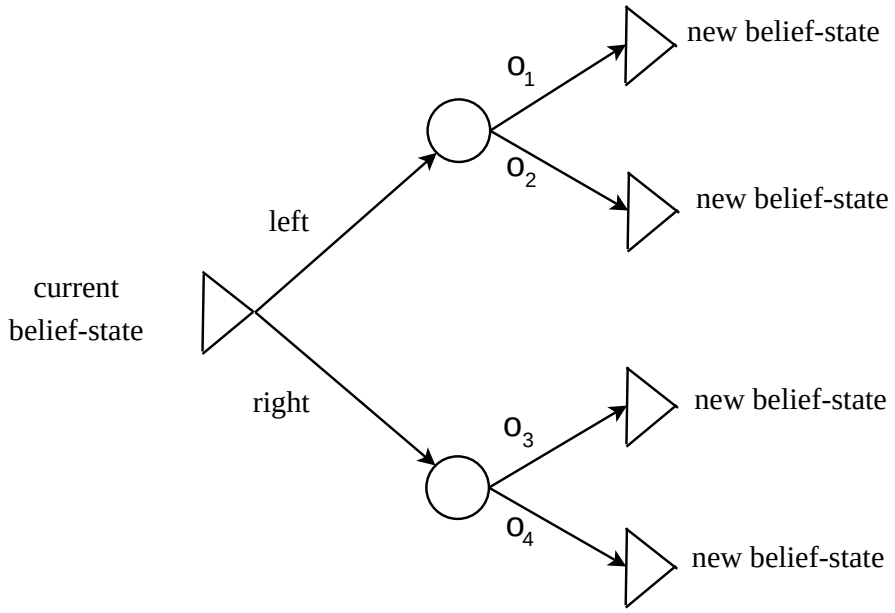


Fig. 3.1: A one-tier belief-decision-tree.

of time or whether the action considered for execution is satisfactory according to some prior knowledge.

Once the whole tree is created, node values are propagated backwards from the leaf nodes to ancestors, upto the root node, using Equation (3.6). That is, $a^* = \arg\max_{a \in A} Q^*(a, b, h')$ (Eq. 3.8) is employed at every belief-state node (represented by b), where h' is the number of steps to go.

Suppose the agent is at the current belief-state b^0 in Figure 3.2. And suppose executing `act` 1 gives more expected future reward than executing `act` 2. Now if the agent executes `act` 1, it could perceive either `obs` 1 or `obs` 2 (at the green circle). If the agent perceives `obs` 2, its belief-state will be updated to the red triangle in the first tier. Assume that `act` 2 gives more expected future reward at this stage than executing `act` 1. Then if `obs` 1 is perceived (at the blue circle), the red triangle leaf node will be reached.

An agent usually does not know which observation will be perceived at each event node. If an agent performs several actions between planning phases, it must maintain a conditional policy, conditioned on observations. For instance, suppose Figure 3.2 is the belief-decision-tree for some agent with b^0 as current belief-state. If the agent wants to do two actions before re-planning, then it must maintain the conditional policy depicted in Figure 3.3 (which is a subtree of the tree depicted in Figure 3.2). Here it is known that `act` 1 must be the first action to perform, but the second action to execute *depends* on whether `obs` 1 or `obs` 2 is perceived. The conditional policy implies that if `obs` 1 is perceived at the green circle, `act` 1 will produce the most expected reward, but if `obs` 2 is perceived, `act` 2 is the preferred action.

If an agent will execute only one action after every planning phase, it needs not maintain a (conditional) policy. It simply executes the first recommended action and then discards the policy. An agent who executes only the one action at every plan-act cycle, may not even construct an actual policy; it may only keep enough information in its working memory to determine which action to perform next.

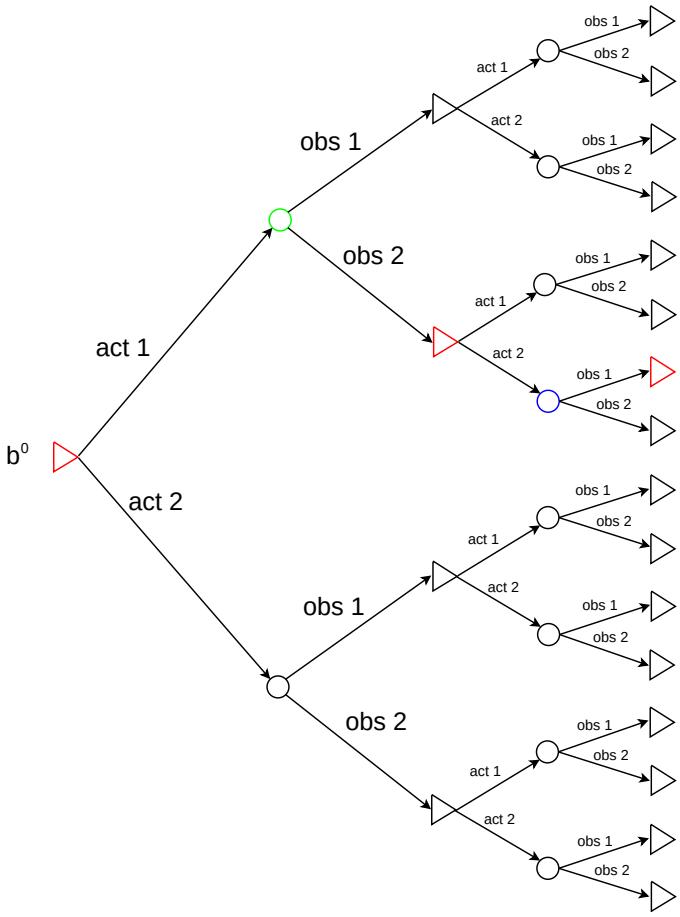


Fig. 3.2: A two-tier belief-decision-tree.

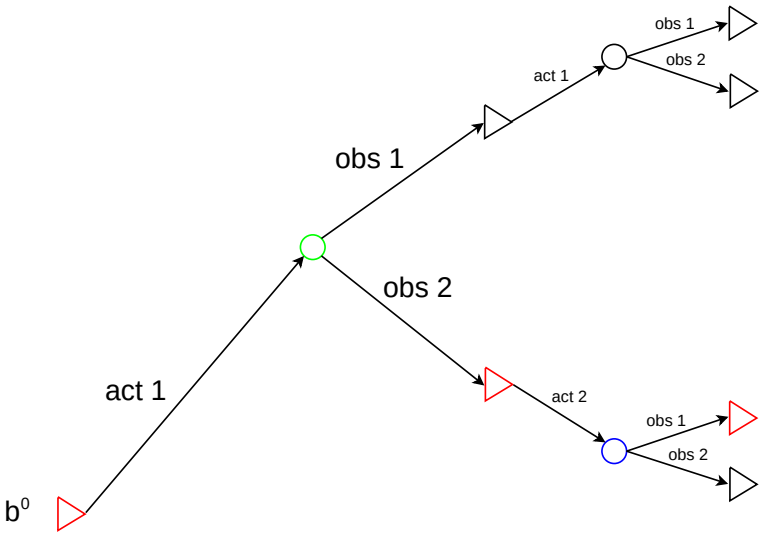


Fig. 3.3: A two-step conditional policy.

3.4 Decision-Making

We show how to make decisions with POMDPs as related to decision-making in the Stochastic Decision Logic (SDL) defined in Chapter 7. We shall assume in this section that a complete

POMDP model is given, including the initial belief-state b^0. A decision is made in the belief-state b' reached after a sequence of actions and observations. We consider decisions conditioned on two kinds of statements, (i) the utility or expected reward of a sequence of actions starting in b' and (ii) the probability of being in a particular set of states in b'.

Suppose a sequence of actions and observations is represented as

$$[\![a^1 + z^1]\!][\![a^2 + z^2]\!] \cdots [\![a^y + z^y]\!].$$

b^y for such a sequence, is determined by finding $b^1 = SE(a^1, z^1, b^0)$, then $b^2 = SE(a^2, z^2, b^1)$, then ..., then $b^y = SE(a^y, z^y, b^{y-1})$. That is,

$$b^y = SE(a^y, z^y, \cdots SE(a^1, z^1, b^0) \cdots).$$

Consider the first kind of statement mentioned at the beginning of this section. Suppose an agent wants to decide which action to execute next, and it is willing or able to reason, by projection, h steps into the future. Let the h steps be represented as $[\![a^1]\!][\![a^2]\!] \cdots [\![a^h]\!]$. The utility of performing each action in the sequence in succession is defined by

$$U([\![a^1]\!][\![a^2]\!] \cdots [\![a^h]\!], b) \stackrel{def}{=} \rho(a^1, b) + \gamma \sum_{z \in Z} Pr(z \mid a^1, b) U([\![a^2]\!] \cdots [\![a^h]\!], SE(a^1, z, b)),$$

where $U([\![a^h]\!], b) \stackrel{def}{=} \rho(a^h, b')$. Suppose Ξ^h is every possible sequence of actions of length h. Then

$$\max_{[\![a^1]\!][\![a^2]\!] \cdots [\![a^h]\!] \in \Xi^h} U([\![a^1]\!][\![a^2]\!] \cdots [\![a^h]\!], b) = V^*(b, h).$$

A straightforward and practical way to use $U(\cdot)$ is to decide which of several possible sequences of actions to execute in b. Suppose the agent is considering executing either $[\![a_1]\!][\![a_1]\!]$, $[\![a_1]\!][\![a_2]\!]$, $[\![a_2]\!][\![a_1]\!]$ or $[\![a_2]\!][\![a_2]\!]$. Then it would determine $U([\![a_1]\!][\![a_1]\!], b') = r_{11}$, $U([\![a_1]\!][\![a_2]\!], b') = r_{12}$, $U([\![a_2]\!][\![a_1]\!], b') = r_{21}$ and $U([\![a_2]\!][\![a_2]\!], b') = r_{22}$, and decide to execute the sequence associated with $max\{r_{11}, r_{12}, r_{21}, r_{22}\}$.

Let $\varphi \subseteq S$ be a set of states. Making decisions based on the second kind of statement mentioned at the beginning of this section is unusual in POMDP theory. Nevertheless, it is defined here to make the link with SDL, which provides for expressing such statements quite naturally. The probability with which the system/agent is in one of the states in φ is defined by

$$B(\varphi, b) \stackrel{def}{=} \sum_{s \in \varphi} b(s).$$

One example of how an agent could use such information follows. Suppose the agent knows (or has calculated) that a sequence Ξ of actions will land it in one of the belief-states in set $\mathcal{B}^{\Xi}$, and the agent never wants to be in a state in φ with a probability greater than p. If for any one of $b' \in \mathcal{B}^{\Xi}$, $B(\varphi, b') > p$, then the agent should not execute Ξ.

Examples

Let $F = \{f_1, f_2, \ldots, f_m\}$ be a set of Boolean features of interest and $S = \{s_1, s_2, \ldots, s_n\}$ the $2^{|F|}$ set of states induced from F.[3] For instance, if $F = \{f_1, f_2\}$, then $S = \{\{(f_1, 0), (f_2, 0)\}, \{(f_1, 1), (f_2, 0)\}, \{(f_1, 0), (f_2, 1)\}, \{(f_1, 1), (f_2, 1)\}\}$, where, in general, a pair $(f, 1)$ is in a state s if f is a feature of the domain/system when the system is in the state represented by s, else a pair $(f, 0)$ is in s. Let $b' = \{(s_1, p_1), (s_2, p_2), \ldots, (s_n, p_n)\}$ be some belief-state.

To better motivate parts of our research and to provide a setting on which to base examples for illustrating concepts, we present the *oil-drinking scenario*. The oil-drinking scenario will be used throughout this thesis, augmented or modified as appropriate. Imagine a robot that is in need of an oil refill. The robot has a single arm with a gripper at the end. There is an open can of oil on the floor within reach of its gripper. If there is nothing else in the robot's gripper, it can grab the can (or miss it, or knock it over) and it can drink the oil by lifting the can to its 'mouth' and pouring the contents in (or miss its mouth and spill). The robot may also want to confirm whether there is anything left in the oil-can by weighing its contents with its arm—there is a weight-sensor built into the arm. And once holding the can, the robot may wish to replace it on the floor.

The robot thus has the set of (intended) actions {`grab`, `drink`, `weigh`, `replace`} with the expected intuitive meanings. The robot can perceive observations only from the set {`obsNil`, `obsHeavy`, `obsMedium`, `obsLight`}. Intuitively, when the robot performs a `weigh` action, it will perceive either `obsHeavy`, `obsMedium` or `obsLight`; for other actions, it will 'perceive' `obsNil`, no perception occurs, or a null observation is perceived. The robot experiences its world (domain) via three Boolean features: {`full`, `drank`, `holding`} meaning respectively that the the oil-can is full, that the robot has drunk the oil and that it is currently holding something in its gripper.

Let f stand for `full` and let h stand for `holding`. Let g stand for `grab`, d for `drink` and w for `weigh`. Let N stand for `obsNil`, L for `obsLight`, M for `obsMedium` and H for `obsHeavy`}. We define the following POMDP model.

- $S = \{\{(f, 0), (h, 0)\}, \{(f, 1), (h, 0)\}, \{(f, 0), (h, 1)\}, \{(f, 1), (h, 1)\}\}$;
- $A = \{g, d, w\}$;
- Zero probability transitions are not shown.

$$T(\{(f, 0), (h, 0)\}, g, \{(f, 0), (h, 0)\}) = 0.1$$
$$T(\{(f, 0), (h, 0)\}, g, \{(f, 0), (h, 1)\}) = 0.1$$
$$T(\{(f, 0), (h, 0)\}, g, \{(f, 1), (h, 1)\}) = 0.8$$
$$T(\{(f, 1), (h, 0)\}, g, \{(f, 0), (h, 0)\}) = 0.1$$
$$T(\{(f, 1), (h, 0)\}, g, \{(f, 0), (h, 1)\}) = 0.1$$
$$T(\{(f, 1), (h, 0)\}, g, \{(f, 1), (h, 1)\}) = 0.8$$
$$T(\{(f, 0), (h, 1)\}, d, \{(f, 0), (h, 0)\}) = 0.05$$

[3] Any state which can be described by a set of multi-valued features can be uniquely described by a sufficiently large set of Boolean features.

$T(\{(f,0),(h,1)\},d,\{(f,0),(h,1)\}) = 0.95$

$T(\{(f,1),(h,1)\},d,\{(f,0),(h,0)\}) = 0.05$

$T(\{(f,1),(h,1)\},d,\{(f,0),(h,1)\}) = 0.95$

$T(\{(f,0),(h,0)\},w,\{(f,0),(h,0)\}) = 1$

$T(\{(f,1),(h,0)\},w,\{(f,1),(h,0)\}) = 1$

$T(\{(f,0),(h,1)\},w,\{(f,0),(h,1)\}) = 1$

$T(\{(f,1),(h,1)\},w,\{(f,1),(h,1)\}) = 1$

- $R(g,\{(f,0),(h,0)\}) = -6$

  $R(g,\{(f,1),(h,0)\}) = -1$

  $R(g,\{(f,0),(h,1)\}) = 9$

  $R(g,\{(f,1),(h,1)\}) = -1$

  $R(d,\{(f,0),(h,0)\}) = -6$

  $R(d,\{(f,1),(h,0)\}) = -1$

  $R(d,\{(f,0),(h,1)\}) = 9$

  $R(d,\{(f,1),(h,1)\}) = -1$

  $R(w,\{(f,0),(h,0)\}) = -5.8$

  $R(w,\{(f,1),(h,0)\}) = -2$

  $R(w,\{(f,0),(h,1)\}) = 9.2$

  $R(w,\{(f,1),(h,1)\}) = -2$

- $Z = \{N, L, M, H\}$
- Zero probability observations are not shown.

  $O(\{(f,0),(h,0)\},g,N) = 1$

  $O(\{(f,0),(h,0)\},d,N) = 1$

  $O(\{(f,0),(h,0)\},w,L) = \frac{1}{3}$

  $O(\{(f,0),(h,0)\},w,M) = \frac{1}{3}$

  $O(\{(f,0),(h,0)\},w,H) = \frac{1}{3}$

  $O(\{(f,1),(h,0)\},g,N) = 1$

  $O(\{(f,1),(h,0)\},d,N) = 1$

  $O(\{(f,1),(h,0)\},w,L) = \frac{1}{3}$

  $O(\{(f,1),(h,0)\},w,M) = \frac{1}{3}$

  $O(\{(f,1),(h,0)\},w,H) = \frac{1}{3}$

$$O(\{(f,0),(h,1)\},g,N) = 1$$
$$O(\{(f,0),(h,1)\},d,N) = 1$$
$$O(\{(f,0),(h,1)\},w,L) = 0.5$$
$$O(\{(f,0),(h,1)\},w,M) = 0.3$$
$$O(\{(f,0),(h,1)\},w,H) = 0.2$$
$$O(\{(f,1),(h,1)\},w,L) = 0.1$$
$$O(\{(f,1),(h,1)\},w,M) = 0.2$$
$$O(\{(f,1),(h,1)\},w,H) = 0.7$$

- $b^0 = \{(\{(f,1),(h,1)\},0.35),(\{(f,1),(h,0)\},0.35),(\{(f,0),(h,1)\},0.2),$
  $(\{(f,0),(h,0)\},0.1)\}.$

It could be that the robot is programmed to grab the oil-can and then weigh it, only if it believes with a probability greater than 0.85 that it is holding the can after the two actions are performed. So the robot must determine whether

$$B(\{\{(f,1),(h,1)\},\{(f,0),(h,1)\}\},b^2) > 0.85,$$

where $b^2 = SE(w,M,b^1)$ and $b^1 = SE(g,N,b_0)$. In other words, it must determine whether the probability of being in a state where feature `holding` is true is greater than 0.85 after executing the sequence $[\![g+N]\!][\![w+M]\!]$ in the initial belief-state. Using Equation (3.1), one can calculate that

$$b^1 = \{(\{(f,1),(h,1)\},0.\overline{81}),(\{(f,1),(h,0)\},0),(\{(f,0),(h,1)\},0.\overline{09}),(\{(f,0),(h,0)\},0.\overline{09})\}$$

and

$$b^2 = \{(\{(f,1),(h,1)\},0.72973),(\{(f,1),(h,0)\},0),(\{(f,0),(h,1)\},0.12162),$$
$$(\{(f,0),(h,0)\},0.14865)\}.$$

Finally, we see that

$$0.72973 + 0.12162 > 0.85$$

There is a subtlety in the interpretation of this result: It is not the case that the probability of grabbing the can and then weighing it and perceiving that it is of medium weight and then still holding the can is greater than 0.85. It is the case that IF the robot grabs the oil-can (and perceives nothing/`Nil`) and IF it weighs the can and IF it is perceived to be medium, THEN the probability of still holding the can is greater than 0.85.

Second, we determine whether $U([\![d]\!][\![d]\!],b^1) \leq 7$, where $b^1 = SE(g,N,b_0)$. In other words, we want to determine whether the utility of drinking twice in a row is less than or equal to 7 units after grabbing the oil-can and perceiving nothing (`Nil`) in the initial belief-state. For simplicity,

assume $\gamma = 1$.

$$U([\![d]\!][\![d]\!], b^1) = \rho(d, b^1) + \sum_{z \in Z} Pr(z \mid d, b^1) U([\![d]\!], SE(d, z, b^1)). \tag{3.9}$$

Notice that $Pr(L \mid d, b^1) = Pr(M \mid d, b^1) = Pr(H \mid d, b^1) = 0$, because $O(s, d, L) = O(s, d, M) = O(s, d, H) = 0$ for all states s. So (3.9) becomes

$$\begin{aligned}
U([\![d]\!][\![d]\!], b^1) &= \rho(d, b^1) + Pr(N \mid d, b^1) U([\![d]\!], SE(d, z, b^1)) \\
&= \rho(d, b^1) + Pr(N \mid d, b^1) U([\![d]\!], b^{2'}) \\
&= \rho(d, b^1) + Pr(N \mid d, b^1) \rho(d, b^{2'}) \\
&= 0.\overline{81}(-1) + 0(-1) + 0.\overline{09}(9) + 0.\overline{09}(-6) + \\
&\qquad 0.\overline{90}(0(-1) + 0(-1) + 0.95(9) + 0.05(-6)) \\
&= 6.95455
\end{aligned}$$

where $Pr(N \mid d, b^1) = 0.\overline{90}$ and

$$b^{2'} = \{(\{(f, 1), (h, 1)\}, 0), (\{(f, 1), (h, 0)\}, 0), (\{(f, 0), (h, 1)\}, 0.95), (\{(f, 0), (h, 0)\}, 0.05)\}.$$

Suppose the robot is programmed to never drink twice in a row if the utility of doing so is less than 7 units. Then the robot knows that if it grabs the oil-can (and certainly perceives `Nil`), it will be in a belief-state where it should not drink twice in a row.

3.5 Concluding Remarks

Arguably, the POMDP-approach's popularity is due to the relative simplicity and intuitiveness of its model and its general applicability to a wide range of stochastic domains. Only a very small part of POMDP theory was covered in this chapter; only the part on which the SDL is based. In particular, infinite-horizon POMDPs were not discussed, algorithms for finding optimal policies, optimization techniques, the many approximate POMDP algorithms and their computational complexity were not discussed.

For further reading, refer to Cassandra et al. [1994], Kaelbling et al. [1998], Boutilier et al. [1999] and Russell and Norvig [2003]. For the foundations of POMDP theory, refer to the following literature: [Aström, 1965, Smallwood and Sondik, 1973, Monahan, 1982, Lovejoy, 1991].

4. THE LOGIC OF ACTIONS AND OBSERVATIONS

An earlier version of the logic presented in this chapter was presented at the twenty-third Australasian Joint Conference on Artificial Intelligence in Adelaide, Australia [Rens et al., 2010].

It is the norm in dynamic logics (and some other agent oriented logics) to deal with observations as elements of knowledge, as propositions; and perception is normally coded as action. That is, observations-as-propositions evaluate to 'true' or 'false' depending on some action(s). However, the approach of interpreting observations as mere propositions may be counterintuitive to some people, because knowledge may be seen as something different from events (observations) that *generate* or *modify* knowledge.

Remark: If an intelligent agent is regarded as a *system*, then there are inputs to the system that affect it, and outputs from the system that affect the environment. The inputs are *observations* and the outputs are *actions*. If one assumes that the system state is represented by a knowledge-base of *propositions*, then from the *systems* view, it is clear that observations and propositions are different in nature.

From a philosophical perspective, an *observation* is a concept in the 'mind' of an agent—observations are just symbols that an agent uses to reason about its input signals. The use of sensors is an integral and indispensable component of the robotic system. Sensors are employed to bring information inherent in the environment into the robot's 'mind'. A sensor transfers energy in the environment into numeric/symbolic values that may be interpreted and stored as propositions in the robot's knowledge-base [Levesque and Lakemeyer, 2008]. When an agent performs an action, a certain kind of sensory input is possible, with a certain range. This work takes the stance that all actions have an ontic (physical) component *and* an epistemic (knowledge) component. The ontic component has an effect on the outside world, and the epistemic component receives signals from / is affected by the world. The signal impinges on the agent's senses and the agent interprets or 'perceives' the signal as an observation, in the context of the immediately preceding action.

For example, if action `open−eyes` is performed, several signals are possible, depending on the situation, like `wall−3−meters−ahead` or `overcast−sky`. If the agent performs an action like `step−once−forward`, there is only one observation possible, viz. `null`, the 'dummy' observation. Unlike the eye, the leg or wheel is not a sensory organ. When our agent activates a device (the agent acts) and the device receives no input signal, it interprets this state of affairs as the `null` observation.[1] Hence, we take the philosophical stance that for every action an agent performs, the agent perceives exactly one observation, that is, actions and observations always appear in pairs, even if implicitly. This is the approach of POMDPs that we rely on in the present work [Pineau, 2004, e.g.].

[1] The `null` observation will be denoted by the special named constant, `obsNil`.

Therefore, the ability to distinguish between observations and propositions allows for a more precise specification of a given domain, as we shall see later in this chapter. It turns out that the notion of observations as explicit syntactic and semantic objects of a logic is not completely new. For instance, Van Benthem et al. [2009] do so (cf. their logic PDEL in Section 8.9). They allude that their "events" are closer to observations than logical propositions. Observations are *mappings* in the approach of Geffner and Wainer [1998]. However, it will be seen that LAO's observations are much simpler than their mappings.

The domain of the oil-drinking scenario is (partially) formalized as follows. The robot has the set of actions $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{weigh}, \texttt{replace}\}$, can make observations from the set $\Omega = \{\texttt{obsNil}, \texttt{obsHeavy}, \texttt{obsMedium}, \texttt{obsLight}\}$ and the robot experiences its world via three Boolean features: $\mathcal{F} = \{\texttt{full}, \texttt{drank}, \texttt{holding}\}$. This formalization seems more intuitive than lumping all observations in with propositions, for instance, by making $\mathcal{F} = \{\texttt{full}, \texttt{drank}, \texttt{holding}, \texttt{obsnil}, \texttt{heavy}, \texttt{medium}, \texttt{light}\}$.

Given a formalization $\mathcal{K}$ of our scenario, the robot may have the following queries:

- Is it possible that after grabbing the oil-can, I will not be holding it?
  That is, does $\langle\texttt{grab}\rangle\neg\texttt{holding}$ follow from $\mathcal{K}$?
- If I weigh the oil-can and perceive that it is light, is it necessary that I have drunk the oil?
  That is, does $[\texttt{obsLight} \mid \texttt{weigh}]\ \texttt{drank}$ follow from $\mathcal{K}$?

The logic we present here allows for expressing observations explicitly, distinct from propositions. It is called the Logic of Actions and Observations (LAO). LAO is a modal logic with quantification and equality over the actions and observations. It will be able to accommodate formal descriptions of nondeterministic uncertainty in the actions and in the observations.

LAO is based on $\mathcal{LAP}$ (the Logic of Actions and Plans [Castilho et al., 1999]), but with one major difference: the addition of *observations*. That is, LAO refers to a set of observations that are explicitly identified by a knowledge engineer or agent-system designer (cf. the Remark above). A minor, yet important difference is the addition of action and observation variables, quantification and equality. Another important difference is that the notion of global semantic consequence in the semantics replaces their $\Box$ operator.

In a discrete, non-probabilistic setting such as LAO, uncertainty about sensor signals can be captured by a knowledge engineer by specifying that more than one observation is possible, given some (sensing) action. Whether one or more observations is possible, the knowledge engineer may know that when a certain action is performed under certain conditions, the correct observation cannot be made or has never been made by agents in the past, in similar conditions. The knowledge engineer will thus model the correct observation as impossible given the action is performed in the particular situation. Discrete logics are limited in how they can model sensory noise. Probabilistic logics should be able to model noise in perception more naturally, to an arbitrary accuracy, given the available information about the two sources of noise associated with perception. Dealing with noisy sensing is the topic of later chapters.

Although there are several formalisms in the literature on reasoning about and specifying agents and their actions, we found them lacking when it comes to treating observations as objects on a par with actions, while retaining important computational properties. Existing first-order based

approaches are in general undecidable or arguably have too complicated semantics. For these reasons, we prefer to anchor our framework on a version of dynamic logic and strive for an extension of it by allowing for observations as explicit entities.

By adding explicit observation constants to a simple dynamic logic, the resulting logic may be slightly more complex, while perhaps simplifying, for the domain expert, dealing with observation.

We also provide a framework to formally specify, in the language of LAO, the domain in which an agent or robot is expected to live.

Section 4.1 defines the syntax and semantics of LAO. We present the decision procedure for semantic consequence in Section 4.2. Soundness, completeness and termination of the decision procedure are proved in Sections 4.3.1, 4.3.2 and 4.3.3. Sections 4.4.1 explains how to specify action effects. Section 4.4.2 discusses the executability of actions and the perceivability of observations. Section 4.5 presents a domain specification of the oil-drinking scenario, and the determination of several entailment examples in that domain is shown in Section 4.6.

4.1 Defining the Logic

First, the syntax of the logic is presented, then its semantics.

4.1.1 Syntax

The vocabulary of our language contains five sorts of objects of interest:

1. a finite set of *fluents* (alias, *propositional atoms*) $\mathcal{F} = \{f_1, \ldots, f_n\}$,
2. a finite set of names of atomic *actions* $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$,
3. a countable set of *action variables* $V_{\mathcal{A}} = \{v_1^a, v_2^a, \ldots\}$,
4. a finite set of names of atomic *observations* $\Omega = \{\varsigma_1, \ldots, \varsigma_n\}$,
5. a countable set of *observation variables* $V_\Omega = \{v_1^o, v_2^o, \ldots\}$.

We shall refer to elements of $\mathcal{A} \cup \Omega$ as *constants* and elements of $V_{\mathcal{A}} \cup V_\Omega$ as *variables*. A *literal* ℓ is a fluent or its negation.

We are going to work in a multi-modal setting, in which we have a modal operator $[\varsigma|\alpha]$, one for each pair (α, ς) in $\mathcal{A} \times \Omega$.

Definition 4.1.1: Let $\alpha \in (\mathcal{A} \cup V_{\mathcal{A}})$, $\varsigma \in (\Omega \cup V_\Omega)$, $v \in (V_{\mathcal{A}} \cup V_\Omega)$ and $f \in \mathcal{F}$. The language of LAO, denoted $\mathcal{L}_{LAO}$, is the least set of those Φ that contain no free variables:

$$\Phi ::= f \mid \top \mid \neg\Phi \mid \Phi \wedge \Phi \mid \alpha = \alpha \mid \varsigma = \varsigma \mid [\varsigma \mid \alpha]\Phi \mid (\forall v)\Phi.$$

The scope of quantifier $(\forall v')$ is determined in the same way as is done in first-order logic. A variable v appearing in a formula Ψ is said to be *bound* by quantifier $(\forall v')$ if and only if v is the same variable as v' and is in the scope of $(\forall v')$. If a variable is not bound by any quantifier, it is

free. In $\mathcal{L}_{LAO}$, variables are not allowed to be free; they are always bound. For example, $[v^o|\alpha]$ is not in $\mathcal{L}_{LAO}$, but $(\forall v^o)[v^o|\alpha]$ is.

$\bot$ is an abbreviation for $\neg\top$, $\Phi \vee \Phi'$ is an abbreviation for $\neg(\neg\Phi \wedge \neg\Phi')$, $\Phi \rightarrow \Phi'$ is an abbreviation for $\neg\Phi \vee \Phi'$, $\Phi \leftrightarrow \Phi'$ is an abbreviation for $(\Phi \rightarrow \Phi') \wedge (\Phi' \rightarrow \Phi)$, $c \neq c'$ is an abbreviation for $\neg(c = c')$ and $(\exists v)$ is an abbreviation for $\neg(\forall v)\neg$. $\rightarrow$ and $\leftrightarrow$ have the weakest bindings and $\neg$ the strongest; parentheses enforce or clarify the scope of operators conventionally.

The sentence $[\varsigma \mid \alpha]\Phi$ is read 'Φ must hold after ς is observed, given α is executed'. For instance, $[\texttt{obsLight} \mid \texttt{weigh}]\neg\texttt{full}$ means 'After perceiving that the oil-can is light, given a weighing action, the can is necessarily not full'. $[\alpha]\Phi$ is an abbreviation for $(\forall v^o)[v^o \mid \alpha]\Phi$ and is read 'Φ must hold (after any/every observation) given α is executed'. For instance, $[\texttt{replace}]\neg\texttt{holding}$ means 'After replacing the oil-can, it is definitely not being held (regardless of observations)'.

$\langle\alpha\rangle\Phi$ and $\langle\varsigma \mid \alpha\rangle\Phi$ abbreviate $\neg[\alpha]\neg\Phi$ and $\neg[\varsigma \mid \alpha]\neg\Phi$ respectively. One conventional reading for $\langle\alpha\rangle\Phi$ is 'It is possible that Φ holds after α is performed'. The reading of $\langle\varsigma \mid \alpha\rangle\Phi$ is 'It is possible that Φ holds after ς is perceived, given α is performed'.

4.1.2 Semantics

Our semantics follows that of multi-modal logic **K** [Popkorn, 1994] with possible worlds [Kripke, 1959, Hintikka, 1962], however, structures do not have the standard Kripke-style semantics (see, e.g., [Chellas, 1980, Popkorn, 1994, Hughes and Cresswell, 1996]). Standard Kripke modal logic structures (alias, possible worlds models) are tuples $\langle W, R, V\rangle$, where W is a (possibly infinite) set of states (possibly without internal structure), R is a binary relation on W, and V is a valuation, assigning subsets of W to each atomic proposition.

Intuitively, when talking about some world w, we mean a set of features (formally, *fluents*) that the agent understands and that describes a state (of affairs) in the world or that describes a possible, alternative world. Let $w : \mathcal{F} \mapsto \{0, 1\}$ be a total function that assigns a truth value to each fluent. Let C be the set of all possible functions w. We call C the *conceivable worlds*.

Definition 4.1.2: A LAO structure is a tuple $\mathcal{S} = \langle W, R, O, N, Q\rangle$ such that

1. $W \subseteq C$ is the nonempty finite set of *possible worlds*;
2. R is a mapping that provides an accessibility relation $R_\alpha \subseteq W \times W$ for each action $\alpha \in \mathcal{A}$;
3. O is a nonempty finite set of observations;
4. $N : \Omega \mapsto O$ is a total bijection that associates to each name in Ω, a unique observation in O;
5. Q is a mapping that provides a perceivability relation $Q_\alpha \subseteq O \times W$ for each action $\alpha \in \mathcal{A}$;

R_α defines which worlds w^+ are accessible via action α performed in world w^- and Q_α defines which observations o are perceivable in worlds w^+ accessible via action α. For $\varsigma \in \Omega$, $N(\varsigma) = o \in O$. Because N is a total bijection, it follows that $|O| = |\Omega|$. Next we define satisfaction of formulae in $\mathcal{L}_{LAO}$.

Let $\mathcal{S}$ be a LAO structure, with $\alpha, \alpha' \in \mathcal{A}$, $v^a \in V_{\mathcal{A}}$, $\varsigma, \varsigma' \in \Omega$, $v^o \in V_\Omega$ and $f \in \mathcal{F}$. And let Φ be any sentence in $\mathcal{L}_{LAO}$.

Definition 4.1.3 (Truth Conditions): We say Φ is *satisfied* at world w in structure $\mathcal{S}$ (written $\mathcal{S}, w \models \Phi$) if and only if the following hold:

1. $\mathcal{S}, w \models f \iff w(f) = 1$;
2. $\mathcal{S}, w \models \top$ for any/every $w \in W$;
3. $\mathcal{S}, w \models \alpha = \alpha' \iff \alpha, \alpha' \in \mathcal{A}$ are the same element;
4. $\mathcal{S}, w \models \varsigma = \varsigma' \iff \varsigma, \varsigma' \in \Omega$ are the same element;
5. $\mathcal{S}, w \models [\varsigma \mid \alpha]\Phi \iff$ for all w', if $(w, w') \in R_\alpha$ and $(N(\varsigma), w') \in Q_\alpha$, then $\mathcal{S}, w' \models \Phi$;
6. $\mathcal{S}, w \models \neg\Phi \iff \mathcal{S}, w \not\models \Phi$;
7. $\mathcal{S}, w \models \Phi \wedge \Phi' \iff \mathcal{S}, w \models \Phi$ and $\mathcal{S}, w \models \Phi'$;
8. $\mathcal{S}, w \models (\forall v^a)\Phi \iff \mathcal{S}, w \models \Phi|^{v^a}_{\alpha}$ for all $\alpha \in \mathcal{A}$;
9. $\mathcal{S}, w \models (\forall v^o)\varphi \iff \mathcal{S}, w \models \Phi|^{v^o}_{\varsigma}$ for all $\varsigma \in \Omega$,

where we write $\Phi|^v_c$ to mean the formula Φ with all variables $v \in (V_{\mathcal{A}} \cup V_\Omega)$ appearing in it replaced by constant $c \in \mathcal{A} \cup \Omega$ of the right sort.

A formula Φ is valid in a LAO structure (denoted $\mathcal{S} \models \Phi$) if $\mathcal{S}, w \models \Phi$ for every $w \in W$. Φ is LAO-valid (denoted $\models \Phi$) if Φ is valid in every structure $\mathcal{S}$. Φ is *satisfiable* if $\mathcal{S}, w \models \Phi$ for some $\mathcal{S}$ and $w \in W$.

Let $\mathcal{K} \subseteq \mathcal{L}_{LAO}$ and $\Phi \in \mathcal{L}_{LAO}$. We define *global semantic consequence* (denoted $\mathcal{K} \models_G \Phi$) as follows:

$$\text{for all } \mathcal{S}, \text{for all } \kappa \in \mathcal{K}, \text{ if } \mathcal{S} \models \kappa, \text{ then } \mathcal{S} \models \Phi.$$

Due to the nature of the 'observation naming' function N, in the rest of this chapter, to simplify notation, we let $O = \Omega$ (such that $N(\varsigma) = \varsigma$).

The motivation behind the definition of $\mathcal{S}, w \models [\varsigma \mid \alpha]\Phi$ is as follows. Just as worlds w' are not considered if $(w, w') \notin R_\alpha$, worlds w' are not considered if $(\varsigma, w') \notin Q_\alpha$. In other words, whether or not a world w' is reachable (via R_α), if the agent perceived ς and the agent knows that ς is not perceivable in w', then the agent knows it is not in w'. Then what is true and false in w' has no influence on the satisfaction of $\mathcal{S}, w \models [\varsigma \mid \alpha]\Phi$. But in every world w' reachable from w *and* in which ς is perceivable, Φ must be true. In yet other words, all worlds w' for which $(w, w') \in R_\alpha$ and $(\varsigma, w') \in Q_\alpha$ hold, are possible successor worlds, and only those worlds are possible, and Φ must be true in all those worlds. While actions can add worlds that an agent believes possible, thus increasing uncertainty, observations eliminate reachable worlds from consideration, thus increasing certainty.

In the proof of the proposition below, we use infix notation, that is, we write xAy instead of $(x, y) \in A$, where A is a binary relation. Infix notation may be used from time to time in this thesis.

Proposition 4.1.1: $[\alpha]$ has the regular multi-modal logic interpretation. That is, if $\mathcal{S}, w \models [\alpha]\Phi$, then For all w', if $wR_\alpha w'$, then $\mathcal{S}, w' \models \Phi$.

Proof:
Let $\mathcal{S}$ be an arbitrary LAO structure and let w be an arbitrary world in $\mathcal{S}$.

$$\begin{aligned}
&\mathcal{S}, w \models [\alpha]\Phi \\
&\Rightarrow \mathcal{S}, w \models (\forall v^o)[v^o \mid \alpha]\Phi \\
&\Rightarrow \text{ For all } o \in O, \text{ for all } w', \text{ if } wR_\alpha w' \text{ and } oQ_\alpha w', \text{ then } \mathcal{S}, w' \models \Phi
\end{aligned}$$

$\Rightarrow$ There are no o, w' s.t. it is not the case that if $wR_\alpha w'$ and $oQ_\alpha w'$, then $\mathcal{S}, w' \models \Phi$

$\Rightarrow$ Not: there exist w' and o s.t. if $wR_\alpha w'$ and $oQ_\alpha w'$, then $\mathcal{S}, w' \not\models \Phi$

$\Rightarrow$ Not: there exists w' s.t. if $wR_\alpha w'$, then $\mathcal{S}, w' \not\models \Phi$

$\Rightarrow$ For all w', if $wR_\alpha w'$, then $\mathcal{S}, w' \models \Phi$.

■

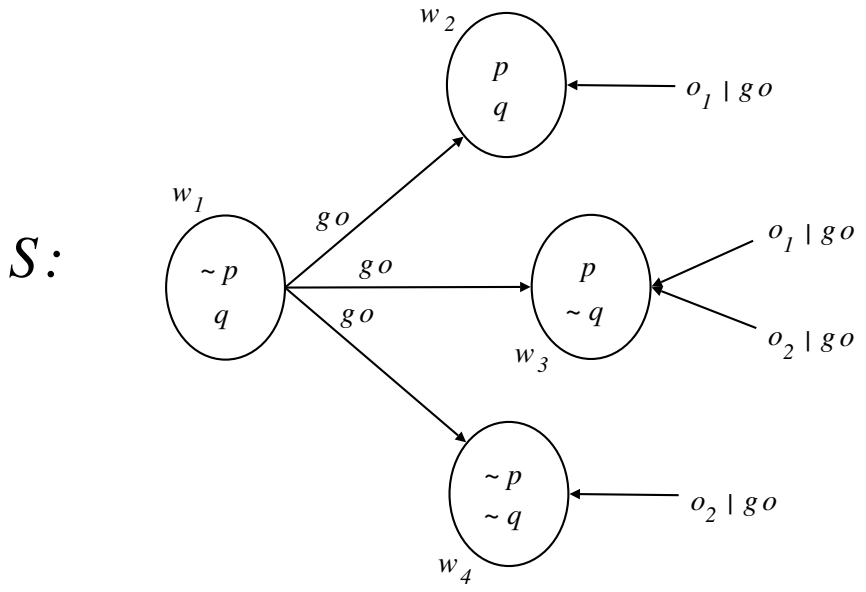


Fig. 4.1: A structure to illustrate the semantics of some basic sentences in LAO. An arrow that does not emanate from a world but has the symbols "$\varsigma \mid \alpha$" at the arrow's tail, represent the fact that ς is perceivable in the world at which the arrow terminates, give action α. $\sim$ denotes $\neg$.

Consider the example based on Figure 4.1. Observe that

- $\mathcal{S}, w_1 \not\models [go]p$.
- $\mathcal{S}, w_1 \models [o_1 \mid go]p$.
- $\mathcal{S}, w_1 \not\models [o_1 \mid go]q$.
- $\mathcal{S}, w_1 \not\models [o_1 \mid go]\neg q$.
- $\mathcal{S}, w_1 \models \langle go\rangle(\neg p \wedge \neg q)$.

- $\mathcal{S}, w_1 \not\models \langle o_1 \mid go \rangle(\neg p \wedge \neg q)$.
- $\mathcal{S}, w_1 \models \langle o_2 \mid go \rangle(\neg p \wedge \neg q)$.

Propositions 4.1.2 and 4.1.3 state that a formula of the form $(\forall v)\varphi$ can be interpreted as a finite conjunction of φ where in each conjunct, variable v is replaced by the appropriate constant (action or observation).

Proposition 4.1.2: Let $v^a \in V_{\mathcal{A}}$ and $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$.
$\mathcal{S}, w \models (\forall v^a)\varphi$ iff $\mathcal{S}, w \models \varphi|^{v^a}_{\alpha_1} \wedge \ldots \wedge \varphi|^{v^a}_{\alpha_n}$.

Proof:
$\mathcal{S}, w \models (\forall v^a)\varphi$
$\iff \mathcal{S}, w \models \varphi|^{v^a}_{a}$ for all $\alpha \in \mathcal{A}$
$\iff \mathcal{S}, w \models \varphi|^{v^a}_{\alpha_1}$ and $\ldots$ and $\mathcal{S}, w \models \varphi|^{v^a}_{\alpha_n}$
$\iff \mathcal{S}, w \models \varphi|^{v^a}_{\alpha_1} \wedge \ldots \wedge \varphi|^{v^a}_{\alpha_n}$. ■

Proposition 4.1.3: Let $v^o \in V_{\Omega}$ and $\Omega = \{\varsigma_1, \ldots, \varsigma_n\}$.
$\mathcal{S}, w \models (\forall v^o)\varphi$ iff $\mathcal{S}, w \models \varphi|^{v^o}_{\varsigma_1} \wedge \ldots \wedge \varphi|^{v^o}_{\varsigma_n}$.

Proof:
Symmetrical to the case for actions. ■

4.2 Decision Procedure for Semantic Consequence

The decision procedure for global semantic consequence in LAO employs a tableau method and then post processing of a saturated tableau tree to determine whether the tree is open or closed, as defined below. The procedure determines whether $\mathcal{K} \models_G \Psi$, where $\mathcal{K}$ is any set of global axioms in $\mathcal{L}_{LAO}$ and Ψ is any (local) sentence in $\mathcal{L}_{LAO}$.

Because the vocabulary of LAO is finite, it can be assumed that $\mathcal{K}$ is finite (assuming $\mathcal{K}$ contains the domain axioms). We also assume Ψ is of finite length.

The necessary definitions and terminology are given next.

Definition 4.2.1: A *labeled formula* is a pair (x, Ψ), where $\Psi \in \mathcal{L}_{LAO}$ is a formula and x is an integer called the *label* of Ψ.

Definition 4.2.2: A *skeleton* Σ is a ternary relation $\Sigma \subseteq (\Omega \cup \mathbb{N}) \times \mathcal{A} \times \mathbb{N}$. Elements $(\cdot, a, n')$ of the relation are denoted $\cdot \xrightarrow{a} n'$.

A skeleton is used to keep track of transitions between worlds and which observations were perceived in which worlds.

Definition 4.2.3: A *node* Γ^j_k with superscript j (the *branch* index) and subscript k (the *node* index) is a pair $\langle \Delta, \Sigma \rangle$, where Δ is a set of labeled formulae and Σ is a skeleton.

Definition 4.2.4: If one has a tree with trunk $\Gamma^0_0 = \langle \{(0, \Psi)\}, \emptyset \rangle$, we'll say one has *a tree for* Ψ.

Definition 4.2.5: A node $\Gamma = \langle \Delta, \Sigma \rangle$ is *closed* if $(x, \bot) \in \Delta$ for any $x \geq 0$. It is *open* if it is not closed. A branch is closed if and only if its leaf node is closed. A tree is closed if all of its leaf nodes are closed, else it is open.

A preprocessing step occurs, where all (sub)formulae of the form $(\forall v^a)\Phi$ and $(\forall v^o)\Phi$ are replaced by, respectively, $(\Phi|^{v^a}_{\alpha_1} \wedge \ldots \wedge \Phi|^{v^a}_{\alpha_n})$ and $(\Phi|^{v^o}_{\varsigma_1} \wedge \ldots \wedge \Phi|^{v^o}_{\varsigma_n})$.

The tableau rules for LAO follow. Let $\Gamma^j_k = \langle \Delta^j_k, \Sigma^j_k \rangle$ be a leaf node.

1. rule $\perp$: If Δ^j_k contains (x, Φ) and $(x, \neg\Phi)$, then create node $\Gamma^j_{k+1} = \langle \Delta^j_k \cup \{(x, \perp)\}, \Sigma^j_k \rangle$.
2. rule $\neg$: If Δ^j_k contains $(x, \neg\neg\Phi)$, then create node $\Gamma^j_{k+1} = \langle \Delta^j_k \cup \{(x, \Phi)\}, \Sigma^j_k \rangle$.
3. rule $\wedge$: If Δ^j_k contains $(x, \Phi \wedge \Phi')$, then create node $\Gamma^j_{k+1} = \langle \Delta^j_k \cup \{(x, \Phi), (x, \Phi')\}, \Sigma^j_k \rangle$.
4. rule $\vee$: If Δ^j_k contains $(x, \neg(\Phi \wedge \Phi'))$, then create node $\Gamma^j_{k+1} = \langle \Delta^j_k \cup \{(x, \neg\Phi)\}, \Sigma^j_k \rangle$ and node $\Gamma^{j'}_0 = \langle \Delta^j_k \cup \{(x, \neg\Phi')\}, \Sigma^j_k \rangle$, where j' is a fresh integer.
5. rule $=$: If Δ^j_k contains $(x, c = c')$ and in fact, constants c and c' are not identical, or if it contains $(x, c \neq c')$ and in fact, constants c and c' are identical, then create node $\Gamma^j_{k+1} = \langle \Delta^j_k \cup \{(x, \perp)\}, \Sigma^j_k \rangle$.
6. rule $\Diamond$: If Δ^j_k contains $(x, \neg[\varsigma \mid \alpha]\Phi)$: (i) If Δ^j_k contains $(x', \neg\Phi)$ for some label x' and $\varsigma \xrightarrow{\alpha} x' \notin \Sigma$, then create node $\Gamma^j_{k+1} = \langle \Delta^j_k, \Sigma^j_k \cup \{x \xrightarrow{\alpha} x', \varsigma \xrightarrow{\alpha} x'\} \rangle$, (ii) If Δ^j_k does not contain $(x', \neg\Phi)$ for some label x', then create node $\Gamma^j_{k+1} = \langle \Delta^j_k \cup \{(x', \neg\Phi), (x', \bigwedge_{\kappa \in \mathcal{K}} \kappa)\}, \Sigma^j_k \cup \{x \xrightarrow{\alpha} x', \varsigma \xrightarrow{\alpha} x'\} \rangle$, where x' is a fresh integer.
7. rule $\Box$: If Δ^j_k contains $(x, [\varsigma \mid \alpha]\Phi)$ and Σ^j_k contains $x \xrightarrow{\alpha} x'$ and $\varsigma \xrightarrow{\alpha} x'$, then create node $\Gamma^j_{k+1} = \langle \Delta^j_k \cup \{(x', \Phi)\}, \Sigma^j_k \rangle$.

To make explicit that the formulae in $\mathcal{K}$ are global, they are all added to each new world (fresh integer) introduced in rule $\langle \varsigma \mid \alpha \rangle$.

Below, a procedure is provided for determining whether a tableau tree is open or closed. First, some definitions and observations are required. A leaf node represents a unique (partial) LAO structure. To ensure soundness and completeness of the decision procedure, a structure must be constructed for an open tree in a particular manner.

In the following construction of a structure, given some open node, truth-values of some fluents may not be specified by the node. This is true even though, by definition, all worlds are everywhere-defined. Let $\Gamma = \langle \Delta, \Sigma \rangle$ be an arbitrary node.

Definition 4.2.6: We denote the world associated with label x as w_x if: $w_x(f) = 1$ if $(x, f) \in \Delta$ and $w_x(f) = 0$ if $(x, \neg f) \in \Delta$ (else $w_x(f)$ is unspecified), where $f \in \mathcal{F}$. We say that w_x is *fully specified* in Γ if and only if either $(x, f) \in \Delta$ or $(x, \neg f) \in \Delta$ (but not both) for every $f \in \mathcal{F}$.

Next we define when a tree is *open* or *closed*. Let $Labels(\Delta) = \{0, \ldots, k\}$ be all the labels in Δ (i.e., 0 and the k fresh integers).

Do the following in sequence.

1. Expand the tree to saturation.
2. If the tree is closed, stop. Else continue.
3. If there is an open leaf node $\Gamma = \langle \Delta, \Sigma \rangle$ for which there exists a label $x \in Labels(\Delta)$ such that w_x and $w_{x'}$ are fully specified and $w_x = w_{x'}$ where $x \neq x'$, continue. Else go to step

5. For each open leaf node $\Gamma = \langle \Delta, \Sigma \rangle$:
   - For every label $x \in \mathit{Labels}(\Delta)$, if w_x and $w_{x'}$ are fully specified and $w_x = w_{x'}$ where $x \neq x'$, then replace all x and x' in Δ and Σ by the fresh integer x''.
4. Go to step 1.
5. For each open leaf node $\Gamma = \langle \Delta, \Sigma \rangle$:
   - If for every label $x \in \mathit{Labels}(\Delta)$, w_x is fully specified, then the tree is open, stop. Else continue.
   - For every label $x \in \mathit{Labels}(\Delta)$, if neither $(x, f) \in \Delta$ nor $(x, \neg f) \in \Delta$ for some $f \in \mathcal{F}$, then create children Γ' and Γ'' of Γ, where $\Gamma' = \langle \Delta \cup \{(x, f)\}, \Sigma \rangle$ and $\Gamma'' = \langle \Delta \cup \{(x, \neg f)\}, \Sigma \rangle$.
6. Go to step 1.

Definition 4.2.7: We call the state of a tree after the decision procedure has stopped *finished*.

In essence, what the decision procedure does is create a new node for each different world a label can represent, merge labels when they represent the same world, and once all labels in a node represent fully specified worlds, a check is made of whether any of these nodes is open after all applicable rules have been applied.

Definition 4.2.8: Let $\mathcal{K}$ be a finite subset of $\mathcal{L}_{LAO}$. If a tree for $\bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \neg\Psi$ is closed, we write $\mathcal{K} \vdash_{LAO} \Psi$. If there is a *finished* open tree for $\bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \neg\Psi$, we write $\mathcal{K} \nvdash_{LAO} \Psi$.

4.3 Properties of the Decision Procedure

All proofs not given here can be found in the appendix Section A.1.

4.3.1 Soundness

Lemma 4.3.1: Let T be a finished tree. For every node $\Gamma = \langle \Delta, \Sigma \rangle$ in T: If there exists a structure $\mathcal{S}$ such that for all $w \in W$, $\mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and for every $x \in \mathit{Labels}(\Delta)$, there exists a $w' \in W$ such that for all $(x, \Phi) \in \Gamma$, $\mathcal{S}, w' \models \Phi$, then the (sub)tree rooted at Γ is open.

Theorem 4.3.1: (Soundness) If $\mathcal{K} \vdash_{LAO} \Phi$ then $\mathcal{K} \models_G \Phi$.

4.3.2 Completeness

We start with the description of the construction of a LAO structure, given the leaf node $\Gamma = \langle \Delta, \Sigma \rangle$ of a finished tree. Let $\mathcal{S} = \langle W, R, O, N, Q \rangle$ be a structure defined as follows:

- $W = \{w_x \mid \text{ for each label } x \in \mathit{Labels}(\Delta)\}$.
- $R = \{R_\alpha \mid \alpha \in \mathcal{A}\}$ s.t. $R_\alpha = \{(w_x, w_{x'}) \mid x \xrightarrow{\alpha} x' \in \Sigma\}$.
- $O = \Omega$.

- $N(\varsigma) = \varsigma$.
- $Q = \{Q_\alpha \mid \alpha \in \mathcal{A}\}$ s.t. $Q_\alpha = \{(\varsigma, w_x) \mid \varsigma \xrightarrow{\alpha} x \in \Sigma\}$.

In the definition of the possible worlds W in the construction above, one can think of each label x representing world w_x. Note that if a structure is being constructed from an open leaf node of a finished tree (after the decision procedure), all worlds associated with labels in the tree will be fully specified. Moreover, each label will represent a unique world, that is, $x, x' \in Labels(\Gamma)$ such that $x \neq x'$ if and only if $w_x, w_{x'} \in W$ such that $w_x \neq w_{x'}$.

Lemma 4.3.2: By Definition 4.1.2, if $\mathcal{S}$ is constructed as above, $\mathcal{S}$ is a LAO structure.

Proof:

1. W is nonempty because w_0 must be in W. It is finite because $W \subseteq C$, in other words, $2^{|\mathcal{F}|}$ is the maximum number of worlds;
2. R is a mapping that provides a relation $R_\alpha : W \mapsto W$ for each action $\alpha \in \mathcal{A}$;
3. $O = \Omega$ is a nonempty finite set of observations;
4. $N : \Omega \mapsto O$ is a total bijection;
5. Q is a mapping that provides a relation $Q_\alpha : O \mapsto W$ for each action $\alpha \in \mathcal{A}$;

■

Let $\Gamma = \langle \Delta, \Sigma \rangle$ be an open leaf node of a finished tree. By the first bullet point of step 5 of the decision procedure and due to Γ being an open leaf node, w_x is fully specified for every label $x \in Labels(\Delta)$ (cf. Def. 4.2.6). Moreover, by step 3, w_x is unique in that $x, x' \in Labels(\Gamma)$ such that $x \neq x'$ if and only if $w_x, w_{x'} \in W$ such that $w_x \neq w_{x'}$.

Lemma 4.3.3: Let Γ be an open leaf node of a finished tree. For all $x \in Labels(\Delta)$, if $(x, \Phi) \in \Delta$, then $\mathcal{S}, w_x \models \Phi$.

Theorem 4.3.2: (Completeness) If $\mathcal{K} \models_G \Phi$ then $\mathcal{K} \vdash_{LAO} \Phi$.

4.3.3 Termination

Definition 4.3.1: Let Φ' be a strict sub-part of Φ. A tableau rule has the *subformula property* if and only if the new node(s) (Γ') created by the application of the rule, contains (x, Φ') or $(x, \neg\Phi')$ for some x, due to applying the rule.

Lemma 4.3.4: A tree for any formula $\Phi \in \mathcal{L}_{LAO}$ becomes saturated at step 1 of the decision procedure.

Proof:
We can divide all the tableau rules into two categories: (i) those which add $\perp$ to the new node and (ii) those with the subformula property. Category-(i) rules never cause rules to become applicable later. As a direct consequence of sentences being finite and the subformula property, every category-(ii) rule must eventually become inapplicable. Therefore, all rules eventually become inapplicable, and it follows that any tree (for any formula) would become saturated. ■

Theorem 4.3.3: The entailment decision procedure for LAO terminates.

Proof:
A tree is either open or closed. If it is closed, the procedure will stop at step 2. We must show that if the tree does not become closed, step 5 will eventually be reached while for some open leaf node $\langle \Delta, \Sigma \rangle$, for every label $x \in Labels(\Delta)$, w_x is fully specified (because then the procedure stops).

Due to Lemma 4.3.4, step 1 always terminates (with a finite number of tree branches).

If the iterative part of step 3 is reached (the bullet point in step 3), the size of $Labels(\Delta)$ (for each leaf node) can only remain the same or become smaller. But could it happen that due to the application of the iterative part of step 3, one or more fresh labels are introduced in step 1 such that the iterative part of step 3 is always executed? Only rule $\Diamond$ can introduce a fresh integer, moreover, only when $(x, \neg[\varsigma \mid \alpha]\Phi) \in \Delta$ and $(x', \neg\Phi) \notin \Delta$ for some labels x, x'. But this condition will not become true when previously false simply because x or x' are changed for new integers. (By the time a tree is saturated, whenever $(x, \neg[\varsigma \mid \alpha]\Phi) \in \Delta$ for some x, $(x', \neg\Phi) \in \Delta$ for some x'.) Hence, eventually, the iterative part of step 3 becomes inapplicable for all open leaf nodes, and when the process enters step 3, it jumps to step 5. Due to successive applications of the second bullet point of step 5, for some open leaf node $\langle \Delta, \Sigma \rangle$, for every label $x \in Labels(\Delta)$, w_x is fully specified, and the procedure stops. ■

Corollary 4.3.1: The entailment problem for LAO is decidable.

Because the procedure is sound (Th. 4.3.1), complete (Th. 4.3.2) and terminating (Th. 4.3.3), entailment is decidable.

4.4 Introducing Domain Specification Concepts

Here we introduce the ideas behind specifying action rules and perception rules. In the context of LAO, we are interested in four things in the domain of interest:

(i) The initial condition, that is, a specification of the world the agent finds itself in when it becomes active. The agent's *initial condition* will be referred to as IC and is a sentence involving only fluents.

(ii) There will likely be facts about the domain that do not change; fixed laws about parts of the environment. *Static laws* are axioms about phenomena such as, 'a roof is above a floor', 'aeroplanes fly', 'my cat's name is Zoë' and so on. Refer to these laws as SL.

(iii) The dynamics of the environment or system must be specified. That is, rules about the effects of actions and about conditions for performing the actions must be provided. Refer to these *action rules* as AR.

(iv) The observability of the environment must be specified. That is, rules about which observations are perceivable in which situations (worlds). Refer to these *perception rules* as PR.

The union of SL, AR and PR is referred to as an agent's *background knowledge* and is denoted BK. The main task in LAO is to determine whether some sentence of interest $\Phi \in \mathcal{L}_{LAO}$ is true in the initial condition, given an agent's background knowledge, that is, whether $BK \models_G IC \rightarrow \Phi$.

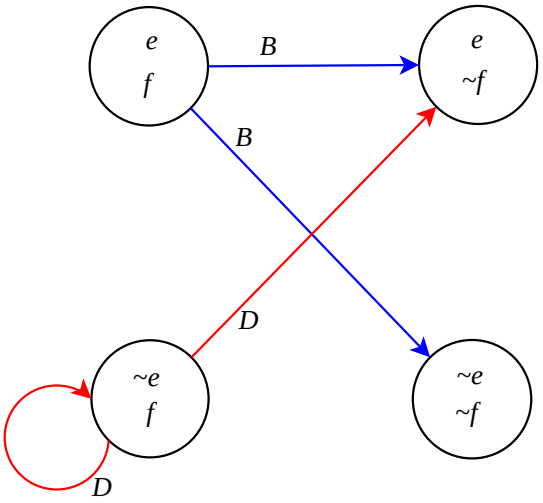


Fig. 4.2: An example of four possible worlds with two nondeterministic actions, B and D. It is assumed that no action is executable from worlds in which f is false. Tilde ($\sim$) represents negation.

These ideas are expanded upon, with the help of the oil-drinking scenario, in Section 4.5. For now, to simplify the exposition, we refer to the transition diagram depicted in Figure 4.2 instead of the oil-drinking scenario.

4.4.1 Action

Figure 4.2 is a simple example with four conceivable worlds—determined by two fluents e and f—and two nondeterministic actions B and D. Observations are not mentioned in the figure, but they need not be considered when developing a theory of the effects of actions. More will be said about observations later in this section.

How can we capture the behavior of B? The following axiom seems reasonable.

$$e \wedge f \rightarrow \langle B \rangle (e \wedge \neg f) \wedge \langle B \rangle (\neg e \wedge \neg f). \tag{4.1}$$

That is, whenever I am in a $(e \wedge f)$-world (i.e., a world which satisfies $e \wedge f$), it is possible to end up in a $(e \wedge \neg f)$-world after executing action B and it is possible to end up in a $(\neg e \wedge \neg f)$-world after executing action B.

One might say that Equation 4.1 leads to a contradiction, but in modal logic, $\{\langle \alpha \rangle \varphi \wedge \langle \alpha \rangle \varphi'\} \not\models_G \langle \alpha \rangle (\varphi \wedge \varphi')$.

As a matter of interest, expressing the effects of an action can usually be done in several ways. For instance, the effects of B could also be written as

$$e \wedge f \rightarrow \langle B \rangle e \wedge \langle B \rangle \neg e \wedge [B] \neg f.$$

To be able to infer what effects B cannot have and what happens when preconditions are not met, we need axioms to capture completeness assumptions, that is, axioms to guarantee that what we have said about the behavior of B, *completely* specifies its behavior.

Adding the 'reverse' implication

$$e \wedge f \leftarrow \langle B\rangle(e \wedge \neg f) \wedge \langle B\rangle(\neg e \wedge \neg f)$$

which is semantically equivalent to

$$\neg(e \wedge f) \rightarrow \neg(\langle B\rangle(e \wedge \neg f) \wedge \langle B\rangle(\neg e \wedge \neg f))$$

which is semantically equivalent to

$$\neg e \vee \neg f \rightarrow [B](\neg e \vee f) \vee [B](e \vee f)$$

will not work. This states that if condition $\neg e \vee \neg f$ holds, then it is always the case that $\neg e \vee f$ or it is always the case that $e \vee f$, which is not what we want. To express that it is impossible for effect $e \wedge \neg f$ or effect $\neg e \wedge \neg f$ *not* to occur under condition $e \wedge f$, one can write

$$e \wedge f \rightarrow \neg\langle B\rangle\neg((e \wedge \neg f) \vee (\neg e \wedge \neg f)). \tag{4.2}$$

In other words, to express that *all* the effects of performing B in a $(e \wedge f)$-world are $e \wedge \neg f$ and $\neg e \wedge \neg f$, one can write (4.2) or

$$e \wedge f \rightarrow [B]((e \wedge \neg f) \vee (\neg e \wedge \neg f)). \tag{4.3}$$

Axioms (4.2) and (4.3) are semantically equivalent. This is the *effect closure* of Axiom (4.1).

To express that $e \wedge f$ is the *only* condition under which action B can have an effect, we need an axiom which states that under any other conditions, there are no effects. Another way of saying this is to say that B cannot be executed when the agent is not in the $(e \wedge f)$-world:

$$\neg(e \wedge f) \rightarrow [B]\bot \tag{4.4}$$

which is semantically equivalent to

$$\neg e \vee \neg f \rightarrow \neg\langle B\rangle\top \tag{4.5}$$

which is semantically equivalent to

$$\langle B\rangle\top \rightarrow e \wedge f. \tag{4.6}$$

In fact, the general form of an axiom to state that action α is inexecutable under condition ϕ, is

$$\phi \rightarrow \neg\langle\alpha\rangle\top.$$

Axioms (4.4), (4.5) and (4.6) are each the *condition closure* of Axiom (4.1).

Notice that Axiom (4.1) entails $e \wedge f \rightarrow \langle B\rangle\top$. In general, an axiom to state that action α is executable under condition ϕ, has the form

$$\phi \rightarrow \langle\alpha\rangle\top.$$

Therefore, effect axioms automatically specify the conditions under which actions can be exe-

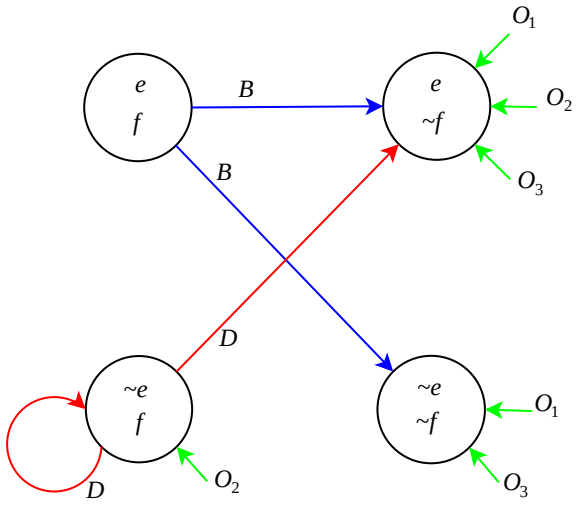


Fig. 4.3: An example indicating which observations (O_1, O_2, O_3, O_4) are perceivable in which possible worlds.

cuted. Hence, we could have written $e \wedge f \leftrightarrow \langle B \rangle \top$ instead of Axiom (4.4), but it is not necessary.

To summarize, the full specification of the transition model represented by Figure 4.2 follows.

$$e \wedge f \rightarrow \langle B \rangle (e \wedge \neg f) \wedge \langle B \rangle (\neg e \wedge \neg f) \qquad \neg e \wedge f \rightarrow \langle D \rangle (\neg e \wedge f) \wedge \langle D \rangle (e \wedge \neg f)$$
$$e \wedge f \rightarrow [B]((e \wedge \neg f) \vee (\neg e \wedge \neg f)) \qquad \neg e \wedge f \rightarrow [D]((\neg e \wedge f) \vee (e \wedge \neg f))$$
$$\langle B \rangle \top \rightarrow e \wedge f \qquad \langle D \rangle \top \rightarrow \neg e \wedge f.$$

4.4.2 *Perception*

Figure 4.3 is the same transition diagram as Figure 4.2, but with the information about observations added. Figure 4.3 shows that four observations (O_1, O_2, O_3, O_4) are perceivable. A short (in this example, green) arrow entering a world from nowhere indicates that the observation which labels the arrow is perceivable in that world. There are a few things that need pointing out. In every reachable world (via B or D), at least one observation is perceivable. No perception is possible in unreachable worlds. There is no fundamental constraint on which observations are perceivable in reachable worlds.

Axioms describing when these observations are perceivable follow.

$$\langle O_1 \mid B \rangle (e \wedge \neg f) \qquad \langle O_1 \mid D \rangle (e \wedge \neg f)$$
$$\langle O_2 \mid B \rangle (e \wedge \neg f) \qquad \langle O_2 \mid D \rangle (e \wedge \neg f)$$
$$\langle O_3 \mid B \rangle (e \wedge \neg f) \qquad \langle O_3 \mid D \rangle (e \wedge \neg f)$$
$$\langle O_1 \mid B \rangle (\neg e \wedge \neg f) \qquad \langle O_2 \mid D \rangle (\neg e \wedge f)$$
$$\langle O_3 \mid B \rangle (\neg e \wedge \neg f)$$

Stating which observations are imperceivable (in the reachable worlds) is done with the following

axioms.

$$\neg\langle O_2 \mid B\rangle(\neg e \wedge \neg f) \qquad \neg\langle O_1 \mid D\rangle(\neg e \wedge f) \qquad \neg\langle O_3 \mid D\rangle(\neg e \wedge f)$$

4.5 Specifying the Oil-drinking Scenario

In this section, we completely specify or model the oil-drinking scenario. Some issues in domain specification, in general, which were not covered in the two previous sections, will be discussed here. These issues are easier to explain with the aid of the scenario.

The set of fluents is $\mathcal{F} = \{\texttt{full}, \texttt{drank}, \texttt{holding}\}$, denoted f, d and h, respectively, in the figures. Figures 4.4, 4.5, 4.6 and 4.7 are the transition diagrams for the scenario, corresponding to the four actions the robot can perform (the $\sim$ indicates negation of a fluent). The set of actions is $\mathcal{A} = \{\texttt{grab}, \texttt{weigh}, \texttt{drink}, \texttt{replace}\}$. The set of observations is $\Omega = \{\texttt{ObsNil}, \texttt{obsLight}, \texttt{obsMedium}, \texttt{obsHeavy}\}$.

What does it mean to say the robot drank the oil? Normally, it would mean that the robot had the oil-can in its gripper, the can was initially full and the oil was successfully poured into its mouth (the robot did not miss its mouth). But what action would one say the robot performed if its arm and gripper made the same movements as in a 'successful' drinking action, but without the oil-can in its gripper, having initially missed grabbing it? We would at least want to say that the robot *attempted* a drinking action. However, the successful drinking attempt is also an attempt. Suppose the oil-can is under a table, the table top's hight is below the robot's mouth and the robot is standing against the table. Suppose further that the robot successfully grabs the can under the table, raises its arm towards its mouth but is barred by the table top (from underneath) from bringing the can to its mouth. We would say that the robot had the *intention* of drinking the oil, although it was a failed attempt. A final example: suppose some unknown agent took the oil-can and poured the oil into the robot's mouth. One could say that the robot drank the oil, but then one is talking about effects, not about the robot's physical activity or intended physical activity. Our approach to modeling actions is thus the following.

> An action is the attempt to execute a recognized physical activity. When it is said that an action was performed, it means the action was attempted, independent of success or failure. The success/failure of the action is measured by the effects of the action.

What we mean by "recognized" is that the physical activity is a skill the agent knows it has and it has a name for the skill.

So even if there is no oil-can in the robot's gripper, if the robot initiated the `drink` action, technically speaking, one may say that the robot drank. However, to avoid confusion while talking about actions in natural language, one should refrain from saying the robot 'drank' without adding whether it was a success or a failed attempt—unless it is clear in the particular discussion that only action attempts are being discussed.

A related issue is *executability*. In our work, executability is with respect to the *ontic* view, not the *deontic* view. In other words, independent of an action being permissible (deontic view), if it is physically possible (ontic view), it is executable. For instance, the action of poking your finger in

your eye is possible (executable), but you usually *decide* not to. Our second principle for actions is thus the following.

> An action is *executable* in a world if and only if the situation represented by the world implies that the action is physically possible.

What constitutes physical possibility of actions is up to the knowledge engineer to decide. However, such design choices must be clearly stated and motivated, and consistently applied.

We do not claim that the following specification is the only correct way. Due to the limited number of fluents, actions and observations, what can be expressed is also limited. In other words, one could have got closer to a satisfactory specification of the scenario with more objects in the vocabulary (given the constraints of the syntax of LAO), however, for illustration purposes, we have kept the vocabulary as small as possible without being (completely) trivial.

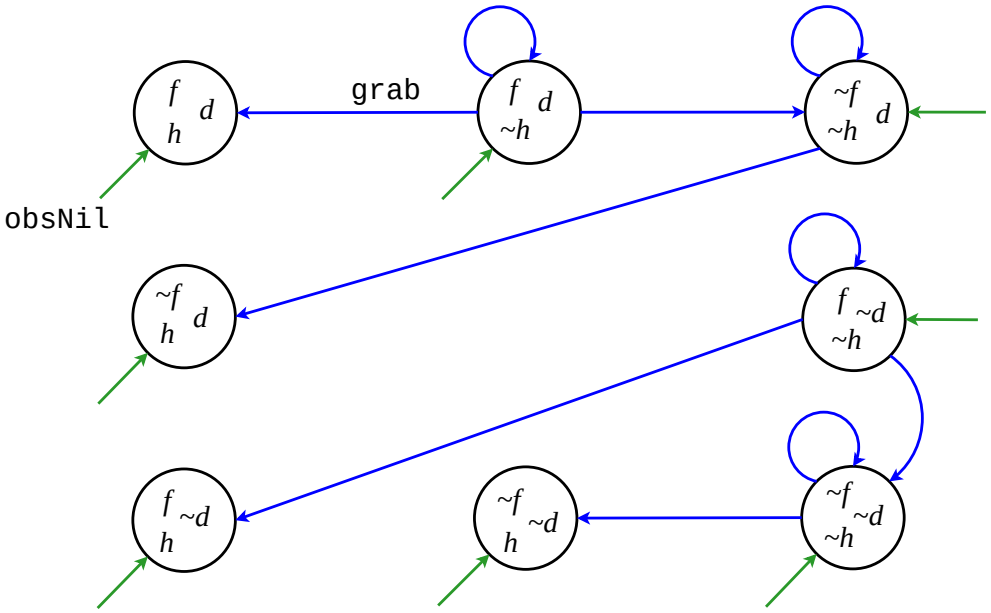


Fig. 4.4: Transition diagram for `grab`. All blue arcs represent transitions due to the action. All green arrows represent an `obsNil` observation.

The robot can grab at the oil-can in the four worlds where it is not holding anything. Due to the possibility of knocking the can and spilling some oil, and the possibility of not grabbing hold of the can, and due to the uncertainty of what will happen, more than one world is reachable from any of the four worlds in which `grab` is executable. If the robot successfully grabs the can, it did not knock it and spill the oil.

The robot will only attempt a `drink` action when holding the oil-can. It is arguable whether it is physically possible to attempt drinking when not holding the can. We have made the design choice that it is impossible. The only reachable worlds via `drink` are where the oil-can is not full and the

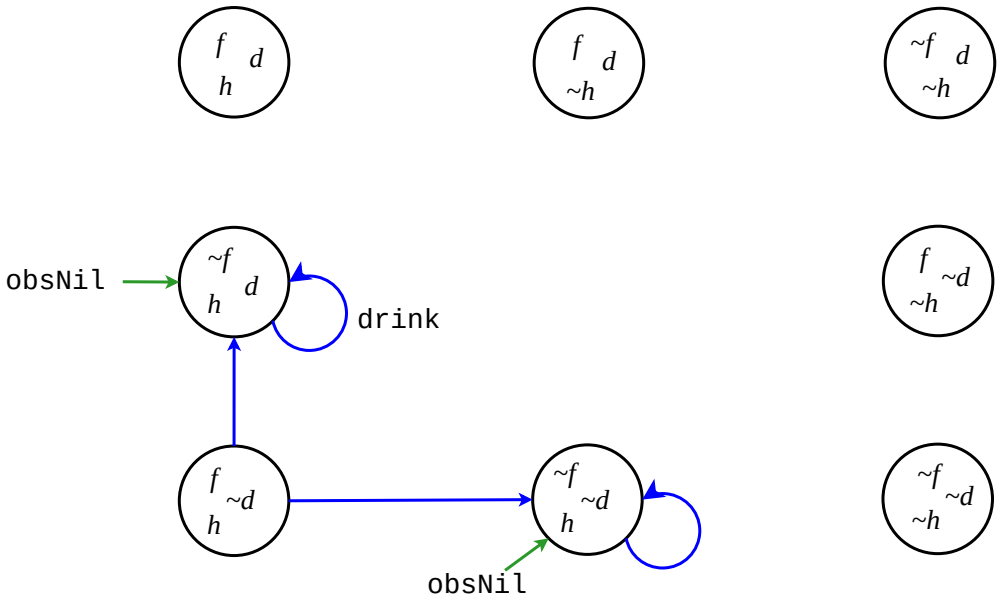


Fig. 4.5: Transition diagram for `drink`. All blue arcs represent transitions due to the action.

robot is holding the can. Drinking always results in the can being empty, because either the robot successfully consumes the oil, or misses its mouth and spills the oil. Note that when the can is full and the robot has not drunk the oil, then after a `drink` action, either `drank` will be true, or it will be false when the robot misses its mouth.

When replacing the oil-can, the robot never spills the oil (if the can contains oil). Here it is assumed that the robot to be modeled is expert at placing the can on the floor.

The motivation for modeling `weigh` as represented in Figure 4.7 is as follows. Firstly, `weigh` is a sensory action; it does not change the state of the world (hence, the loop arcs). However, the robot can tell which observations are perceivable in the four reachable worlds. Weighing can only occur when the oil-can is being held. In the $(f \wedge d \wedge h)$-world, it is a contradiction to say that the can is full and the robot has drunk the oil. The best way to reflect this 'chaotic' situation is to say that all observations associated with `weigh` are perceivable. (The reason why `ObsNil` is not associated with `weigh` is discussed a little later.) In the $(\neg f \wedge d \wedge h)$-world, it seems highly likely that there is no more oil in the can: it has either been drunk or spilt. The can thus weighs light. In the $(f \wedge \neg d \wedge h)$-world, the fact that the oil-can is said to be full, means that nothing has been spilt. Moreover, due to `drank` being negative, the `drink` action has probably not been attempted. The can is thus very likely completely full and heavy. The $(\neg f \wedge \neg d \wedge h)$-world seems to indicate that some oil was spilt out of the can when the can was grabbed. The can could have been knocked over, spilling all the oil, or just bumped, spilling a little bit of oil. Either way, if the can is not full, it is not heavy.

Besides the more obvious information represented, one can glean from the diagrams that

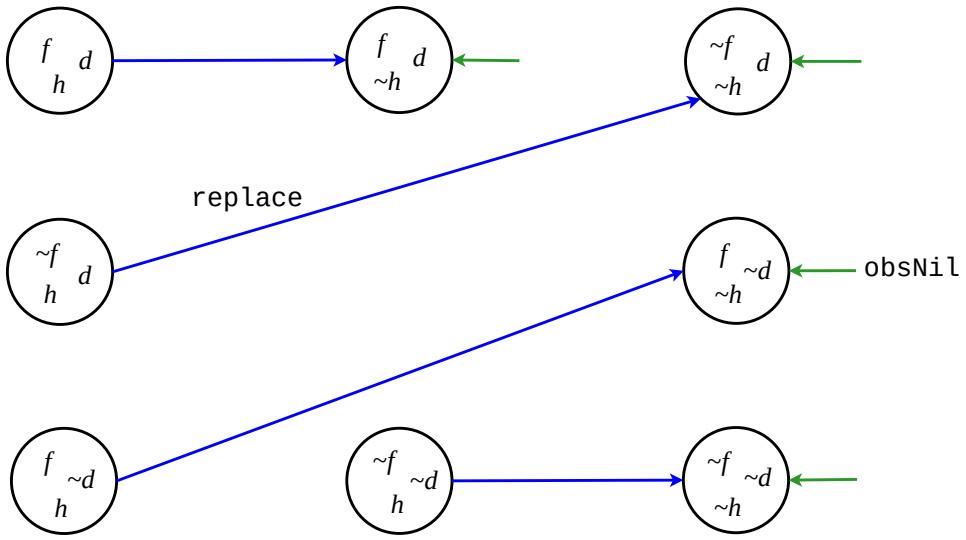


Fig. 4.6: Transition diagram for `replace`. All blue arcs represent transitions due to the action.

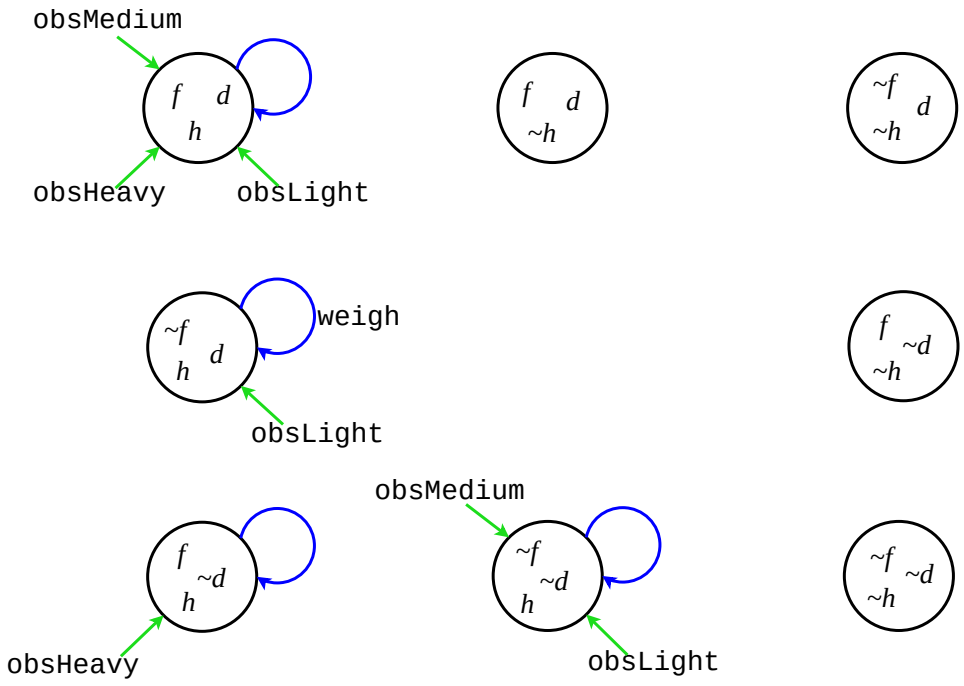


Fig. 4.7: Transition diagram for `weigh`. All blue arcs represent transitions due to the action.

- `drank` is invariant with `grab`.
- All fluents are invariant with `weigh`.
- `full` and `drank` are both invariant with `replace`.

We now write the effect axioms, effect closure axioms and inexecutability axioms for the four actions.

4.5.1 Effect Axioms

Effect axioms have the form

$$\phi \rightarrow \langle\alpha\rangle\varphi_1 \wedge \langle\alpha\rangle\varphi_2 \wedge \cdots \wedge \langle\alpha\rangle\varphi_k,$$

where ϕ is the condition for effects $\varphi_1, \varphi_2, \ldots, \varphi_k$, given α is executed in a ϕ-world. For every action, there is a set of such axioms, an axiom for each condition.

$$\begin{aligned}
f \wedge d \wedge \neg h &\rightarrow \langle\texttt{grab}\rangle(f \wedge d \wedge h) \wedge \langle\texttt{grab}\rangle(f \wedge d \wedge \neg h) \wedge \langle\texttt{grab}\rangle(\neg f \wedge d \wedge \neg h)\\
\neg f \wedge d \wedge \neg h &\rightarrow \langle\texttt{grab}\rangle(\neg f \wedge d \wedge h) \wedge \langle\texttt{grab}\rangle(\neg f \wedge d \wedge \neg h)\\
f \wedge \neg d \wedge \neg h &\rightarrow \langle\texttt{grab}\rangle(f \wedge \neg d \wedge h) \wedge \langle\texttt{grab}\rangle(f \wedge \neg d \wedge \neg h) \wedge \langle\texttt{grab}\rangle(\neg f \wedge \neg d \wedge \neg h)\\
\neg f \wedge \neg d \wedge \neg h &\rightarrow \langle\texttt{grab}\rangle(\neg f \wedge \neg d \wedge h) \wedge \langle\texttt{grab}\rangle(\neg f \wedge \neg d \wedge \neg h)\\
\neg f \wedge d \wedge h &\rightarrow \langle\texttt{drink}\rangle(\neg f \wedge d \wedge h)\\
f \wedge \neg d \wedge h &\rightarrow \langle\texttt{drink}\rangle(\neg f \wedge d \wedge h) \wedge \langle\texttt{drink}\rangle(\neg f \wedge \neg d \wedge h)\\
\neg f \wedge \neg d \wedge h &\rightarrow \langle\texttt{drink}\rangle(\neg f \wedge \neg d \wedge h)\\
f \wedge d \wedge h &\rightarrow \langle\texttt{replace}\rangle(f \wedge d \wedge \neg h)\\
\neg f \wedge d \wedge h &\rightarrow \langle\texttt{replace}\rangle(\neg f \wedge d \wedge \neg h)\\
f \wedge \neg d \wedge h &\rightarrow \langle\texttt{replace}\rangle(f \wedge \neg d \wedge \neg h)\\
\neg f \wedge \neg d \wedge h &\rightarrow \langle\texttt{replace}\rangle(\neg f \wedge \neg d \wedge \neg h)\\
f \wedge d \wedge h &\rightarrow \langle\texttt{weigh}\rangle(f \wedge d \wedge h)\\
\neg f \wedge d \wedge h &\rightarrow \langle\texttt{weigh}\rangle(\neg f \wedge d \wedge h)\\
f \wedge \neg d \wedge h &\rightarrow \langle\texttt{weigh}\rangle(f \wedge \neg d \wedge h)\\
\neg f \wedge \neg d \wedge h &\rightarrow \langle\texttt{weigh}\rangle(\neg f \wedge \neg d \wedge h)
\end{aligned}$$

4.5.2 Effect Closure Axioms

For each effect axiom, there must be an associated effect closure axiom of the form

$$\phi \rightarrow [\alpha](\varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_k)$$

or a semantically equivalent sentence.[2] Effect closure axioms are similar to but not the same as frame axioms. Frame axioms are not used in this chapter; see the discussion in the last section.

$$
\begin{aligned}
f \wedge d \wedge \neg h &\rightarrow [\texttt{grab}]((f \wedge d) \vee (\neg f \wedge d \wedge \neg h)) \\
\neg f \wedge d \wedge \neg h &\rightarrow [\texttt{grab}](\neg f \wedge d) \\
f \wedge \neg d \wedge \neg h &\rightarrow [\texttt{grab}]((f \wedge \neg d) \vee (\neg f \wedge \neg d \wedge \neg h)) \\
\neg f \wedge \neg d \wedge \neg h &\rightarrow [\texttt{grab}](\neg f \wedge \neg d) \\
\neg f \wedge d \wedge h &\rightarrow [\texttt{drink}](\neg f \wedge d \wedge h) \\
f \wedge \neg d \wedge h &\rightarrow [\texttt{drink}](\neg f \wedge h) \\
\neg f \wedge \neg d \wedge h &\rightarrow [\texttt{drink}](\neg f \wedge \neg d \wedge h) \\
f \wedge d \wedge h &\rightarrow [\texttt{replace}](f \wedge d \wedge \neg h) \\
\neg f \wedge d \wedge h &\rightarrow [\texttt{replace}](\neg f \wedge d \wedge \neg h) \\
f \wedge \neg d \wedge h &\rightarrow [\texttt{replace}](f \wedge \neg d \wedge \neg h) \\
\neg f \wedge \neg d \wedge h &\rightarrow [\texttt{replace}](\neg f \wedge \neg d \wedge \neg h) \\
f \wedge d \wedge h &\rightarrow [\texttt{weigh}](f \wedge d \wedge h) \\
\neg f \wedge d \wedge h &\rightarrow [\texttt{weigh}](\neg f \wedge d \wedge h) \\
f \wedge \neg d \wedge h &\rightarrow [\texttt{weigh}](f \wedge \neg d \wedge h) \\
\neg f \wedge \neg d \wedge h &\rightarrow [\texttt{weigh}](\neg f \wedge \neg d \wedge h)
\end{aligned}
$$

4.5.3 *Inexecutability Axioms*

We now extend the discussion about condition closure started in Section 4.4.1. Suppose there are ℓ effect axioms for action α, with conditions $\phi_1, \phi_2, \ldots, \phi_\ell$, respectively. Then, assuming that effect axioms are meant to say all there is to be said about actions (the completeness assumption), we want to express that if a world does not satisfy one of the ℓ conditions, then it is not possible to execute. This can be written as

$$\neg(\phi_1 \vee \phi_2 \vee \cdots \vee \phi_\ell) \rightarrow \neg\langle\alpha\rangle\top$$

or

$$\langle\alpha\rangle\top \rightarrow (\phi_1 \vee \phi_2 \vee \cdots \vee \phi_\ell).$$

Often $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_\ell$ has a compact equivalent form.

[2] Note that one does not employ effect closure axioms in the subsequent logics, because of the extra information provided by probabilities.

$$\begin{aligned} \langle \texttt{grab} \rangle \top &\rightarrow \neg h \\ \langle \texttt{drink} \rangle \top &\rightarrow h \wedge \neg(f \wedge d) \\ \langle \texttt{replace} \rangle \top &\rightarrow h \\ \langle \texttt{weigh} \rangle \top &\rightarrow h \end{aligned}$$

Incidentally, the inverse of the inexecutability axioms state the executability of the incumbent action. We know that, for instance, `grab` is executable only under the condition that `holding` is false; given the completeness assumption, we thus know that `grab` is inexecutable when `holding` is true. That is,

$$\texttt{holding} \rightarrow \neg\langle \texttt{grab} \rangle \top.$$

However, this is semantically equivalent to

$$\langle \texttt{grab} \rangle \top \rightarrow \neg h.$$

Hence,

Remark 4.5.1: Due to inexecutability axioms being derived from the process of condition closure, they can also be called *condition closure axioms*.

4.5.4 Perceivability Axioms

Next we discuss the specification of perception. Let $E(\alpha) = \{\varphi_1, \varphi_2, \ldots, \varphi_{k\times\ell}\}$ be the set of all effects of action α executed under all executable conditions. Then the *perceivability axioms* for α are

$$\begin{aligned} &(\forall v^o)\ v^o = \varsigma_{11} \vee v^o = \varsigma_{12} \vee \cdots \vee v^o = \varsigma_{1m} \leftrightarrow \langle v^o \mid \alpha \rangle \phi_1 \\ &(\forall v^o)\ v^o = \varsigma_{21} \vee v^o = \varsigma_{22} \vee \cdots \vee v^o = \varsigma_{2m} \leftrightarrow \langle v^o \mid \alpha \rangle \phi_2 \\ &\vdots \\ &(\forall v^o)\ v^o = \varsigma_{11} \vee v^o = \varsigma_{12} \vee \cdots \vee v^o = \varsigma_{1m} \leftrightarrow \langle v^o \mid \alpha \rangle \phi_j, \end{aligned}$$

where, (i) $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_j \equiv \bigvee_{\varphi \in E(\alpha)} \varphi$ and (ii) for any pair ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$.

The first axiom, for instance, expresses that in a ϕ_1-world, given α was executed, $\varsigma_{11}, \varsigma_{12}, \ldots, \varsigma_{1m}$ are perceivable, and any other observations are imperceivable when ϕ_1.

It is convenient to define *ontic* (physical) actions and *sensory* actions. Ontic actions have intentional ontic effects, that is, effects on the environment that were the main intention of the agent. In every world reached via an ontic action, there is exactly one particular perception, that is, the null perception, or the perception of the special observation $obsNil$. Sensory actions result in perceptions beside $obsNil$, and might only have (unintended) side-effects. For now, however, we constrain all sensory actions to have no side-effects.

Suppose α^{ont} is an ontic action. Then it will have perception axioms of the form

$$
\begin{aligned}
&(\forall v^o)\ v^o = \texttt{obsNil} \leftrightarrow \langle v^o \mid \alpha^{ont} \rangle \varphi_1 \\
&(\forall v^o)\ v^o = \texttt{obsNil} \leftrightarrow \langle v^o \mid \alpha^{ont} \rangle \varphi_2 \\
&\vdots \\
&(\forall v^o)\ v^o = \texttt{obsNil} \leftrightarrow \langle v^o \mid \alpha^{ont} \rangle \varphi_m,
\end{aligned}
$$

which can be written as

$$
\begin{aligned}
&(\forall v^o)\ v^o = \texttt{obsNil} \rightarrow \langle v^o \mid \alpha^{ont} \rangle \varphi_1 \\
&(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle \varphi_1 \rightarrow v^o = \texttt{obsNil} \\
&(\forall v^o)\ v^o = \texttt{obsNil} \rightarrow \langle v^o \mid \alpha^{ont} \rangle \varphi_2 \\
&(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle \varphi_2 \rightarrow v^o = \texttt{obsNil} \\
&\vdots \\
&(\forall v^o)\ v^o = \texttt{obsNil} \rightarrow \langle v^o \mid \alpha^{ont} \rangle \varphi_m \\
&(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle \varphi_m \rightarrow v^o = \texttt{obsNil},
\end{aligned}
$$

which can be written as

$$
\begin{aligned}
&(\forall v^o)\ v^o = \texttt{obsNil} \rightarrow \langle v^o \mid \alpha^{ont} \rangle \varphi_1 \wedge \langle v^o \mid \alpha^{ont} \rangle \varphi_2 \wedge \cdots \wedge \langle v^o \mid \alpha^{ont} \rangle \varphi_m \\
&(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle \varphi_1 \rightarrow v^o = \texttt{obsNil} \\
&(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle \varphi_2 \rightarrow v^o = \texttt{obsNil} \\
&\vdots \\
&(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle \varphi_m \rightarrow v^o = \texttt{obsNil},
\end{aligned}
$$

which can be written as

$$
\begin{aligned}
&\langle \texttt{obsNil} \mid \alpha^{ont} \rangle \varphi_1 \wedge \langle \texttt{obsNil} \mid \alpha^{ont} \rangle \varphi_2 \wedge \cdots \wedge \langle \texttt{obsNil} \mid \alpha^{ont} \rangle \varphi_m \\
&(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle \varphi_1 \vee \langle v^o \mid \alpha^{ont} \rangle \varphi_2 \vee \cdots \vee \langle v^o \mid \alpha^{ont} \rangle \varphi_m \rightarrow v^o = \texttt{obsNil},
\end{aligned}
$$

which can be written as

$$
\langle \texttt{obsNil} \mid \alpha^{ont} \rangle \varphi_1 \wedge \langle \texttt{obsNil} \mid \alpha^{ont} \rangle \varphi_2 \wedge \cdots \wedge \langle \texttt{obsNil} \mid \alpha^{ont} \rangle \varphi_m \tag{4.7}
$$

$$
(\forall v^o)\ \langle v^o \mid \alpha^{ont} \rangle (\varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_m) \rightarrow v^o = \texttt{obsNil}. \tag{4.8}
$$

Often $\varphi_1 \vee \varphi_2 \vee \cdots \vee \varphi_m$ has a compact equivalent form.

The perceivability axioms for ontic actions can thus be written in this compact form (4.7 and 4.8). Due to sensory actions typically involving more than one observation and the observations possibly being different for different worlds, such a generally applicable compact form cannot be suggested for sensory actions. The perceivability axioms for the oil-drinking scenario follow.

$$\langle \texttt{obsNil} \mid \texttt{grab} \rangle (f \wedge d \wedge h) \wedge \langle \texttt{obsNil} \mid \texttt{grab} \rangle (f \wedge d \wedge \neg h) \wedge$$
$$\langle \texttt{obsNil} \mid \texttt{grab} \rangle (f \wedge \neg d \wedge h) \wedge \langle \texttt{obsNil} \mid \texttt{grab} \rangle (f \wedge \neg d \wedge \neg h) \wedge$$
$$\langle \texttt{obsNil} \mid \texttt{grab} \rangle (\neg f \wedge d \wedge h) \wedge \langle \texttt{obsNil} \mid \texttt{grab} \rangle (\neg f \wedge d \wedge \neg h) \wedge$$
$$\langle \texttt{obsNil} \mid \texttt{grab} \rangle (\neg f \wedge \neg d \wedge h) \wedge \langle \texttt{obsNil} \mid \texttt{grab} \rangle (\neg f \wedge \neg d \wedge \neg h)$$
$$(\forall v^o)\ \langle v^o \mid \texttt{grab} \rangle \top \rightarrow v^o = \texttt{obsNil}$$
$$\langle \texttt{obsNil} \mid \texttt{drink} \rangle (\neg f \wedge d \wedge h) \wedge \langle \texttt{obsNil} \mid \texttt{drink} \rangle (\neg f \wedge \neg d \wedge h)$$
$$(\forall v^o)\ \langle v^o \mid \texttt{drink} \rangle (\neg f \wedge h) \rightarrow v^o = \texttt{obsNil}$$
$$\langle \texttt{obsNil} \mid \texttt{replace} \rangle (f \wedge d \wedge \neg h) \wedge \langle \texttt{obsNil} \mid \texttt{replace} \rangle (f \wedge \neg d \wedge \neg h) \wedge$$
$$\langle \texttt{obsNil} \mid \texttt{replace} \rangle (\neg f \wedge d \wedge \neg h) \wedge \langle \texttt{obsNil} \mid \texttt{replace} \rangle (\neg f \wedge \neg d \wedge \neg h)$$
$$(\forall v^o)\ \langle v^o \mid \texttt{replace} \rangle \neg h \rightarrow v^o = \texttt{obsNil}$$
$$(\forall v^o)\ v^o \neq \texttt{obsNil} \leftrightarrow \langle v^o \mid \texttt{weigh} \rangle (f \wedge d \wedge h)$$
$$(\forall v^o)\ v^o = \texttt{obsLight} \leftrightarrow \langle v^o \mid \texttt{weigh} \rangle (\neg f \wedge d \wedge h)$$
$$(\forall v^o)\ v^o = \texttt{obsHeavy} \leftrightarrow \langle v^o \mid \texttt{weigh} \rangle (f \wedge \neg d \wedge h)$$
$$(\forall v^o)\ v^o = \texttt{obsLight} \vee v^o = \texttt{obsMedium} \leftrightarrow \langle v^o \mid \texttt{weigh} \rangle (\neg f \wedge \neg d \wedge h).$$

4.6 Example Entailments

This section includes several examples of the decision procedure at work, all involving our oil-drinking scenario. The main task in LAO is to determine whether an arbitrary sentence Φ is implied by the initial condition IC, given the agent's background knowledge of static laws SL, action rules AR and perception rules PR. That is, we want to determine whether

$$BK \models_G IC \rightarrow \Phi,$$

where $BK = SL \cup AR \cup PR$.

As explained in the previous chapter, the trunk of the tree must be

$$\langle \{(0, \bigwedge_{\kappa \in BK} \kappa \wedge \neg(IC \rightarrow \Phi))\}, \emptyset \rangle,$$

which can be written as

$$\langle \{(0, \kappa_1), (0, \kappa_2), \ldots, (0, \kappa_z), (0, IC), (0, \neg\Phi)\}, \emptyset \rangle$$

with some pre-processing, where $BK = \{\kappa_1, \kappa_2, \ldots, \kappa_z\}$.

For the examples in this section, let SL be empty, AR consists of the effect axioms, effect closure axioms and inexecutability axioms, and PR consists of the perceivability axioms of the oil-drinking scenario presented in Section 4.5. Let $IC = \{f \wedge \neg d \wedge \neg h\}$.

Tables 4.1 to 4.8 depict the tableaux of the different examples. In the examples, to shorten and clarify proofs, we shall use syntactic abbreviations, and we shall not show every rule application,

as long as the steps remain clear. The 'Comment' column mentions the rule applied and the numbers in the 'Comment' column refer to the line to which the rule was applied. Furthermore, the following abbreviations for constants will be used: `grab` $:= g$, `drink` $:= di$, `weigh` $:= w$, `replace` $:= r$, `full` $:= f$, `drank` $:= d$, `holding` $:= h$, `obsNil` $:= oN$, `obsHeavy` $:= oH$, `obsMedium` $:= oM$ and `obsLight` $:= oL$.

Standard logical equivalences will be used to transform formulae into more 'normal' forms: "nf. x" in the 'Comment' column means that 'normal forming' was applied to line x. "rule $R\ell$ xw.z" means that rule $R\ell$ was applied to a formula in line x with the aid of an element of Σ in line z. If there is not enough space in the 'Comment' column, the comment will be written just beneath the applicable node.

Occurrences of sentences of the form

$$(x, \ell_1 \wedge \ell_2 \wedge \cdots \wedge \ell_k \rightarrow \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m)$$

will be processed into

$$(x, \bar{\ell_1}) \text{ OR } (x, \bar{\ell_2}) \text{ OR } \cdots \text{ OR } (x, \bar{\ell_k}) \text{ OR } (x, \Phi_1), (x, \Phi_2), \cdots (x, \Phi_m),$$

where ℓ_i is a fluent literal and $\bar{\ell_i}$ is a fluent literal with polarity opposite to ℓ_i, and OR indicates branching.

Line	Γ & Σ				Comment
1	$(0,\kappa_1),\ (0,\kappa_2),\ \ldots,\ (0,\kappa_z),\ (0,IC),\ (0,\neg\neg[g]h)$				trunk
2	$(0, f \wedge \neg d \wedge \neg h \rightarrow \cdots \wedge \langle g\rangle(f \wedge \neg d \wedge \neg h) \wedge \cdots),\ (0,f),\ (0,\neg d),\ (0,\neg h),\ (0,[g]h)$				rule $\wedge$ & nf. 1
3	$(0,\neg f)$	$(0,d)$	$(0,h)$	$(0,\langle g\rangle(f \wedge \neg d \wedge \neg h))$	rule $\vee$ & nf. 2
4	$(0,\bot)$	$(0,\bot)$	$(0,\bot)$	$(1,f),\ (1,\neg d),\ (1,\neg h),\ 0 \xrightarrow{g} 1,\ \ldots$	rule $\Diamond$ & $\wedge$ 3
5	rule $\bot$ 2,3	rule $\bot$ 2,3	rule $\bot$ 2,3	$(1,h)$	rule $\Box$ 1w.4
6				$(1,\bot)$	rule $\bot$ 4,5

Tab. 4.1: Proof that `BK` $\models_G$ `IC` $\rightarrow \langle$`grab`$\rangle\neg$`holding`.

Table 4.1 proves that it is possible for the robot not to be holding the oil-can (to have missed it) after grabbing at it in the initial condition.

Line	Γ & Σ							Comment
1	$(0,\kappa_1),\ (0,\kappa_2),\ \ldots,\ (0,\kappa_z),\ (0,h),\ (0,\neg[r]\neg h))$							trunk
2	$(1,h),\ 0 \xrightarrow{r} 1,\ oN \xrightarrow{r} 1,\ oH \xrightarrow{r} 1,\ oM \xrightarrow{r} 1,\ oL \xrightarrow{r} 1$							rule $\Diamond$ 1
3	$(0, \neg f \wedge \neg d \wedge h \rightarrow [r](\neg f \wedge \neg d \wedge \neg h))$							AR
4	$(0, f \wedge d \wedge h \rightarrow [r](f \wedge d \wedge \neg h))$							AR
5	$(0, \neg f \wedge d \wedge h \rightarrow [r](\neg f \wedge d \wedge \neg h))$							AR
6	$(0, f \wedge \neg d \wedge h \rightarrow [r](f \wedge \neg d \wedge \neg h))$							AR
7	$(0,f)$				$(0,d)$	$(0,\neg h)$	$(0,[r](\neg f \wedge \neg d \wedge \neg h))$	from 3
8	$(0,\neg f)$	$(0,\neg d)$	$(0,\neg h)$	$(0,[r](\neg f \wedge d \wedge \neg h))$	Tbl. 4.4	$(0,\bot)$	$(1,\neg f), (1,\neg d), (1,\neg h)$	rule $\Box$ 7w.2
9	from 4	from 4	from 4	from 4		rule $\bot$ 1,7	$(1,\bot)$	rule $\bot$ 2,8
10	$(0,\bot)$	Tbl. 4.3	$(0,\bot)$	$(1,\neg f), (1,d), (1,\neg h)$				
11	rule $\bot$ 7,8		rule $\bot$ 1,8	$(1,\bot)$				
12				rule $\bot$ 2,10				

Tab. 4.2: Proof that `BK` $\models_G$ `holding` $\rightarrow$ [`replace`]$\neg$`holding`. Continues in Tables 4.3 and 4.4.

Tables 4.2 through 4.4 prove that if all the robot initially knows—besides its background knowledge—is that it is holding the can, then it is necessary that it is not holding it after replacing the can.

Line	Γ & Σ				Comment
1	$(0, \neg f)$	$(0, d)$	$(0, \neg h)$	$(0, [r](f \wedge \neg d \wedge \neg h))$	from 4 in Tbl. 4.2
2	$(0, \bot)$	$(0, \bot)$	$(0, \bot)$	$(1, f), (1, \neg d), (1, \neg h)$	
3	rule $\bot$ 1, and	rule $\bot$ 1, and	rule $\bot$ 1, and	$(1, \bot)$	rule $\bot$ 2, and
4	7 in Tbl. 4.2	8 in Tbl. 4.2	1 in Tbl. 4.2		2 in Tbl. 4.2

Tab. 4.3: Proof that $\mathtt{BK} \models_G \mathtt{holding} \rightarrow [\mathtt{replace}]\neg\mathtt{holding}$. Continued from Table 4.2.

Line	Γ & Σ							Comment
1	$(0, \neg f)$				$(0, \neg d)$	$(0, \neg h)$	$(0, [r](f \wedge d \wedge \neg h))$	from 2 in Tbl. 4.2
2	the following is from 3 in Tbl. 4.2:				$(0, \bot)$	$(0, \bot)$	$(1, f), (1, d), (1, \neg h)$	rule $\Diamond$ 1w.
3	$(0, f)$	$(0, \neg d)$	$(0, \neg h)$	$(0, [r](\neg f \wedge d \wedge \neg h))$	rule $\bot$ 1, and	rule $\bot$ 1, and	$(1, \bot)$	2 in Tbl. 4.2
4	$(0, \bot)$	$(0, \bot)$	$(0, \bot)$	$(1, \neg f), (1, d), (1, \neg h)$	7 in Tbl. 4.2	1 in Tbl. 4.2	rule $\bot$ 2, and	
5	rule $\bot$ 1,3	rule $\bot$ 3, and	rule $\bot$ 3, and	$(1, \bot)$			2 in Tbl. 4.2	
6		7 in Tbl. 4.2	1 in Tbl. 4.2	rule $\bot$ 4, and				
				2 in Tbl. 4.2				

Tab. 4.4: Proof that $\mathtt{BK} \models_G \mathtt{holding} \rightarrow [\mathtt{replace}]\neg\mathtt{holding}$. Continued from Table 4.2.

Line	Γ & Σ							Comment
1	$(0, \kappa_1),\ (0, \kappa_2),\ \ldots,\ (0, \kappa_z),\ (0, f),\ (0, \neg d),\ (0, \neg h),\ (0, \neg\neg[g][di]d)$							trunk
2	$(0, f),\ (0, \neg d),\ (0, \neg h),\ (0, [g][di]d)$							rule $\neg$ 1
3	$(0, f \wedge \neg d \wedge \neg h \rightarrow [g]((f \wedge \neg d) \vee (\neg f \wedge \neg d \wedge \neg h)))$							AR
4	$(0, \neg f)$	$(0, d)$	$(0, h)$	$(0, [g]((f \wedge \neg d) \vee (\neg f \wedge \neg d \wedge \neg h))$				from 3
5	$(0, \bot)$	$(0, \bot)$	$(0, \bot)$	$(0, f \wedge \neg d \wedge \neg h \rightarrow \langle g\rangle(f \wedge \neg d \wedge h) \wedge \cdots)$				AR
6	rule $\bot$ 1,4	rule $\bot$ 1,4	rule $\bot$ 1,4	$(0, \neg f)$	$(0, d)$	$(0, h)$	$(0, \langle g\rangle(f \wedge \neg d \wedge h)), \ldots$	from 5
7				$(0, \bot)$	$(0, \bot)$	$(0, \bot)$	$(1, f \wedge \neg d \wedge h),\ 0 \xrightarrow{g} 1$	rule $\Diamond$ 6
8				rule $\bot$ 1,6	rule $\bot$ 1,6	rule $\bot$ 1,6	Tbl. 4.6	

Tab. 4.5: Proof that $\mathtt{BK} \models_G IC \rightarrow \langle\mathtt{grab}\rangle\langle\mathtt{drink}\rangle\neg\mathtt{drank}$. Continues in Table 4.6.

Tables 4.5 and 4.6 prove that it is possible to not have drunk the oil after performing the `drink` action after performing the `grab` action in the initial condition.

Line	Γ & Σ				Comment
1	$(1, f), (1, \neg d), (1, h)$				from 7 in Tbl. 4.5
2	$(1, [di]d)$				rule $\Box$ 2w.
3	$(0, f \wedge \neg d \wedge h \rightarrow \langle di\rangle(\neg f \wedge d \wedge h) \wedge \langle di\rangle(\neg f \wedge \neg d \wedge h)))$				7 in Tbl. 4.5
4	$(0, \neg f)$	$(0, d)$	$(0, \neg h)$	$(1, \langle di\rangle(\neg f \wedge d \wedge h)),\ (1, \langle di\rangle(\neg f \wedge \neg d \wedge h))$	from 3
5	$(0, \bot)$	$(0, \bot)$	$(0, \bot)$	$(2, \neg f \wedge \neg d \wedge h),\ 1 \xrightarrow{di} 2$	rule $\Diamond$ 4
6	rule $\bot$ 1,4	rule $\bot$ 1,4	rule $\bot$ 1,4	$(2, \neg f), (2, \neg d), (2, h)$	rule $\wedge$ 5
7				$(2, d)$	rule $\Box$ 2w.5
8				$(2, \bot)$	rule $\bot$ 6,7

Tab. 4.6: Proof that $\mathtt{BK} \models_G IC \rightarrow \langle\mathtt{grab}\rangle\langle\mathtt{drink}\rangle\neg\mathtt{drank}$. Continued from Table 4.5.

Tables 4.7 and 4.8 prove that it is not possible to perceive that the can has a medium weight after performing the `grab` action in the initial condition and then the `weigh` action. Note that this statement could be due to grabbing or weighing being inexecutable; it is however, simply because `obsMedium` is inperceivable in the world resulting from doing `grab` then `weigh` in the initial world. Due to limited space, Table 4.7 has no Comment column.

Line	Γ & Σ									
1	$(0,\kappa_1),\ (0,\kappa_2),\ \ldots,\ (0,\kappa_z),\ (0,f),\ (0,\neg d),\ (0,\neg h),\ (0,\neg[g][oM \mid w]\bot)$									
2	$(1,\neg[oM \mid w]\bot),\ 0 \xrightarrow{g} 1$									
3	$(2,\neg\bot),\ 1 \xrightarrow{w} 2,\ oM \xrightarrow{w} 2$									
4	$(0, f \wedge \neg d \wedge \neg h \rightarrow [g]((f \wedge \neg d) \vee (\neg f \wedge \neg d \wedge \neg h)))$									
5	$(0,\neg f)$	$(0,d)$	$(0,h)$	$(0,[g]((f \wedge \neg d) \vee (\neg f \wedge \neg d \wedge \neg h)))$						
6	$(0,\bot)$	$(0,\bot)$	$(0,\bot)$	$(1,f),(1,\neg d)$					$(1,\neg f),(1,\neg d),(1,\neg h)$	
7	rule $\bot$ 1,5	rule $\bot$ 1,5	rule $\bot$ 1,5	$(1, f \wedge \neg d \wedge h \rightarrow [w](f \wedge \neg d \wedge h))$					$(1,\langle w\rangle\top \rightarrow h)$	
8				$(1,\neg f)$	$(1,d)$	$(1,\neg h)$		$(1,[w](f \wedge \neg d \wedge h))$	$(1,\neg\langle w\rangle\top)$	$(1,h)$
9				$(1,\bot)$	$(1,\bot)$	$(1,\langle w\rangle\top \rightarrow h)$		$(2, f \wedge \neg d \wedge h)$	$(2,\bot)$	$(1,\bot)$
10						$(1,\neg\langle w\rangle\top)$	$(1,h)$	$(2,f),(2,\neg d),(2,h)$		
11						$(2,\bot)$	$(1,\bot)$	Tbl. 4.8		

Tab. 4.7: Proof that $\mathtt{BK} \models_G IC \rightarrow \neg\langle\mathtt{grab}\rangle\langle\mathtt{obsMedium} \mid \mathtt{weigh}\rangle\top$. Continued in Table 4.8.

Line	Γ & Σ			Comment
1	$(1,(\forall v^o)\ v^o = oH \leftrightarrow \langle v^o \mid w\rangle(f \wedge \neg d \wedge h))$			from PR
2	$(1, oN = oH \leftrightarrow \langle oN \mid w\rangle(f \wedge \neg d \wedge h) \wedge$			
	$oL = oH \leftrightarrow \langle oL \mid w\rangle(f \wedge \neg d \wedge h) \wedge$			expanding 1
	$oM = oH \leftrightarrow \langle oM \mid w\rangle(f \wedge \neg d \wedge h) \wedge$			
	$oH = oH \leftrightarrow \langle oH \mid w\rangle(f \wedge \neg d \wedge h))$			
3	$(1,\bot \rightarrow \langle oM \mid w\rangle(f \wedge \neg d \wedge h)),\ (1,\langle oM \mid w\rangle(f \wedge \neg d \wedge h) \rightarrow \bot)$			simplifying 2
4	$(1,\neg\langle oM \mid w\rangle(f \wedge \neg d \wedge h))$			simplifying 3
5	$(2,\neg(f \wedge \neg d \wedge h))$			rule $\Box$ 4w.3 in Tbl. 4.7
6	$(2,\neg f)$	$(2,d)$	$(2,\neg h)$	rule $\vee$ 5
7	$(2,\bot)$	$(2,\bot)$	$(2,\bot)$	rule $\bot$ 6, and 10 in Tbl. 4.7

Tab. 4.8: Proof that $\mathtt{BK} \models_G IC \rightarrow \neg\langle\mathtt{grab}\rangle\langle\mathtt{obsMedium} \mid \mathtt{weigh}\rangle\top$. Continues from Table 4.7.

4.7 Concluding Remarks

We have presented the definition of a modal logic involving actions and observations. A procedure for deciding whether entailments hold was provided, and the procedure was shown to be sound, complete and terminating. The logic is thus decidable on the question of entailment.

In the previous version of LAO [Rens et al., 2010], a modal operator $[\alpha]$ was part of the language. In this version, we have abbreviated it using quantification and modal operator $[\varsigma \mid \alpha]$. The meaning of $[\alpha]$ is the same in both versions, however, by taking it out of the language here, proofs of soundness and completeness are simpler than before. The main reason for defining $[\alpha]$ as an abbreviation is to obviate any issues of interaction with sentences involving $[\varsigma \mid \alpha]$. Tableau rules $\langle\alpha\rangle$ and $[\alpha]$ of the previous version [Rens et al., 2010] are thus also removed here.

The previous version of rule $\Diamond$ [Rens et al., 2010] may cause the procedure not to terminate for some particular kinds of sentences. For instance, suppose $\mathcal{K} = \{\neg[\varsigma_1 \mid \alpha_1]f_1\}$ and suppose we have the query $\mathcal{K} \models_G f_2$. The trunk of a tableaux tree for the query is $\langle\{(0,\neg[\varsigma_1 \mid \alpha_1]f_1 \wedge \neg f_2)\},\emptyset\rangle$. Then one sequence of (nondeterministic) rule applications is as follows.

rule $\wedge$: $\Delta = \{(0,\neg[\varsigma_1 \mid \alpha_1]f_1),\ (0,\neg f_2)\}$ and $\Sigma = \emptyset$ are in the new node.

rule $\Diamond$: $\Delta = \{(1,\neg f_1),\ (0,\neg f_2),\ (1,\neg[\varsigma_1 \mid \alpha_1]f_1)\}$ and $\Sigma = \{0 \xrightarrow{\alpha_1} 1,\ \varsigma_1 \xrightarrow{\alpha_1} 1\}$ are in the new node.

rule $\Diamond$: $\Delta = \{(2, \neg f_1),\ (1, \neg f_1),\ (0, \neg f_2),\ (1, \neg[\varsigma_1 \mid \alpha_1]f_1),\ (2, \neg[\varsigma_1 \mid \alpha_1]f_1)\}$ and $\Sigma = \{1 \xrightarrow{\alpha_1} 2,\ \varsigma_1 \xrightarrow{\alpha_1} 2,\ 0 \xrightarrow{\alpha_1} 1,\ \varsigma_1 \xrightarrow{\alpha_1} 1\}$ are in the new node.

rule $\Diamond$: $\cdots$

The new version of rule $\Diamond$ ensures that the rule is not infinitely applicable.

It may be sufficient to represent an agent's 'belief-state' with a certain set of propositions, for example, the robot does not require `heavy`, `medium` and `light` to be propositions. Computational complexity in logics with possible world semantics is usually affected by the number of possible worlds being considered. Because the number of possible worlds representable increases exponentially with the number of propositions in the vocabulary, it would be advisable to minimize the size of the vocabulary. For instance, if the robot's vocabulary were {`full`, `drank`, `holding`, `heavy`, `medium`, `light`}, it would have to consider $2^6 = 64$ worlds instead of $2^3 = 8$. We were thus tempted to cite this as another motivation to separate observations and propositions as different sorts of objects. Unfortunately, instead of having $|\mathcal{A}|$ $[\alpha]$ operators, we have $|\mathcal{A}| \times |\Omega|$ operators of the form $[\varsigma \mid \alpha]$. Exactly what the implication of the larger number of $[\varsigma \mid \alpha]$ operators is for computational complexity, must still be determined.

Although laws can be captured with global axioms, eliminating the need for $\Box$ to express laws, as in our logic LAO, it is not obvious how *immutable propositions* [Castilho et al., 1999] and goals for planning can be specified without $\Box$ and $\Diamond$. This deficiency is addressed to some degree in our logics SLAP and SLAOP by the addition of a $\Box$ (necessity) operator. More about this, with respect to our other logics, is discussed in later chapters.

A relatively straightforward approach to domain specification with LAO was presented. We could arguably have presented a more economical approach, for instance, focusing on each fluent in turn, and specifying how it changes due to the different actions. This is the Reiter-style [Reiter, 1991], which also goes towards solving the frame problem in the situation calculus. The approach we presented, however, is an almost direct translation of transition diagrams, which are quite easy to understand. We thus preferred this approach as an introduction of domain specification in this thesis. Furthermore, solving the frame problem for LAO is not important for this thesis; LAO is defined as a 'stepping-stone' for defining more expressive logics, for which the frame problem *will* be addressed.

If α^{ont} is an ontic action, then presently, one of the perceivability axioms is

$$\langle \texttt{obsNil} \mid \alpha \rangle \varphi_1 \wedge \langle \texttt{obsNil} \mid \alpha \rangle \varphi_2 \wedge \cdots \langle \texttt{obsNil} \mid \alpha \rangle \varphi_m$$

where $\varphi_1, \varphi_2, \ldots, \varphi_m$ identify all the reachable worlds via α^{ont} (see Axiom 4.7). This axiom could be very large—in the worst case, it could have $2^{|\mathcal{F}|}$ conjuncts. With the LAO syntax as it is, we cannot see how to express (4.7) more compactly. Essentially, what we are trying to say is that for all worlds reachable via an ontic action, only $obsNil$ is perceivable in those worlds. That is,

$$\text{for all } w, w' \in W, \text{ for all } \varsigma \in \Omega, \text{ if } (w, w') \in R_{\alpha^{ont}}, \text{ then } (\varsigma, w') \in Q_{\alpha^{ont}} \text{ s.t. } \varsigma = obsNil.$$

We leave this problem for future work.

LAO is a formalism for reasoning about actions based on multi-modal logic which allows for

expressing observations as first-class objects. We introduced a new modal operator, namely $[\varsigma \mid \alpha]$, which allows us to capture the notion of perceiving an observation given that an action has taken place. Formulae of the type $[\varsigma \mid \alpha]\varphi$ mean 'after perceiving observation ς, given α was performed, necessarily φ'. This work focuses on the challenges concerning sensing with explicit observations, and acting with nondeterministic effects. We presented a correct and decidable tableau method for the logic.

Uncertainty is dealt with in LAO using nondeterminism in actions and observations. This is a rather course grained method. In the rest of this thesis, we develop logics which deal with uncertainty using probabilities. The next chapter introduces our approach to expressing stochastic uncertainty via a logic with probabilistic action effects—stochastic observations are added later in the thesis.

5. THE SPECIFICATION LOGIC OF ACTIONS WITH PROBABILITY

The first three sections of this chapter are more or less a reproduction of an article published in the Journal of Applied Logic [Rens et al., 2014a]. Preliminary research about the work appearing in this chapter concerning domain specification was presented at the Symposium on Logical Formalizations of Commonsense Reasoning in Ayia Napa, Cyprus [Rens et al., 2013].

The previous two chapters presented the Logic of Actions and Observations (LAO). In this chapter, we present a logic for specifying agents' stochastic action models, or more generally, for specifying probabilistic transition systems—the Specification Logic of Actions with Probability (SLAP). Our logic takes the possible worlds semantics of modal logic and draws inspiration from Markov decision process (MDP) theory to deal with probabilities. MDP theory [Bellman, 1957, Howard, 1960, Puterman, 1994] has proven to be a good general framework for formalizing dynamic stochastic systems with complete state observability.

LAO is more expressive in the sense that one can reason about *sequences* of actions and observations. SLAP does not have a notion of observation and one cannot reason about sequences of actions in SLAP. However, SLAP is more expressive than LAO in the sense that the uncertainty in action transitions can be expressed much more finely.

Another important difference between LAO and SLAP is that entailment in LAO is defined via global semantic consequence, whereas entailment in SLAP is defined via local semantic consequence. The reason for changing from global to local semantic consequence is because at the time of developing LAO, global entailment seemed appropriate, and to produce a 'cleaner' syntax. However, while developing SLAP, local entailment seemed more appropriate: In SLAP, a 'box' modal operator is added to the language to mark axioms as *globally* applicable. Being able to *mark* sentences as globally applicable puts more control in the hands of the knowledge engineer. However, with a box operator defined, *local* consequence is appropriate. Technically though, global semantic consequence could have been defined for SLAP too. Because SLAOP and SDL build directly on SLAP, they also employ local consequence.

The oil-drinking scenario—introduced in Chapter 4—is partially formalized as follows. The robot has the set of (intended) actions $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{replace}\}$ with expected meanings. Note that there is no set of observations, and having a weighing action would thus not make sense. The robot experiences its world (domain) through three Boolean features: $\mathcal{F} = \{\texttt{full}, \texttt{drank}, \texttt{holding}\}$ meaning respectively that the oil-can is full, that the robot has drunk the oil and that it is currently holding something in its gripper. Given a formalization $\mathcal{K}$ of our scenario, the robot may have the following queries:

- If the oil-can is full, I have not drunk the contents and I am holding the can, is there a 0.15 probability that after 'drinking' the contents, the oil-can is still full, I have still not

drunk the oil and I am still holding the can? That is, does $(\texttt{full} \wedge \neg\texttt{drank} \wedge \texttt{holding}) \rightarrow [\texttt{drink}]_{0.15}(\texttt{full} \wedge \neg\texttt{drank} \wedge \texttt{holding})$ follow from $\mathcal{K}$?

- If the oil-can is empty and I'm not holding it, is there a 0.9 probability that I'll be holding it after grabbing it, and a 0.1 probability that I'll have missed it? That is, does $(\neg\texttt{full} \wedge \neg\texttt{holding}) \rightarrow ([\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \texttt{holding}) \wedge [\texttt{grab}]_{0.1}(\neg\texttt{full} \wedge \neg\texttt{holding}))$ follow from $\mathcal{K}$?

In SLAP we are interested in whether $IC \rightarrow \Phi$ follows from $\bigwedge_{\kappa\in\mathcal{K}} \Box\kappa$, where Φ is any 'legal' sentence of interest, IC is the initial condition and $\Box$ marks sentences as laws of the domain, that is, sentences which must be true in every possible world.

In the second half of this chapter we show how SLAP can be used for specifying probabilistic transition models; we especially investigate strategies for providing smaller 'full' specifications. The work in this chapter also constitutes our solution to the frame problem in the context of probabilistic transitions or stochastic actions.

Many environments can be modeled as probabilistic transition systems. For instance, a robot which is uncertain about the outcomes of its actions could rely on such a model. Or to simulate some biological process may require a model of how likely it is that a particular state of the process will arise, given some (cellular/molecular/chemical) event occurs in another process state. Usually, a full specification of transition probabilities is required so that the likelihood of the system changing from one current state s_c to a resulting state s_r can be deduced, for *all* system states.

There are naïve ways of specifying a system's dynamics and there are more sophisticated ways which attempt to make the task of specification easier and the specifications more compact, by making use of regularities and common sense. Intuitive lines of reasoning are followed, relying on two kinds of default assumptions when transition information is deficient. Transition information may be unobtainable or difficult to deduce, or the knowledge engineer may know that the default assumption is correct for a given domain and thus knows that s/he needs not (re)state the information.

For us, it thus makes sense to tackle these issues using SLAP. However, the research discussed in this chapter may well be applied to other logics with probabilistic transition semantics.

Section 5.1 defines SLAP. Section 5.2 provides a decision procedure for determining entailment of sentences in SLAP. In Section 5.3, we prove that the procedure is sound, complete and that it terminates, that is, we show that SLAP is decidable with respect to entailment. In Section 5.4, we introduce the basic approach to specifying a domain. Section 5.5 investigates how invariance of features of the world under certain conditions can be captured. The domain specification approach is then extended employing the new insights. Section 5.6 tackles the issue of how to complete underspecified specifications. In Section 5.7, we prove that our two approaches for completing specifications lead to 'full' specifications.

5.1 Defining the Logic

First, the syntax of the logic is presented, then its semantics.

5.1.1 Syntax

The vocabulary of our language contains three sorts of objects of interest:

1. a finite set of *fluents* (alias, *propositional atoms*) $\mathcal{F} = \{f_1, \ldots, f_n\}$,
2. a finite set of names of atomic *actions* $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$,
3. all *rational numbers* $\mathbb{Q}$.

From now on, we denote $\mathbb{Q} \cap [0, 1]$ as $\mathbb{Q}_{[0,1]}$. We are going to work in a multi-modal setting, in which we have modal operators $[\alpha]_q$, one for each $\alpha \in \mathcal{A}$ and $q \in \mathbb{Q}_{[0,1]}$.

Definition 5.1.1: Let $\alpha \in \mathcal{A}$, $q \in \mathbb{Q}_{[0,1]}$ and $f \in \mathcal{F}$. The language of SLAP, denoted $\mathcal{L}_{SLAP}$, is the least set of Ψ defined by the grammar:[1]

$$\begin{aligned}\varphi &::= f \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi.\\ \Phi &::= \varphi \mid \neg\Phi \mid \Phi \wedge \Phi \mid [\alpha]_q\varphi.\\ \Psi &::= \Phi \mid \Box\Phi \mid \Psi \wedge \Psi.\end{aligned}$$

We shall also require the definition of $\mathcal{L}^{-\Box}_{SLAP}$, the least set of Φ as defined above.

In SLAP, sentences of the form $\neg\Box\Phi$ are not in the language. The reason is that the decision procedure for SLAP entailment would not notice certain contradictions which may occur due to such sentences being allowed. Note that formulae with nested modal operators of the form $\Box\Box\Phi$, $\Box\Box\Box\Phi$, etc. or of the form $[\alpha]_q[\alpha]_q\varphi$, $[\alpha]_q[\alpha]_q[\alpha]_q\varphi$, etc. are not in $\mathcal{L}_{SLAP}$. 'Single-step' or 'flat' formulae are sufficient to *specify* action transitions and transition probabilities. As usual, we treat $\bot, \vee, \rightarrow$ and $\leftrightarrow$ as abbreviations. $\rightarrow$ and $\leftrightarrow$ have the weakest bindings and $\neg$ the strongest; parentheses enforce or clarify the scope of operators conventionally.

Two distinguished schemata are $[\alpha]_q\varphi$ and $\neg[\alpha]_q\varphi$ and shall be referred to as *dynamic literals*. Any formula which includes a dynamic literal shall be referred to as *dynamic*. $[\alpha]_q\varphi$ is read 'The probability of reaching a world in which φ holds after executing α, is equal to q'.[2] $[\alpha]$ abbreviates $[\alpha]_1$. $\langle\alpha\rangle\varphi$ abbreviates $\neg[\alpha]_0\varphi$ and is read 'It is possible to reach a world in which φ holds after executing α'. Note that $\langle\alpha\rangle\varphi$ does not mean $\neg[\alpha]\neg\varphi$. One reads $\Box\Phi$ as 'Φ holds in every possible world'. We require the $\Box$ operator to mark certain information (sentences) as holding in *all* possible worlds—essentially, the axioms which model the domain of interest.

Definition 5.1.2: A formula $\Psi \in \mathcal{L}_{SLAP}$ is in *conjunctive normal form* (CNF) if and only if it is in the form

$$\Psi_1 \wedge \Psi_2 \wedge \cdots \wedge \Psi_n,$$

where each of the Ψ_i is a disjunction of literals, whether dynamic or propositional. The Ψ_is of a formula in CNF are called *clauses*.

A formula $\Psi \in \mathcal{L}_{SLAP}$ is in *disjunctive normal form* (DNF) if and only if it is in the form

$$\Psi_1 \vee \Psi_2 \vee \cdots \vee \Psi_n,$$

[1] In one of our publications [Rens et al., 2013], we erroneously omitted $\Psi \wedge \Psi$ from the definition of Ψ.

[2] To put the reader's mind at ease, the final logic defined in this thesis (SDL) can express boundary probabilities, for instance, $[\alpha]\varphi \geq p$ (see Def. 7.1.1).

where each of the Ψ_i is a conjunction of literals, whether dynamic or propositional. The Ψ_is of a formula in DNF are called *terms*.

Note that if a dynamic literal $[\alpha]_q\varphi$ or $\neg[\alpha]_q\varphi$ is a disjunct/conjunct of Ψ_i, φ is allowed to have any form, as long as $\varphi \in \mathcal{L}_{SLAP}$.

5.1.2 Semantics

SLAP structures are derived from Markov decision processes (MDPs) [Bellman, 1957, Howard, 1960, Puterman, 1994]. An MDP model is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, s^0 \rangle$: $\mathcal{S}$ is a finite set of states the agent can be in; $\mathcal{A}$ is a finite set of actions the agent can choose to execute; $\mathcal{T}$ is the function defining the probability of reaching one state from another, for each action; $\mathcal{R}$ is a function, giving the expected immediate reward gained by the agent, for any state and agent action; and s^0 is the initial state in $\mathcal{S}$. However, rewards are not modeled in SLAP structures.

Just like LAO structures, SLAP structures are not Kripke-style models: Its semantics has a structure of the form $\langle W, R \rangle$, where W is a *finite* set of worlds such that each world assigns a truth-value to each atomic proposition, and R is a binary relation on W. Moreover, SLAP is multi-modal in that there are multiple accessibility relations.

Intuitively, when talking about some world w, we mean a set of features (*propositions*) that the agent understands and that describes a state of affairs in the world or that describes a possible, alternative world. Let $w : \mathcal{F} \mapsto \{0, 1\}$ be a total function that assigns a truth-value to each proposition. Let C be the set of all possible functions w. We call C the *conceivable worlds*.

Definition 5.1.3: A SLAP structure is a tuple $\mathcal{S} = \langle W, R \rangle$ such that

1. $W \subseteq C$ a non-empty set of *possible worlds*.
2. $R : \mathcal{A} \mapsto R_\alpha$, where $R_\alpha : (W \times W) \mapsto \mathbb{Q}_{[0,1]}$ is a total function from pairs of worlds into the rationals; That is, R is a mapping that provides an accessibility relation R_α for each action $\alpha \in \mathcal{A}$; For every $w^- \in W$, it is required that either $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 1$ or $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 0$.

Note that the set of possible worlds may be the whole set of conceivable worlds.

R_α defines the transition probability $pr \in \mathbb{Q}_{[0,1]}$ between worlds w^+ and w^- via action α. If $(w^-, w^+, 0) \in R_\alpha$, then w^+ is said to be *inaccessible* or *not reachable* via α performed in w^-, else if $(w^-, w^+, pr) \in R_\alpha$ for $pr \in (0, 1]$, then w^+ is said to be *accessible* or *reachable* via action α performed in w^-. If for some w^-, $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 0$, we say that α is *inexecutable* in w^-.

Figure 5.1 is a pictorial representation of transitions and their probabilities for the action `grab` of the oil-can scenario. The eight circles represent the eight conceivable worlds with their valuations.

Definition 5.1.4 (Truth Conditions): Let $\mathcal{S}$ be an SLAP structure, with $\alpha, \alpha' \in \mathcal{A}$ and $q, pr \in \mathbb{Q}_{[0,1]}$. Let $f \in \mathcal{F}$ and let Ψ and φ be sentence in $\mathcal{L}_{SLAP}$. We say Ψ is *satisfied* at world w in structure $\mathcal{S}$ (written $\mathcal{S}, w \models \Psi$) if and only if the following hold:

1. $\mathcal{S}, w \models \top$ for all $w \in W$;

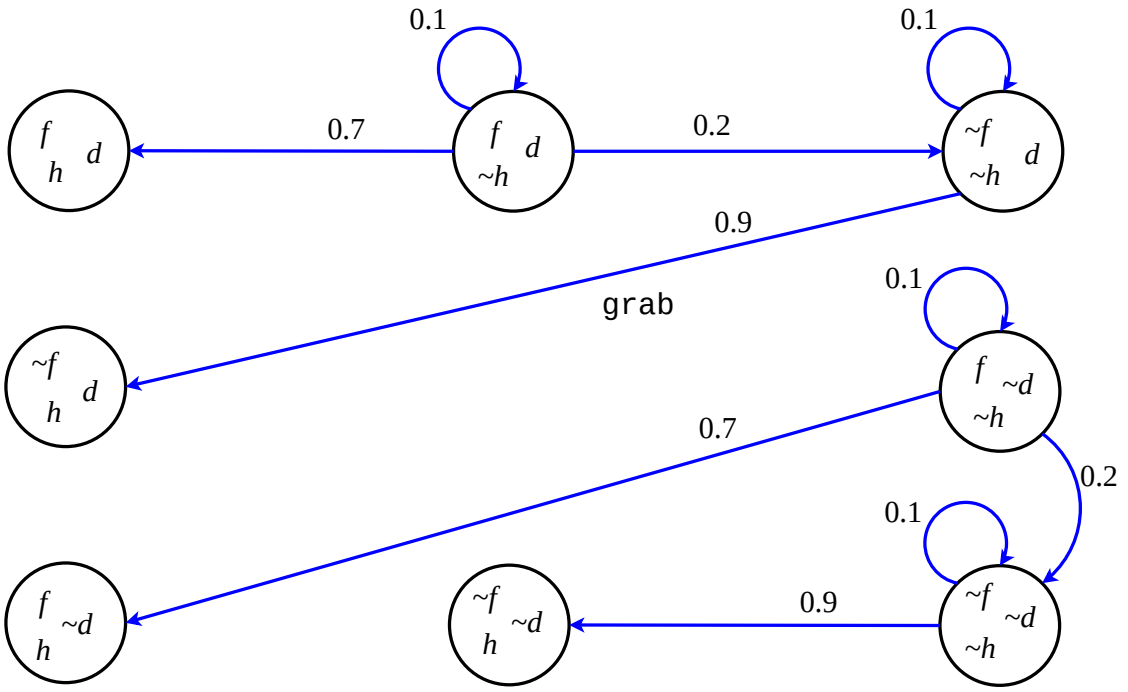


Fig. 5.1: A transition diagram for the `grab` action. The letters f, d and h, respectively represent propositional literals `full`, `drank` and `holding`. And $\sim$ reads 'not'.

2. $\mathcal{S}, w \models f \iff w(f) = 1$ for $w \in W$;
3. $\mathcal{S}, w \models \neg\Psi \iff \mathcal{S}, w \not\models \Psi$;
4. $\mathcal{S}, w \models \Psi \wedge \Psi' \iff \mathcal{S}, w \models \Psi$ and $\mathcal{S}, w \models \Psi'$;
5. $\mathcal{S}, w \models [\alpha]_q\varphi \iff \left(\sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \varphi} pr\right) = q$;
6. $\mathcal{S}, w \models \Box\Psi \iff$ for all $w' \in W, \mathcal{S}, w' \models \Psi$.

Looking at Figure 5.1, for instance, if the robot is in a situation where the oil-can is full, the oil has not been drunk and the can is not being held, then the probability that the oil-can is still full after grabbing the can is 0.7 + 0.1 = 0.8. Thus, in the syntax of SLAP, given a formalization BK of the scenario, $(\texttt{full} \wedge \neg\texttt{drank} \wedge \neg\texttt{holding}) \rightarrow [\texttt{grab}]_{0.8}\texttt{full}$ follows from BK.

A formula Ψ is *valid* in an SLAP structure (denoted $\mathcal{S} \models \Psi$) if $\mathcal{S}, w \models \Psi$ for every $w \in W$. Ψ is *SLAP-valid* (denoted $\models \Psi$) if Ψ is true in every structure $\mathcal{S}$. If $\models \theta \leftrightarrow \psi$, we say θ and ψ are *semantically equivalent* (abbreviated $\theta \equiv \psi$).

Ψ is *satisfiable* if $\mathcal{S}, w \models \Psi$ for some $\mathcal{S}$ and $w \in W$. A formula that is not satisfiable is *unsatisfiable* or a *contradiction*. The truth of a propositional formula depends only on the world in which it is evaluated. We may thus write $w \models \Psi$ instead of $\mathcal{S}, w \models \Psi$ when Ψ is a propositional formula.

Let $\mathcal{K} \subseteq \mathcal{L}_{SLAP}$ and $\Phi \in \mathcal{L}_{SLAP}$. We say that Φ is a *local semantic consequence* of $\mathcal{K}$ (denoted $\mathcal{K} \models \Phi$) if for all structures $\mathcal{S}$ and all $w \in W$ of $\mathcal{S}$, if for all $\kappa \in \mathcal{K}$, $\mathcal{S}, w \models \kappa$, then $\mathcal{S}, w \models \Phi$. We also say that $\mathcal{K}$ *entails* Φ whenever $\mathcal{K} \models \Phi$.

Recall that $\mathcal{L}^{-\Box}_{SLAP}$ is all formulae in $\mathcal{L}_{SLAP}$ such that the formulae contain no $\Box$ operators.

Proposition 5.1.1: For every $\Psi \in \mathcal{L}^{-\Box}_{SLAP}$, there exists a formula $\Psi' \in \mathcal{L}^{-\Box}_{SLAP}$ in CNF and there exists a formula $\Psi'' \in \mathcal{L}^{-\Box}_{SLAP}$ in DNF such that $\Psi \equiv \Psi' \equiv \Psi''$.

The proof is straightforward, appealing to basic logical equivalences.

5.1.3 Reducing Entailment to Unsatisfiability

Let $\mathcal{K}$ be a finite subset of $\mathcal{L}_{SLAP}$ and let Φ be an element of $\mathcal{L}^{-\Box}_{SLAP}$.

Proposition 5.1.2: $\mathcal{K} \models \Phi \iff \bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \neg\Phi$ is unsatisfiable.

Proof:
$\mathcal{K} \models \Phi$
$\iff \forall \mathcal{S}, w,$ if $\mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa,$ then $\mathcal{S}, w \models \Phi$
$\iff \forall \mathcal{S}, w,\ \mathcal{S}, w \not\models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ or $\mathcal{S}, w \models \Phi$
$\iff \nexists \mathcal{S}, w$ s.t. $\mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and $\mathcal{S}, w \not\models \Phi$
$\iff \nexists \mathcal{S}, w$ s.t. $\mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and $\mathcal{S}, w \models \neg\Phi$
$\iff \bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \neg\Phi$ is unsatisfiable. ■

Note that $\neg\Box\Phi' \notin \mathcal{L}_{SLAP}$. This is why Φ in Proposition 5.1.2 is restricted to be in $\mathcal{L}^{-\Box}_{SLAP}$. The restriction is not a problem when $\Box$ is used only to define domain axioms or laws, which are always in the knowledge-base or an agent's background knowledge BK (here represented by $\mathcal{K}$) and not in Φ. The decision procedure for entailment in SLAP is thus based on Proposition 5.1.2 (with restricted arguments).

In the introduction, we mentioned that we are interested in whether $IC \rightarrow \Phi$ follows from $\bigwedge_{\kappa \in \mathcal{K}} \Box\kappa$, in general. In practical terms, this is stated as

$$\{\Box\beta \mid \beta \in BK\} \models IC \rightarrow \Phi,$$

where $\Phi \in \mathcal{L}^{-\Box}_{SLAP}$ is any sentence of interest, $IC \in \mathcal{L}^{-\Box}_{SLAP}$ is a sentence describing an agent's initial condition and $BK \subset \mathcal{L}^{-\Box}_{SLAP}$ is the agent's background knowledge.

A complete specification of the probabilistic effects of action `grab` (Fig. 5.1), for instance, is

$$\begin{aligned}
\texttt{full} \wedge \texttt{drank} \wedge \neg\texttt{holding} \ &\rightarrow\ [\texttt{grab}]_{0.7}(\texttt{full} \wedge \texttt{drank} \wedge \texttt{holding}) \wedge \\
&\quad\ [\texttt{grab}]_{0.3}(\texttt{drank} \wedge \neg\texttt{holding}); \\
\texttt{full} \wedge \neg\texttt{drank} \wedge \neg\texttt{holding} \ &\rightarrow\ [\texttt{grab}]_{0.7}(\texttt{full} \wedge \neg\texttt{drank} \wedge \texttt{holding}) \wedge \\
&\quad\ [\texttt{grab}]_{0.3}(\neg\texttt{drank} \wedge \neg\texttt{holding}); \\
\neg\texttt{full} \wedge \texttt{drank} \wedge \neg\texttt{holding} \ &\rightarrow\ [\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \texttt{drank} \wedge \texttt{holding}) \wedge \\
&\quad\ [\texttt{grab}]_{0.1}(\neg\texttt{full} \wedge \texttt{drank} \wedge \neg\texttt{holding}); \\
\neg\texttt{full} \wedge \neg\texttt{drank} \wedge \neg\texttt{holding} \ &\rightarrow\ [\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \neg\texttt{drank} \wedge \texttt{holding}) \wedge \\
&\quad\ [\texttt{grab}]_{0.1}(\neg\texttt{full} \wedge \neg\texttt{drank} \wedge \neg\texttt{holding}).
\end{aligned}$$

The above sentences would appear in BK. IC could, for example, be $\texttt{full} \wedge \neg\texttt{drank} \wedge \texttt{holding}$ and Φ could, for example, be $[\texttt{drink}]_{0.15}(\texttt{full} \wedge \neg\texttt{drank} \wedge \texttt{holding})$.

In terms of the decision procedure, the question of whether

$$\{\Box\beta \mid \beta \in BK\} \models IC \rightarrow \Phi$$

is posed as the question of whether

$$\bigwedge_{\beta \in BK} \Box\beta \wedge \neg(IC \rightarrow \Phi)$$

is (un)satisfiable.

In the soundness and completeness proofs (§ 5.3.1 and § 5.3.2), even though formulae of the form $\neg\Phi$ are analyzed, $\neg\Phi$ always has the form $\bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \neg\Phi'$, where Φ' does not mention the $\Box$ operator.

5.2 Decision Procedure for Semantic Consequence

In this section we describe a proof procedure which has two phases: creation of a tableau tree (the *tableau* phase) and then seeking solutions for a linear system of inequalities (equations and disequations; the *SLI* phase). The tableau method we propose is adapted from Castilho et al. [1999]. The basic approach is similar, but it has been extensively modified to suit our needs. In the SLI phase, solutions are sought for systems of inequalities generated from formulae involving dynamic literals appearing in a particular form in the open branches of the tree created in the tableau phase. Depending on the results, certain branches may become closed. Depending on the final structure and contents of the tree, the sentence for which the tree was created can be determined as valid or not.

To clarify the reasoning behind the processes of the two phases, we introduce an example and apply the processes of each phase to it. The example will not illustrate how every kind of scenario which can occur is dealt with, but it should give a flavour for how the procedure works.

5.2.1 The Tableau Phase

The tableau rules for SLAP follow. Let Γ^j_k be a leaf node.

1. rule $\bot$: If Γ^j_k contains (x, Ψ) and $(x, \neg\Psi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \bot)\}$.
2. rule $\neg$: If Γ^j_k contains $(x, \neg\neg\Psi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Psi)\}$.
3. rule $\wedge$: If Γ^j_k contains $(x, \Psi \wedge \Psi')$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Psi), (x, \Psi')\}$.
4. rule $\vee$: If Γ^j_k contains $(x, \neg(\Psi \wedge \Psi'))$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \neg\Psi)\}$ and node $\Gamma^{j'}_0 = \Gamma^j_k \cup \{(x, \neg\Psi')\}$, where j' is a fresh integer.
5. rule $\Diamond\varphi$: If Γ^j_k contains $(0, \neg[\alpha]_0\varphi)$ or $(0, [\alpha]_q\varphi)$ for $q > 0$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \varphi)\}$, where x is a fresh integer.
6. rule $\Box$: If Γ^j_k contains $(0, \Box\Phi)$ and (x, Φ') for any $x \geq 0$, and if it does not yet contain (x, Φ), then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Phi)\}$.

Example

Suppose that

$$\Box(\neg\texttt{full} \wedge \neg\texttt{holding} \rightarrow [\texttt{grab}]_{0.9}\texttt{holding} \wedge [\texttt{grab}]_{0.1}\neg\texttt{holding})$$

and

$$\Box(\neg\texttt{full} \wedge \neg\texttt{holding} \rightarrow [\texttt{grab}]\neg\texttt{full})$$

are two domain axioms in our oil-drinking robot's background knowledge. Figures 5.2 and 5.3 depict a tableau tree for (partially) deciding whether the robot's background knowledge entails

$$\neg\texttt{full} \wedge \texttt{drank} \wedge \neg\texttt{holding} \rightarrow [\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \texttt{holding}).$$

That is, we want to determine whether the probability of being in a situation where the oil-can is not full while being held is 0.9 after grabbing the can in a situation where it is not full, it is on the floor and the oil has already been drunk—given the axioms about how the domain works. `full`, `drank`, `holding` and `grab` are respectively abbreviated as f, d, h and g.

Deciding whether

$$\{\Box(\neg f \wedge \neg h \rightarrow ([g]_{0.9}h \wedge [g]_{0.1}\neg h)), \Box(\neg f \wedge \neg h \rightarrow [g]\neg f)\}$$
$$\models$$
$$\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h)$$

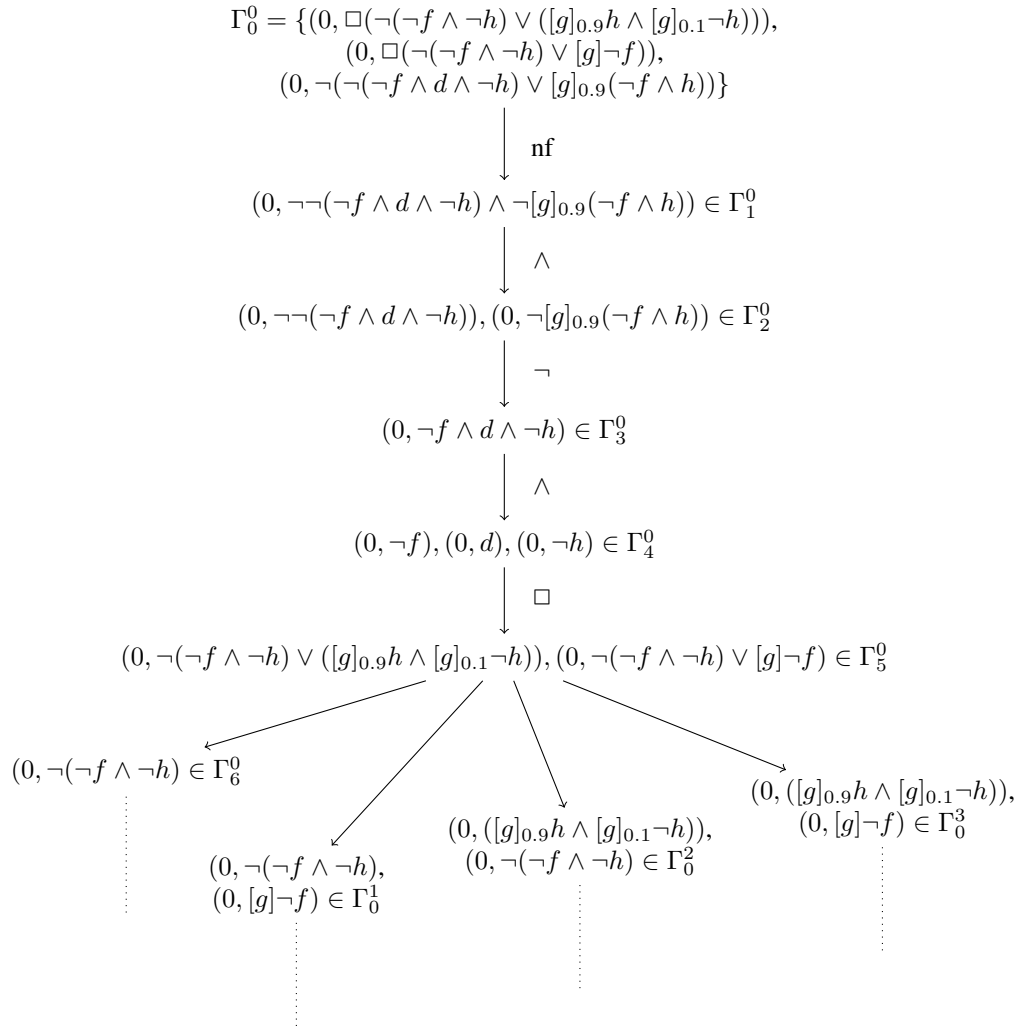
holds is equivalent to determining whether the tree for

$$\begin{aligned}&\Box(\neg f \wedge \neg h \rightarrow ([g]_{0.9}h \wedge [g]_{0.1}\neg h))\wedge\\ &\Box(\neg f \wedge \neg h \rightarrow [g]\neg f)\wedge\\ &\neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h))\end{aligned}$$

closes. (The trunk in the figure is written in a slightly more convenient form.)

The vertices represent nodes and the arcs represent the application of tableau rules. Arcs are labeled with the rule they represent, except when branching occurs, in which case, obviously the $\vee$ rule was applied. In Figures 5.2 and 5.3, it is shown how the vertices relate to the corresponding node. The reader should keep in mind that the node corresponding to a vertex v contains all the labeled formulae in vertices above v on the same branch (v's ancestors)—the vertices show only the elements of nodes which are 'added' to a node due to the application of some rule. An exception is the top vertex of a tree, which is the trunk and not the result of any rule application. In order to show the development of the tree, some liberties were taken with respect to rule application: In some cases, rule application is not shown, that is, from parent node to child node, a formula may be 'processed' more than is possible by the application of the rule represented by the arc from parent to child in the figure.

The arc labeled "nf" denotes *normal forming*: $\neg(\neg(\neg f \wedge d \wedge \neg h) \vee [g]_{0.9}(\neg f \wedge h))$ is an abbreviation for $\neg\neg(\neg f \wedge d \wedge \neg h) \wedge \neg[g]_{0.9}(\neg f \wedge h)$.



continues below

Fig. 5.2: First part of a tableau tree for $\Box(\neg f \wedge \neg h \rightarrow ([g]_{0.9} h \wedge [g]_{0.1} \neg h)) \wedge \Box(\neg f \wedge \neg h \rightarrow [g]\neg f) \wedge \neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h))$.

On the left-hand side of the Figure 5.3, it seems that three nodes 'share' two children. This is only to fit all the information in the diagram. Actually, each of nodes Γ_6^0, Γ_0^1 and Γ_0^2 branches to two of its own children, and these children share some of the same formulae with their 'cousins' as indicated. In other words, strictly speaking, there should be six branches instead of two.

Determining whether the tree finally closes will be seen in the Example section after the SLI phase is described (§ 5.2.3). Each open leaf node will be involved in the SLI phase for a final decision.

5.2.2 Systems of Linear Inequalities

Suppose a tree has been 'grown' till saturation and it has some open branches. It might be the case that there are formulae in the leaf nodes of some open branches, specifying transition probabilities of some action, which are mutually unsatisfiable.

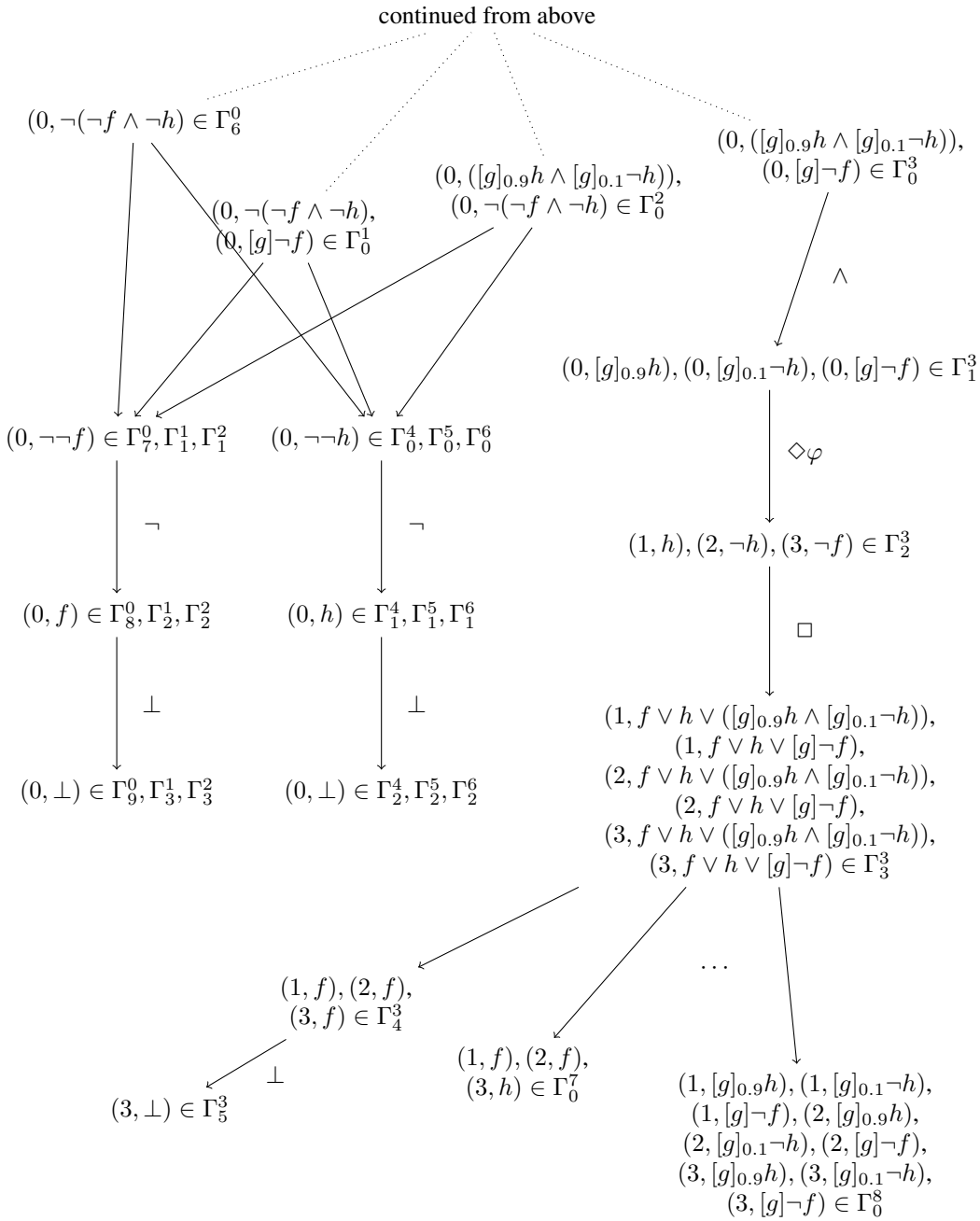


Fig. 5.3: Last part of the tableau tree for $\Box(\neg f \wedge \neg h \to ([g]_{0.9}h \wedge [g]_{0.1}\neg h)) \wedge \Box(\neg f \wedge \neg h \to [g]\neg f) \wedge \neg(\neg f \wedge d \wedge \neg h \to [g]_{0.9}(\neg f \wedge h))$.

For instance, the tree for $[\texttt{grab}]_{0.9}\texttt{holding} \wedge [\texttt{grab}]_{0.6}\texttt{holding}$ has exactly four nodes:

$$\Gamma_0^0 = \{(0, [\texttt{grab}]_{0.9}\texttt{holding} \wedge [\texttt{grab}]_{0.6}\texttt{holding})\}$$

and after the application of rule $\wedge$,

$$\begin{aligned}\Gamma_1^0 = \{&(0, [\texttt{grab}]_{0.9}\texttt{holding} \wedge [\texttt{grab}]_{0.6}\texttt{holding}),\\ &(0, [\texttt{grab}]_{0.9}\texttt{holding}),\\ &(0, [\texttt{grab}]_{0.6}\texttt{holding})\}\end{aligned}$$

and after the application of rule $\Diamond\varphi$ to $(0, [\texttt{grab}]_{0.9}\texttt{holding})$,

$$\begin{aligned}\Gamma_2^0 = \{&(0, [\texttt{grab}]_{0.9}\texttt{holding} \wedge [\texttt{grab}]_{0.6}\texttt{holding}),\\ &(0, [\texttt{grab}]_{0.9}\texttt{holding}),\\ &(0, [\texttt{grab}]_{0.6}\texttt{holding}),\\ &(1, \texttt{holding})\}\end{aligned}$$

and finally, after another application of rule $\Diamond\varphi$, but this time to $(0, [\texttt{grab}]_{0.6}\texttt{holding})$,

$$\begin{aligned}\Gamma_3^0 = \{&(0, [\texttt{grab}]_{0.9}\texttt{holding} \wedge [\texttt{grab}]_{0.6}\texttt{holding}),\\ &(0, [\texttt{grab}]_{0.9}\texttt{holding}),\\ &(0, [\texttt{grab}]_{0.6}\texttt{holding}),\\ &(1, \texttt{holding}),\\ &(2, \texttt{holding})\}.\end{aligned}$$

But stating that a transition to a world at which `holding` is true can be reached with two different probabilities (0.9 and 0.6) is a contradiction.

To determine whether a formula is valid or not, the decision procedure checks whether all branches of a tree are closed or not. Because the branch/tree for the incumbent example should close, we need a procedure which will check for contradictions in sets of dynamic formulae, and if a contradiction is found, create a new node containing $(x, \bot)$ at the end of the applicable branch.

A naïve solution might be to add a tableau rule which deals with this case. However, there are many subtle cases and designing rules to cover all cases is very difficult. And proving that the tableau system with all these rules is complete is challenging, to say the least. One instance of a formula which is a contradiction yet not obviously so, is

$$\begin{aligned}&[\texttt{grab}]_{0.9}\texttt{holding} \quad \wedge\\ &[\texttt{grab}]_{0.1}\neg\texttt{holding} \quad \wedge\\ &[\texttt{grab}]\neg\texttt{full} \quad \wedge\\ &\neg[\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \texttt{holding}),\end{aligned}$$

where the set of possible worlds is all conceivable worlds.

To be certain that all possible contradictions are noticed, a system of equations and disequations

(a system of linear inequalities (SLI)) is generated from a set of dynamic literals concerning the same action appearing in the leaf node of an open branch. Fagin et al. [1990] use a similar idea to prove that the axiomatization of their logic for reasoning about probabilities is complete. This is done for every action in $\mathcal{A}$. The SLI phase (§ 5.2.3) will determine whether to make an open branch closed, depending on whether some SLI generated is feasible or infeasible (feasibility of an SLI is defined in Def. 5.2.4).

Now we explain how a system of linear inequalities can be generated from a set of dynamic literals.

Definition 5.2.1: $W(\Gamma, x) \stackrel{def}{=} \{w \in C \mid w \models \ell \text{ for all } (x, \ell) \in \Gamma \text{ where } \ell \text{ is a propositional literal}\}$.

Definition 5.2.2: $X(\Gamma) \stackrel{def}{=} \{0, 1, \ldots, x'\}$ are all the labels mentioned in Γ.

Definition 5.2.3: $W(\Gamma) \stackrel{def}{=} \bigcup_{x \in X(\Gamma)} W(\Gamma, x)$.

Let $n = |W(\Gamma)|$. Let $W(\Gamma)^{\#} = (w_1, w_2, \ldots, w_n)$ be an ordering of the worlds in $W(\Gamma)$. With each world $w_k \in W(\Gamma)^{\#}$, we associate a rational variable $pr_k \in \mathbb{Q}_{[0,1]}$. One can generate

$$c_{i,1}pr_1 + c_{i,2}pr_2 + \cdots + c_{i,n}pr_n = q_i,$$

for a formula $(x, [\alpha]_{q_i}\varphi_i) \in \Gamma$ and

$$c_{i,1}pr_1 + c_{i,2}pr_2 + \cdots + c_{i,n}pr_n \neq q_i,$$

for a formula $(x, \neg[\alpha]_{q_i}\varphi_i) \in \Gamma$, such that $c_{i,k} = 1$ if $w_k \models \varphi_i$, else $c_{i,k} = 0$, where x represents a label.

Let $\Delta(\alpha)$ be a set of dynamic literals mentioning α, and let $\Delta(\alpha)^{\#} =$

$$([\alpha]_{q_1}\varphi_1, [\alpha]_{q_2}\varphi_2, \ldots, [\alpha]_{q_g}\varphi_g, \neg[\alpha]_{q_{g+1}}\varphi_{g+1}, \neg[\alpha]_{q_{g+2}}\varphi_{g+2}, \ldots, \neg[\alpha]_{q_{g+h}}\varphi_{g+h})$$

be an ordering of the members of $\Delta(\alpha)$. (When SLIs are actually generated in the SLI phase, exactly *which* literals are involved will be specified.)

With this notation in hand, given some α, we define the system

$$\begin{array}{rl}
c_{1,1}pr_1 + c_{1,2}pr_2 + \cdots + c_{1,n}pr_n & = q_1 \\
c_{2,1}pr_1 + c_{2,2}pr_2 + \cdots + c_{2,n}pr_n & = q_2 \\
\vdots & \\
c_{g,1}pr_1 + c_{g,2}pr_2 + \cdots + c_{g,n}pr_n & = q_g \\
c_{g+1,1}pr_1 + c_{g+1,2}pr_2 + \cdots + c_{g+1,n}pr_n & \neq q_{g+1} \\
c_{g+2,1}pr_1 + c_{g+2,2}pr_2 + \cdots + c_{g+2,n}pr_n & \neq q_{g+2} \\
\vdots & \\
c_{g+h,1}pr_1 + c_{g+h,2}pr_2 + \cdots + c_{g+h,n}pr_n & \neq q_{g+h} \\
pr_1 + pr_2 + \cdots + pr_n & = q*,
\end{array} \tag{5.1}$$

where each of the first $g + h$ (in)equalities represents a member in $\Delta(\alpha)^{\#}$ and such that $q* = 1$ or $q* = 0$. Note that due to $q*$ having two possible values, system (5.1) represents two distinct

systems of equations.

Definition 5.2.4: A *solution set* for an SLI S is the set of all solutions of the form $(s_1, s_2, \ldots, s_n)$ for S, where assigning s_i to pr_i for $i = 1, 2, \ldots, n$ solves all the (in)equalities in S simultaneously. An SLI is *feasible* if and only if its solution set is not empty.

Assigning s_i to pr_i for $i = 1, 2, \ldots, n$ simply means that the values which the pr_i variables must take for all the (in)equalities to be simultaneously true are the s_i. These are the feasible transition probabilities to all possible worlds, given some action executed in some world, and given a set of formulae (partially) specifying the action's transition behavior for/from that world.

We shall say that system (5.1), generated as above, is *feasible* if and only if one or both of the two systems (either with $q* = 1$ or with $q* = 0$) has a feasible solution, that is, if and only if the union of their solution sets is not empty.

The equation

$$pr_1 + pr_2 + \cdots + pr_n = q*,$$

is to ensure that either $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 1$ or $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 0$, as stated in Definition 5.1.3 on page 63.

5.2.3 The SLI Phase

Definition 5.2.5: Let $\Delta(\alpha)$ be a set of dynamic literals mentioning α. $Z(\Delta(\alpha))$ is the solution set for the SLI generated from $\Delta(\alpha)$.

Definition 5.2.6: $F(\Gamma, \alpha, x) \stackrel{def}{=} \{[\alpha]_q\varphi \mid (x, [\alpha]_q\varphi) \in \Gamma\} \cup \{\neg[\alpha]_q\varphi \mid (x, \neg[\alpha]_q\varphi) \in \Gamma\}$.

After the tableau phase has completed, the SLI phase begins. For each leaf node Γ^j_k of an open branch, do the following.

> If $Z(F(\Gamma^j_k, \alpha, x)) = \emptyset$ for some action $\alpha \in \mathcal{A}$ and some label $x \in X(\Gamma^j_k)$, then create new leaf node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \bot)\}$.

Definition 5.2.7: A tree is called *finished* after the SLI phase is completed.

Note that all branches of a finished tree are saturated.

Definition 5.2.8: If a tree for $\neg\Psi$ is closed, we write $\vdash \Psi$. If there is a finished tree for $\neg\Psi$ with an open branch, we write $\nvdash \Psi$.

Example

We continue with the example of Figures 5.2 and 5.3. Only the leaf node of the right-most (open) branch of the tree is considered. Using the same kind of analysis made below, it can be shown that every branch which is open after the tree is saturated should close due to the infeasibility of some SLI generated from a set of formulae in the applicable leaf node.

For brevity, denote w_1 as 111 where $w_1 \models \texttt{full} \wedge \texttt{drank} \wedge \texttt{holding}$, w_2 as 110 where $w_2 \models \texttt{full} \wedge \texttt{drank} \wedge \neg\texttt{holding}$, ..., w_8 as 000 where $w_8 \models \neg\texttt{full} \wedge \neg\texttt{drank} \wedge \neg\texttt{holding}$. We shall refer to the open leaf node on the RHS in Figures 5.2 and 5.3 as Γ. Observe that

- $W(\Gamma, 0) = \{010\}$,
- $W(\Gamma, 1) = \{111, 101, 011, 001\}$,
- $W(\Gamma, 2) = \{110, 100, 010, 000\}$,
- $W(\Gamma, 3) = \{011, 010, 001, 000\}$,

and $W(\Gamma) = \{111, 101, 011, 001, 110, 100, 010, 000\} = C$.

0. $F(\Gamma, \texttt{grab}, 0) = \{[\texttt{grab}]_{0.9}\texttt{holding}, [\texttt{grab}]_{0.1}\neg\texttt{holding}, [\texttt{grab}]\neg\texttt{full}, \neg[\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \texttt{holding})\}$.
1. $F(\Gamma, \texttt{grab}, 1) = \{[\texttt{grab}]_{0.9}\texttt{holding}, [\texttt{grab}]_{0.1}\neg\texttt{holding}, [\texttt{grab}]\neg\texttt{full}\}$.
2. $F(\Gamma, \texttt{grab}, 2) = \{[\texttt{grab}]_{0.9}\texttt{holding}, [\texttt{grab}]_{0.1}\neg\texttt{holding}, [\texttt{grab}]\neg\texttt{full}\}$.
3. $F(\Gamma, \texttt{grab}, 3) = \{[\texttt{grab}]_{0.9}\texttt{holding}, [\texttt{grab}]_{0.1}\neg\texttt{holding}, [\texttt{grab}]\neg\texttt{full}\}$.

The system generated from $F(\Gamma, \texttt{grab}, 0)$ is

$$\begin{array}{ccccccccccccccc}
0 & + & 0 & + & 0 & + & 0 & + & pr_5 & + & 0 & + & pr_7 & + & 0 & = & 0.9 \\
0 & + & pr_2 & + & 0 & + & pr_4 & + & 0 & + & pr_6 & + & 0 & + & pr_8 & = & 0.1 \\
0 & + & 0 & + & 0 & + & 0 & + & pr_5 & + & pr_6 & + & pr_7 & + & pr_8 & = & 1 \\
pr_1 & + & 0 & + & pr_3 & + & 0 & + & pr_5 & + & 0 & + & pr_7 & + & 0 & \neq & 0.9 \\
pr_1 & + & pr_2 & + & pr_3 & + & pr_4 & + & pr_5 & + & pr_6 & + & pr_7 & + & pr_8 & = & 1.
\end{array}$$

Due to $pr_5 + pr_6 + pr_7 + pr_8 = 1$ (3rd equation), it must be the case that $pr_5 + pr_7 \neq 0.9$ (4th disequation). But it is required by the first equation that $pr_5 + pr_7 = 0.9$, which forms a contradiction. Thus, there exists an action and a label for which $Z(F(\Gamma, \texttt{grab}, x) = \emptyset$ and the branch closes.

5.3 Properties of the Decision Procedure

All proofs not given here can be found in the appendix Section A.2.

5.3.1 Soundness

Lemma 5.3.1: Let T be a finished tree. For every node Γ in T: If there exists a structure $\mathcal{S}$ such that for all $(x, \Phi) \in \Gamma$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$, then the (sub)tree rooted at Γ is open.

Theorem 5.3.1: (Soundness) If $\vdash \Psi$ then $\models \Psi$. (Contrapositively, if $\not\models \Psi$ then $\not\vdash \Psi$.)

5.3.2 Completeness

We start with the description of the construction of an SLAP structure, given the leaf node Γ of some open branch of a finished tree. First, we define $X(\Gamma)^{\#}$ to be some sequence of labels $(x_1, x_2, \ldots, x_n)$ such that $w_1 \in W(\Gamma, x_1)$, $w_2 \in W(\Gamma, x_2)$, ..., $w_n \in W(\Gamma, x_n)$, where $(w_1, w_2, \ldots, w_n) = W(\Gamma)^{\#}$. By the definition of $W(\Gamma, x_n)$ (Def. 5.2.1), $X(\Gamma)^{\#}$ is not necessarily unique, and it is guaranteed to define at least one sequence of labels. $\mathcal{S} = \langle W, R\rangle$ can be constructed as follows:

- Let $W = W(\Gamma)$.
- For every action $\alpha \in \mathcal{A}$, the accessibility relation R_α can be constructed as follows. Let $R_\alpha = \{(w_i, w_j, s_j^\alpha) \mid$
  - $w_i, w_j \in W(\Gamma)^{\#}$,
  - $x_i \in X(\Gamma)^{\#}$,
  - $(s_1, s_2, \ldots, s_n) \in Z(F(\Gamma, \alpha, x_i))\}$.

Lemma 5.3.2: $\mathcal{S}$ is an SLAP structure.

Proof:
The components of the structure are well-formed:

- $W = W(\Gamma) = \bigcup_{x \in X(\Gamma)}\{w \in C \mid w \models \ell$ for all $(x, \ell) \in \Gamma$ where ℓ is a propositional literal$\}$. That is, $W = \{w \in C \mid$ for all $x, w \models \ell$ for all $(x, \ell) \in \Gamma$ where ℓ is a propositional literal$\}$. Thus, for W to be empty, it must be the case that for all $w \in C$, there exists some $(x, \ell) \in \Gamma$, for which $w \not\models \ell$. But this is a contradiction. Hence, W is not empty.
- Due to Γ being open, we know that $Z(F(\Gamma, \alpha, x))$ is not empty for all $x \in X(\Gamma)$ and all $\alpha \in \mathcal{A}$.

  By construction, R maps each action $\alpha \in \mathcal{A}$ to R_α such that R_α is a relation in $W \times W \times \mathbb{Q}_{[0,1]}$. Moreover, if $(w, w', pr), (w, w', pr') \in R_\alpha$, then $pr = pr'$. This is because in the SLI generated from $F(\Gamma, \alpha, x)$, the same variable represents pr and pr' and it can have only one value. Hence, R_α is a (total) function $R_\alpha : (W \times W) \mapsto \mathbb{Q}_{[0,1]}$.

  And by construction, the fact that $pr_1 + pr_2 + \cdots + pr_n = 1$ or $pr_1 + pr_2 + \cdots + pr_n = 0$ is an equation in any SLI generated, either $\sum_{(w,w',pr)\in R_\alpha} pr = 1$ or $\sum_{(w,w',pr)\in R_\alpha} pr = 0$, for every $w \in W$.

■

W.l.o.g., one can assume that, for every $(x, \Box\Phi) \in \Gamma$, Φ is in DNF.

Lemma 5.3.3: Let Γ be the leaf node of a finished tree, where $(0, \Box\Phi) \in \Gamma$, for some $\Box\Phi \in \mathcal{L}_{SLAP}$. For every label $x \in X(\Gamma)$, there exists a term $(\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m)$ of Φ such that $(x, \Phi_1), (x, \Phi_2), \ldots, (x, \Phi_m) \in \Gamma$.

Proposition 5.3.1: Let Γ be the leaf node of an open branch of a finished tree and let $\mathcal{S}$ be constructed as described above. Let $(x, \delta) \in \Gamma$ where δ is a dynamic literal. If $\mathcal{S}, w \models \delta$ for some $w \in W(\Gamma, x)$, then $\mathcal{S}, w' \models \delta$ for all $w' \in W(\Gamma, x)$.

Proof:
By construction, $R_\alpha = \{(w, w_j, s_j) \mid x \in X(\Gamma), w \in W(\Gamma, x), w_j \in W(\Gamma)^\# \text{ and } (s_1, s_2, \ldots, s_n) \in Z(F(\Gamma, \alpha, x))\}$. Now suppose that $\mathcal{S}, w \models \delta$ for some $w \in W(\Gamma, x)$. Notice that by the definition of R_α above, if there is a solution in $Z(F(\Gamma, \alpha, x)$ for $w \in W(\Gamma, x)$, that solution is available for all $w' \in W(\Gamma, x)$. Therefore, $\mathcal{S}, w' \models \delta$ for all $w' \in W(\Gamma, x)$. The proposition follows. ■

Lemma 5.3.4: If Γ is the leaf node of an open branch of a finished tree, then there exists a structure $\mathcal{S}$ such that for all $(x, \Psi) \in \Gamma$, $\mathcal{S}, w \models \Psi$ for some $w \in W(\Gamma, x)$.

Theorem 5.3.2: (Completeness) If $\models \Psi$ then $\vdash \Psi$. (Contrapositively, if $\nvdash \Psi$ then $\not\models \Psi$.)

5.3.3 Termination

Lemma 5.3.5: A tree for any formula $\Phi \in \mathcal{L}_{SLAP}$ becomes saturated. That is, the tableau phase terminates.

Proof:
We can divide all the tableau rules into two categories: (i) those which add $\perp$ to the new node and (ii) those with the subformula property. Category-(i) rules never cause rules to become applicable later. As a direct consequence of sentences being finite and the subformula property, every category-(ii) rule must eventually become inapplicable. Therefore, all rules eventually become inapplicable, and it follows that any tree (for any formula) would become saturated. ■

Theorem 5.3.3: The entailment decision procedure for SLAP terminates.

Proof:
Due to Lemma 5.3.5, the tableau phase terminates (with a finite number of branches).

Let Γ be the leaf node of an open branch. There is a finite number of labels in $X(\Gamma)$. In the SLI phase: for each open branch of a tree for Φ, a solution set for an SLI is sought (at most) once for each action in $\mathcal{A}$, for each label in $X(\Gamma)$.[3] Hence, a solution set for an SLI is sought a finite number of times in the SLI phase.

Finding the solution set for an SLI is decidable as used in the SLI phase [Dantzig, 1963 & 1998, Kroening and Strichman, 2008] and the process thus terminates in this phase. ■

Corollary 5.3.1: The entailment problem for SLAP is decidable.

Because the procedure is sound (Th. 5.3.1), complete (Th. 5.3.2) and terminating (Th. 5.3.3), entailment is decidable.

5.4 Specifying Domains with SLAP

We provide a framework to formally specify—in the language of SLAP—the domain in which an agent or robot is expected to live. In the context of SLAP, we are interested in three things in the domain of interest:

[3] The proof in the article [Rens et al., 2014a] erroneously refers to "label assignments".

(i) The initial condition IC.

(ii) The static laws SL.

(iii) The action rules AR.

How to write the axioms of AR is the focus of this chapter.

Let the union of all the axioms in SL and AR be denoted by the set BK—the agent's *background knowledge*.

In the context of SLAP we are interested in

$$\{\Box\beta \mid \beta \in BK\} \models IC \rightarrow \Phi,$$

where $\Phi \in \mathcal{L}^{-\Box}_{SLAP}$ is any sentence of interest, $IC \in \mathcal{L}^{-\Box}_{SLAP}$ and $BK \subset \mathcal{L}^{-\Box}_{SLAP}$.

Recall that we use the following abbreviations for constants in our scenario: $\texttt{grab} := g$, $\texttt{drink} := di$, $\texttt{replace} := r$, $\texttt{full} := f$, $\texttt{drank} := d$ and $\texttt{holding} := h$.

In SLAP, one can express that action α has effect φ with probability q under condition ϕ as $\phi \rightarrow [\alpha]_q\varphi$. In general, an effect axiom has the form

$$\phi \rightarrow [\alpha]_{q_1}\varphi_1 \wedge [\alpha]_{q_2}\varphi_2 \wedge \ldots \wedge [\alpha]_{q_n}\varphi_n$$

for expressing the different effects of an action and their associated occurrence probabilities, under a particular condition. To set the stage, we provide a definition of a 'proper' specification of the probabilistic effects of an action.

Definition 5.4.1: For some action $\alpha \in \mathcal{A}$, a set of effect axioms is a *proper effects specification* (or PES for short) if and only if it takes the form

$$\begin{aligned}
\phi_1 &\rightarrow [\alpha]_{q_{11}}\varphi_{11} \wedge \cdots \wedge [\alpha]_{q_{1n}}\varphi_{1n} \\
\phi_2 &\rightarrow [\alpha]_{q_{21}}\varphi_{21} \wedge \cdots \wedge [\alpha]_{q_{2n}}\varphi_{2n} \\
&\vdots \\
\phi_j &\rightarrow [\alpha]_{q_{j1}}\varphi_{j1} \wedge \cdots \wedge [\alpha]_{q_{jn}}\varphi_{jn},
\end{aligned}$$

where (i) no $q_{ik} = 0$, (ii) the transition probabilities $q_{i1}, \ldots, q_{in}$ of any axiom i must sum to 1, (iii) for every i, for any pair of effects φ_{ik} and $\varphi_{ik'}$, $\varphi_{ik} \wedge \varphi_{ik'} \equiv \bot$ and (iv) for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$.

We insist that no $q_{ik} = 0$, because the definition is of the specification of an action's *effects*: Suppose

$$\phi \rightarrow \ldots \wedge [\alpha]_0\varphi \wedge \cdots$$

is an axiom of our background knowledge, then due to no φ-world being reachable via α under condition ϕ, φ cannot be an effect in this case. This axiom should thus not be an *effect* axiom.

Proper specifications of the probabilistic effects of actions g, di and r, respectively, are

$$\begin{aligned}
f \wedge d \wedge \neg h &\rightarrow [g]_{0.7}(f \wedge d \wedge h) \wedge [g]_{0.3}(d \wedge \neg h);\\
f \wedge \neg d \wedge \neg h &\rightarrow [g]_{0.7}(f \wedge \neg d \wedge h) \wedge [g]_{0.3}(\neg d \wedge \neg h);\\
\neg f \wedge d \wedge \neg h &\rightarrow [g]_{0.9}(\neg f \wedge d \wedge h) \wedge [g]_{0.1}(\neg f \wedge d \wedge \neg h);\\
\neg f \wedge \neg d \wedge \neg h &\rightarrow [g]_{0.9}(\neg f \wedge \neg d \wedge h) \wedge [g]_{0.1}(\neg f \wedge \neg d \wedge \neg h).
\end{aligned}$$

$$\begin{aligned}
f \wedge \neg d \wedge h &\rightarrow [di]_{0.85}(\neg f \wedge d \wedge h) \wedge [d]_{0.15}(\neg f \wedge \neg d \wedge h);\\
\neg f \wedge d \wedge h &\rightarrow [di](\neg f \wedge d \wedge h);\\
\neg f \wedge \neg d \wedge h &\rightarrow [di](\neg f \wedge h).
\end{aligned}$$

$$\begin{aligned}
f \wedge d \wedge h &\rightarrow [r](f \wedge d \wedge \neg h);\\
f \wedge \neg d \wedge h &\rightarrow [r](f \wedge \neg d \wedge \neg h);\\
\neg f \wedge d \wedge h &\rightarrow [r](\neg f \wedge d \wedge \neg h);\\
\neg f \wedge \neg d \wedge h &\rightarrow [r](\neg f \wedge \neg d \wedge \neg h).
\end{aligned}$$

The above sets of axioms will be denoted as PES_g, PES_{di} and PES_r respectively. Let $PES_g \cup PES_{di} \cup PES_r = PES_1$.

When trying to capture the behavior or dynamics of an action, one typically wants to capture what objects in the environment the action affects, what objects are not affected, in what situations/conditions the action can be performed and when it can physically not be performed. Observe that action α is *executable* under condition ϕ if there exists an effect axiom with condition ϕ in a PES for α. But one cannot say—given only a PES—when α is *inexecutable* or whether the action may be executable under unmentioned conditions. Finally, one can only say what fluents do not change, under the conditions of the given axioms. However, a PES does not carry the information of whether the axioms are meant to cover *all* conditions. The rest of this chapter is dedicated to dealing with these deficits.

If a knowledge engineer for some reason does not specify what an action α's effects are, given some condition ϕ, but s/he wants to specify that the action is executable in ϕ, then s/he can simply write $\phi \rightarrow [\alpha]_1\top$. According to the following proposition, one can also write $\phi \rightarrow \langle\alpha\rangle\top$ to express executability.

Proposition 5.4.1: $[\alpha]\top \equiv \langle\alpha\rangle\top$.

Proof:
Let $\mathcal{S}$ be an arbitrary SLAP structure and w a world in it.

$\mathcal{S}, w \models [\alpha]\top$
$\iff \left(\sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \top} pr\right) = 1$ (by the truth condition for $[\alpha]\top$)
$\iff$ there exists a $w' \in W$ s.t. $(w, w', pr) \in R_\alpha$ where $pr > 0$ (by Def. 5.1.3)
$\iff \left(\sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \top} pr\right) \neq 0$
$\iff \neg[\alpha]_0\top$
$\iff \langle\alpha\rangle\top$. ∎

To express that α is inexecutable under condition ϕ, the knowledge engineer can write $\phi \rightarrow [\alpha]_0 \top$.

5.5 Invariance

A *frame axiom* [Reiter, 1991] captures the idea of the 'momentum' of a state. That is, things which are unaffected by an action, should remain unaffected after the completion of the action. The general problem of how to minimize or avoid specifying the multitude of frame axioms usually required is known as the *frame problem* [McCarthy and Hayes, 1969]. Bacchus et al. [1999] supply one approach to deal with the frame problem in a language able to express probabilistic transitions. More will be said about this in the concluding remarks at the end of this chapter.

We see in PES_r that for the action r, only h is affected. So for r, the four frame axioms are

$$h \wedge f \rightarrow [r]f; \quad h \wedge \neg f \rightarrow [r]\neg f; \quad h \wedge d \rightarrow [r]d; \quad h \wedge \neg d \rightarrow [r]\neg d.$$

Here, h is the *condition* under which the frame axioms are applicable.

In general, a positive frame axiom has the form

$$FrCond^{+}(\alpha, f) \wedge f \rightarrow [\alpha]f$$

and a negative frame axiom has the form

$$FrCond^{-}(\alpha, f) \wedge \neg f \rightarrow [\alpha]\neg f,$$

where $FrCond^{+}(\alpha, f)$ is a formula stating the conditions under which literal f remains positive and $FrCond^{-}(\alpha, f)$ is a formula stating the conditions under which literal $\neg f$ remains negative.

Instead of stating frame axioms directly, we shall use a slightly more concise expression by collecting all fluents invariant under the same conditions. We define the following abbreviation.

$$Inv(\alpha, \phi, \{f_1, \ldots, f_m\}) \stackrel{def}{=} \text{for } i \in \{1, \ldots, m\},\ \phi \rightarrow ((f_i \rightarrow [\alpha]f_i) \wedge (\neg f_i \rightarrow [\alpha]\neg f_i)),$$

where $f_1, \ldots, f_m \in \mathcal{F}$. $Inv(\alpha, \phi, \{f_1, \ldots, f_m\})$ is called the *invariance predicate*. It is read 'When α is executed under condition ϕ, the truth-values of fluents $f_1, \ldots, f_m$ are invariant.

To relate frame axioms and invariance predicates, note that the following two statements hold ($\Rightarrow$ is read 'implies').

$$\begin{aligned} &\mathcal{S}, w \models Inv(\alpha, FrCond^{+}(\alpha, f) \wedge f, F) \text{ s.t. } f \in F \\ &\Rightarrow \mathcal{S}, w \models FrCond^{+}(\alpha, f) \wedge f \rightarrow [\alpha]f. \end{aligned}$$

$$\begin{aligned} &\mathcal{S}, w \models Inv(\alpha, FrCond^{-}(\alpha, f) \wedge \neg f, F) \text{ s.t. } f \in F \\ &\Rightarrow \mathcal{S}, w \models FrCond^{-}(\alpha, f) \wedge \neg f \rightarrow [\alpha]\neg f. \end{aligned}$$

Note the subtlety that the literal of the right polarity must be included in the condition of the invariance predicate.

We shall collect all invariance predicates in the set INV. Our approach assumes that for every/any α, for all $Inv(\alpha, \phi, F), Inv(\alpha, \phi', F') \in INV$, $\phi \wedge \phi' \equiv \bot$. Furthermore, for every effect axiom $\phi \rightarrow \Phi$ for α, for all $Inv(\alpha, \phi', F) \in INV$, either $\phi \wedge \phi' \equiv \bot$ or $\phi \models \phi'$. These assumptions keep things organized.

Now suppose we have the following invariance predicates (denoted INV_1).

$$
\begin{aligned}
&Inv(g, f \wedge \neg h, \{d\});\\
&Inv(g, \neg f \wedge \neg h, \{f, d\});\\
&Inv(di, \neg f \wedge \neg d \wedge h, \{f, h\});\\
&Inv(di, \neg f \wedge d \wedge h, \{f, d, h\});\\
&Inv(di, f \wedge \neg d \wedge h, \{h\});\\
&Inv(r, h, \{f, d\}).
\end{aligned}
$$

INV_1 is a partial specification of action effects of the oil-drinking scenario. To further specify effects, one can supply the following effect axioms (denoted as PES_2).

$$
\begin{aligned}
f \wedge \neg h &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}\neg h;\\
\neg f \wedge \neg h &\rightarrow [g]_{0.9} h \wedge [g]_{0.1}\neg h;\\
f \wedge \neg d \wedge h &\rightarrow [di]_{0.85}(\neg f \wedge d) \wedge [di]_{0.15}(\neg f \wedge \neg d);\\
h &\rightarrow [r]\neg h.
\end{aligned}
$$

Note that $\bigwedge_{\beta \in PES_1} \beta \equiv \bigwedge_{\delta \in INV_1 \cup PES_2} \delta$, but $INV_1 \cup PES_2$ is significantly smaller than PES_1.

Furthermore, we shall assume that effect and invariance specifications are complete, that is, that the knowledge engineer makes the completeness assumption about these specifications. Recall that $\langle\alpha\rangle\top$ abbreviates $\neg[\alpha]_0\top$ and note that $\mathcal{S}, w \models [\alpha]_q\varphi$ for $q > 0$ and $\varphi \not\equiv \bot$ if and only if $\mathcal{S}, w \models \neg[\alpha]_0\top$ (for all $\mathcal{S}$ and w). Therefore, if there is an effect axiom invariance predicate with condition ϕ, then one can assume the presence of an executability axiom $\phi \rightarrow \langle\alpha\rangle\top$. However, we must still specify that an action is inexecutable when none of the effect axiom conditions or invariance predicate conditions is met. Hence, the following *inexecutability* axiom is assumed present.[4]

$$\langle\alpha\rangle\top \rightarrow (\phi_1 \vee \cdots \vee \phi_j) \vee \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi,$$

where $\phi_1, \ldots, \phi_j$ are the conditions of the effect axioms for α and $Cond^{Inv}(\alpha)$ is the set of all the conditions mentioned in the invariance predicates for α. The inexecutability axioms for our example are

$$\langle g\rangle\top \rightarrow \neg h; \qquad \langle di\rangle\top \rightarrow h \wedge \neg(f \wedge d); \qquad \langle r\rangle\top \rightarrow h. \tag{5.2}$$

We shall collect all inexecutability axioms in the set INX. We shall refer to the set (5.2) in particular as INX_1.

[4] Inexecutability axioms are also called *condition closure* axioms.

5.6 From Underspecified to Completely Specified Transition Models

In this section, we define a *complete* transition model specification to capture the notion of a model which provides information about transitions between *every* pair of states. The two subsections provide approaches for completing partial specifications. A transition model specification is partial when some fluent in the specification is *underspecified*. The notion of an "underspecified" fluent is formalized by Definitions 5.6.2 and 5.6.3 below. We propose two alternative approaches for completing partial specifications in the subsections below. In Section 5.7, it is proved that these approaches do in fact generate complete transition model specifications.

Definition 5.6.1: A *complete transition model specification* (short: CTS) for action α is any set $B \subset \mathcal{L}$ involving only action α, such that there exists a structure $\mathcal{S} = \langle C, R\rangle$ and $\mathcal{S} \models \bigwedge_{\beta\in B} \beta$, and there is no $\mathcal{S}' = \langle C, R'\rangle$ such that $\mathcal{S}' \models \bigwedge_{\beta\in B} \beta$.

A PES is, in general, not a CTS: Let $\mathcal{F} = \{f_1\}$, $\alpha_1 \in \mathcal{A}$ and $B = \{f_1 \rightarrow [\alpha_1]f_1\}$. Then B is a PES for α_1. And let $w_1 \models f_1$ and $w_2 \models \neg f_1$. Assume $\mathcal{S}' \models f_1 \rightarrow [\alpha_1]f_1$, where $\mathcal{S}' = \langle C, R'\rangle$, $(w_2, w_2, 0.4) \in R'_{\alpha_1}$ and $(\alpha_1, R'_{\alpha_1}) \in R'$ and assume $\mathcal{S}'' \models f_1 \rightarrow [\alpha_1]f_1$, where $\mathcal{S}'' = \langle C, R''\rangle$, $(w_2, w_2, 0.5) \in R''_{\alpha_1}$ and $(\alpha_1, R''_{\alpha_1}) \in R''$. But the two structures $\mathcal{S}'$ and $\mathcal{S}''$ are different. Therefore, $f_1 \rightarrow [\alpha_1]f_1$ does not uniquely specify the accessibility relation for α_1. But the definition of a CTS says it must be unique.

Suppose a *completeness assumption* about effect axioms is as follows: The conditions of effect axioms for action α specifies all the conditions under which α has an effect, that is, under which α causes a fluent to change (see, e.g., Reiter [1991], §2.3). In deterministic systems, if one makes the completeness assumption about effect conditions, one can deduce frame axioms from the effect axioms [Reiter, 1991]. But effect axioms for non-deterministic systems are different, and frame axioms are not enough: Let $BK^{od} := INV_1 \cup PES_2 \cup INX_1$. Note that $BK^{od} \not\models f \wedge \neg h \rightarrow [g]_q(f \wedge \neg h) \vee [g]_{q'}(\neg f \wedge \neg h)$ for any q and q'. One could assume, due to lack of knowledge, that the truth-value of f does not change, that is

$$BK^{od} \models f \wedge \neg h \rightarrow [g]_{0.3}(f \wedge \neg h),$$

or one could assume a uniform distribution of probability over the possible values of f, that is

$$BK^{od} \models f \wedge \neg h \rightarrow [g]_{0.15}(f \wedge \neg h) \wedge [g]_{0.15}(\neg f \wedge \neg h).$$

There seems to be no clear way to decide between the two assumptions without knowledge of the domain; it depends on the domain of interest. In the following approaches dealing with this issue, we require two rather complicated definitions. Each definition is followed by an intuitive explanation.

Definition 5.6.2: Let $\Upsilon^{\mathit{eff}}(\alpha, \phi) \stackrel{\mathit{def}}{=} \{\phi' \mid Inv(\alpha, \phi', F) \in INV, \phi \wedge \phi' \not\equiv \bot\}$. Given effect axiom $\phi \rightarrow [\alpha]_{q_1}\varphi_1 \wedge [\alpha]_{q_2}\varphi_2 \wedge \ldots \wedge [\alpha]_{q_n}\varphi_n$ for α of a PES, fluent f is *effectively underspecified* in effect $\varphi \in \{\varphi_1, \varphi_2, \cdots, \varphi_n\}$ under condition $\phi \wedge \phi'$ if $[\alpha]_q\varphi \not\models [\alpha]_q(\varphi \wedge f)$ and $[\alpha]_q\varphi \not\models [\alpha]_q(\varphi \wedge \neg f)$ and $f \notin F$, where $Inv(\alpha, \phi', F) \in INV$ for $\phi' \in \Upsilon^{\mathit{eff}}(\alpha, \phi)$.

Intuitively, a fluent is underspecified in an effect φ (of the effect axiom for α with condition ϕ) under condition $\phi \wedge \phi'$, if its truth-value is unknown after transition $[\alpha]_q\varphi$, and the invariance

predicate for α with condition ϕ' does not reveal the truth-value of the fluent.

Definition 5.6.3: Let $\Upsilon^{inv}(\alpha, \phi') \stackrel{def}{=} \{\phi \mid \phi \rightarrow \Phi \text{ is an effect axiom for } \alpha, \phi' \wedge \phi \not\equiv \bot\}$. Given invariance predicate $Inv(\alpha, \phi, F) \in INV$, fluent f is *invariantly underspecified* under condition $\phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi$ if this condition is not a contradiction and if $f \notin F$.

Intuitively, a fluent is underspecified under satisfiable condition $\phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi$, if the invariance predicate for α with condition ϕ' does not reveal the truth-value of the fluent.

The definitions assume that all relevant information about effects of actions is contained in a clearly defined PES and set INV. If effect information were not easily located in this manner, it would be very difficult to 'complete' the specifications of effects of actions as is done subsequently. In other words, our proposal for the management of probabilistic transition models includes the requirement that a PES and a set INV are clearly defined and accessible by the system or system-user.

When we say a fluent is underspecified, we mean it in the sense of one or more of Definitions 5.6.2 and 5.6.3. Next, we propose two alternative approaches for completing incompletely specified transition models. The approaches are: When a fluent is underspecified under a particular condition, (1) assume that it is invariant under that condition or (2) assume that it is uniformly distributed under that condition.

It is important to know that when we say "assume the presence of" some formulae in the approaches below, we do not intend that the formulae necessarily be added to the agent's background knowledge. Our intention is that the knowledge engineer decides before hand which default assumption must be made when information is underspecified. The inference engine should then *simulate* the presence of the applicable formulae assumed present in some efficient manner. In this way, representation, that is, the work that the knowledge engineer must do, remains concise.

5.6.1 Always Assuming Invariance

For every $\alpha \in \mathcal{A}$ and $f \in \mathcal{F}$, for all conditions ϕ of effect axioms for α and conditions ϕ' of invariance predicates for α, if f is effectively underspecified in effect φ under condition $\phi \wedge \phi'$, assume the presence of *frame axioms*

$$\phi \wedge \phi' \wedge f \rightarrow [\alpha]_q(\varphi \wedge f) \quad \text{and} \quad \phi \wedge \phi' \wedge \neg f \rightarrow [\alpha]_q(\varphi \wedge \neg f).$$

For every $\alpha \in \mathcal{A}$ and $f \in \mathcal{F}$, for all conditions ϕ' of invariance predicates for α, if f is invariantly underspecified under condition $\phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi$, assume the presence of frame axioms

$$\phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi \wedge f \rightarrow [\alpha]_q(\varphi \wedge f) \quad \text{and} \quad \phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi \wedge \neg f \rightarrow [\alpha]_q(\varphi \wedge \neg f).$$

Given PES_1, INV is empty and there are thus no invariably underspecified fluents. However, f

and d are effectively underspecified, leading to the assumption of these invariance formulae:

$$\begin{array}{rcl} f \wedge d \wedge \neg h & \rightarrow & [g]_{0.3}(d \wedge \neg h \wedge f); \\ f \wedge \neg d \wedge \neg h & \rightarrow & [g]_{0.3}(\neg d \wedge \neg h \wedge f); \\ \neg f \wedge \neg d \wedge h & \rightarrow & [di](\neg f \wedge \neg d \wedge h). \end{array} \tag{5.3}$$

Given PES_2 and INV_1: f is effectively underspecified in effect $\neg h$ under condition $(f \wedge \neg h) \wedge (f \wedge \neg h)$. The invariance formulae due to f are thus

$$(f \wedge \neg h) \wedge f \rightarrow [g]_{0.3}(\neg h \wedge f) \quad \text{and} \quad (f \wedge \neg h) \wedge \neg f \rightarrow [g]_{0.3}(\neg h \wedge \neg f),$$

which simplifies to

$$f \wedge \neg h \rightarrow [g]_{0.3}(\neg h \wedge f). \tag{5.4}$$

d is invariantly underspecified under condition $(\neg f \wedge \neg d \wedge h) \wedge \neg(f \wedge \neg d \wedge h)$ (which is semantically equivalent to $\neg f \wedge \neg d \wedge h$). The invariance formulae due to d are thus

$$(\neg f \wedge \neg d \wedge h) \wedge d \rightarrow [di]d \quad \text{and} \quad (\neg f \wedge \neg d \wedge h) \wedge \neg d \rightarrow [di]\neg d,$$

which simplifies to

$$\neg f \wedge \neg d \wedge h \rightarrow [di]\neg d. \tag{5.5}$$

Proposition 5.6.1: $PES_1 \cup \{(5.3)\}$ is semantically equivalent to $PES_2 \cup INV_1 \cup \{(5.4), (5.5)\}$.

Proof:
Observe that

$$\begin{array}{l} (f \wedge d \wedge \neg h \rightarrow [g]_{0.3}(d \wedge \neg h \wedge f)) \ \wedge \ (f \wedge \neg d \wedge \neg h \rightarrow [g]_{0.3}(\neg d \wedge \neg h \wedge f)) \\ \equiv Inv(g, f \wedge \neg h, \{d\}) \wedge (f \wedge \neg h \rightarrow [g]_{0.3}(\neg h \wedge f)), \end{array}$$

where the formulae on the LHS are in $\{(5.3)\}$ and on the RHS, $Inv(g, f \wedge \neg h, \{d\}) \in INV_1$ and $f \wedge \neg h \rightarrow [g]_{0.3}(\neg h \wedge f)$ is (5.4). And observe that

$$\begin{array}{l} \neg f \wedge \neg d \wedge h \rightarrow [di](\neg f \wedge \neg d \wedge h) \\ \equiv (\neg f \wedge \neg d \wedge h \rightarrow [di](\neg f \wedge h)) \wedge (\neg f \wedge \neg d \wedge h \rightarrow [di]\neg d), \end{array}$$

where the formula on the LHS is in $\{(5.3)\}$ and on the RHS, $\neg f \wedge \neg d \wedge h \rightarrow [di](\neg f \wedge h) \in PES_1$ and $\neg f \wedge \neg d \wedge h \rightarrow [di]\neg d$ is (5.5). ■

Proposition 5.6.1 is supporting evidence that our strategies are correct.

5.6.2 Always Assuming Uniform Distribution

Let $U^{\mathit{eff}}(\alpha, \phi, \phi', \varphi) \stackrel{def}{=} \{f \in \mathcal{F} \mid f$ is effectively underspecified for α in effect φ under condition $\phi \wedge \phi'\}$, where ϕ is the condition of some effect axiom for α and ϕ' is the condition of some invariance predicate for α, and $U^{inv}(\alpha, \phi') \stackrel{def}{=} \{f \in \mathcal{F} \mid f$ is invariantly underspecified for α under condition $\phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi$ as described in Definition 5.6.3$\}$.

For every $\alpha \in \mathcal{A}$, for all conditions ϕ of effect axioms for α, for every transition $[\alpha]_q\varphi$ of the axiom, for all conditions ϕ' of invariance predicates for α, assume the presence of *equiprob formula*

$$\phi \rightarrow [\alpha]_{q_1}(\varphi \wedge \gamma_1) \wedge \cdots \wedge [\alpha]_{q_m}(\varphi \wedge \gamma_m),$$

where $\{\gamma_1, \ldots, \gamma_m\}$ is the $m = 2^{|U^{\mathit{eff}}(\alpha,\phi,\phi',\varphi)|}$ permutations of conjunctions of literals, given all the fluents in $U^{\mathit{eff}}(\alpha, \phi, \phi', \varphi)$ and $q_1 = \cdots = q_m = q/m$. For instance, if $U^{\mathit{eff}}(\alpha, \phi, \phi', \varphi) = \{f_2, f_4\}$ then the literal conjunction permutations are $\{f_2 \wedge f_4,\ f_2 \wedge \neg f_4,\ \neg f_2 \wedge f_4,\ \neg f_2 \wedge \neg f_4\}$.

For every action $\alpha \in \mathcal{A}$, for all conditions ϕ' of invariance predicate for α, assume the presence of *equiprob formula*

$$\phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi \rightarrow [\alpha]_{q_1}\gamma_1 \wedge \cdots \wedge [\alpha]_{q_n}\gamma_n,$$

where $\{\gamma_1, \ldots, \gamma_n\}$ are the $n = 2^{|U^{inv}(\alpha,\phi')|}$ permutations of conjunctions of literals, given all the fluents in $U^{inv}(\alpha, \phi')$ and $q_1 = \cdots = q_n = q/n$. Note that an equiprob formula needs not be stated if its condition $\phi' \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi$ is a contradiction. The same goes for cases when $U^{inv}(\alpha, \phi')$ is empty.

Given PES_1, INV is empty and there are thus no invariably underspecified fluents. However, f and d are effectively underspecified, leading to the assumption of these equiprob formulae:

$$\begin{array}{rcl} f \wedge d \wedge \neg h & \rightarrow & [g]_{0.15}(d \wedge \neg h \wedge f) \wedge [g]_{0.15}(d \wedge \neg h \wedge \neg f); \\ f \wedge \neg d \wedge \neg h & \rightarrow & [g]_{0.15}(\neg d \wedge \neg h \wedge f) \wedge [g]_{0.15}(\neg d \wedge \neg h \wedge \neg f); \\ \neg f \wedge \neg d \wedge h & \rightarrow & [d]_{0.5}(\neg f \wedge d \wedge h) \wedge [d]_{0.5}(\neg f \wedge \neg d \wedge h). \end{array} \tag{5.6}$$

Given INV_1 and PES_2: $U^{\mathit{eff}}(g, f \wedge \neg h, f \wedge \neg h, \neg h) = \{f\}$ (due to $[g]_{0.3}\neg h$ in effect axiom $f \wedge \neg h \rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}\neg h$). Hence, equiprob formula

$$f \wedge \neg h \rightarrow [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f) \tag{5.7}$$

is assumed present. For all other conditions (combinations of ϕ and ϕ'), $U^{\mathit{eff}}(\alpha, \phi, \phi', \varphi) = \emptyset$. And due to invariance predicate $Inv(di, \neg f \wedge \neg d \wedge h, \{f, h\})$ in INV_1, $U^{inv}(di, \neg f \wedge \neg d \wedge h) = \{d\}$. Hence, equiprob formula

$$\neg f \wedge \neg d \wedge h \rightarrow [di]_{0.5}d \wedge [di]_{0.5}\neg d \tag{5.8}$$

is assumed present. For all other conditions ϕ', $U^{inv}(\alpha, \phi') = \emptyset$.

Remark 5.6.1: $PES_1 \cup \{(5.6)\}$ is semantically equivalent to $PES_2 \cup INV_1 \cup \{(5.7), (5.8)\}$.

5.7 The Two Approaches are Full Specifications

This section presents a theorem (Thm. 5.7.1) which proves that there exists a process for transforming a PES, a set INV and the associated frame axioms or equiprob formulae into a complete transition model specification (CTS). After the theorem and its proof, we transform PES_2 and

INV_1 and the associated frame axioms or equiprob formulae given in Sections 5.6.1 and 5.6.2, respectively, to illustrate the transformation process.

If there exists a world $w \in C$ such that $w \models \delta$, where δ is a propositional formula, and for all $w' \in C$, if $w' \neq w$ then $w' \not\models \delta$, we say that δ is *definitive* (then, δ defines a world; δ is a *complete propositional theory*). Let cpt be the smallest set of all definitive formulae induced from $\mathcal{F}$.

Definition 5.7.1: A *verbose effects specification* (VES) is a PES where all effect axiom conditions (the ϕ left of the $\rightarrow$) and effects (the φ right of the $\rightarrow$) are definitive formulae.

Lemma 5.7.1: Let INX^V be $\neg\langle\alpha\rangle\top \rightarrow \neg(\phi_1 \vee \cdots \vee \phi_j)$, where $\phi_1, \ldots, \phi_j$ are the j conditions of the j effect axioms in a VES V for α, and V is generated using the process described in the proof of Theorem 5.7.1. Then $INX^V \wedge \bigwedge_{\beta \in V} \beta$ is a CTS.

Proof:
We must show that there exists a unique $R_\alpha : (C \times C) \mapsto \mathbb{Q}_{[0,1]}$ which is a total function from pairs of worlds into the rationals, and for every $w^- \in C$, either $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 1$ or $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 0$, such that $(\alpha, R_\alpha) \in R$ and $\langle C, R\rangle \models INX^V \wedge \bigwedge_{\beta\in V} \beta$.

For the sake of reference, let

$$\phi \rightarrow [\alpha]_{q_1}\varphi_1 \wedge [\alpha]_{q_2}\varphi_2 \wedge \ldots \wedge [\alpha]_{q_n}\varphi_n$$

be an arbitrary effect axiom of V. We may refer to the axiom as η. Construct R_α as follows: For all $w^-, w^+ \in C$: If $w^- \not\models (\phi_1 \vee \cdots \vee \phi_j)$, then $(w^-, w^+, 0) \in R_\alpha$. Else if $w^- \models \phi$: if $w^+ \models \varphi_k$ then $(w^-, w^+, q_k) \in R_\alpha$, else if $w^+ \not\models \varphi_1 \vee \cdots \vee \varphi_n$ then $(w^-, w^+, 0) \in R_\alpha$.

Now, the domain and co-domain of R_α are clearly adhered to. R_α is a *function* because of the constraint of a PES that for every i, for any pair of effects φ_{ik} and $\varphi_{ik'}$, $\varphi_{ik} \wedge \varphi_{ik'} \equiv \bot$, that is, never is more than one probability specified for reaching a world w^+ from some world w^-.

R_α is a *total* function because, given any pair $(w^-, w^+) \in (C \times C)$, if $w^- \models \psi_i$ where ϕ_i is the condition of the i-th effect axiom, then either (i) $w^+ \models \varphi_{ik}$ for some transition $[\alpha]_q\varphi_{ik}$ in the axiom, in which case $(w^-, w^+, q) \in R_\alpha$ or (ii) $w^+ \not\models \varphi_{ik}$ for all transitions in the axiom, in which case $(w^-, w^+, 0) \in R_\alpha$, due to the PES constraint that the transition probabilities $q_{i1}, \ldots, q_{in}$ of any axiom i must sum to 1. Else, for all $w^- \in C$ such that $w^- \models \neg(\phi_1 \vee \cdots \vee \phi_j)$, $(w^-, w^+, 0) \in R_\alpha$ for all $w^+ \in C$, due to the PES constraint that for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$. It follows implicitly that for every $w^- \in C$, either $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 1$ or $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 0$.

Simply, by construction of R_α, it follows that $\langle C, R\rangle \models INX^V$. And as a direct consequence of the construction of R_α, it follows that $\langle C, R\rangle \models \bigwedge_{\beta\in V} \beta$.

We shall now show that no other R'_α ($\neq R_\alpha$) can be constructed such that $(\alpha, R'_\alpha) \in R'$ and $\langle C, R'\rangle \models INX^V \wedge \bigwedge_{\beta\in V} \beta$. Let (w^-, w^+, q_k) be some element of R_α as constructed. Let $q' \in \mathbb{Q}_{[0,1]}$ such that $|q' - q_k| > 0$. If $w^- \not\models (\phi_1 \vee \cdots \vee \phi_j)$, then $(w^-, w^+, q') \in R'_\alpha$, where $q' > 0$. But then $\langle C, R'\rangle \not\models INX^V$. And if $w^- \models \phi$ and $w^+ \models \varphi_k$, then $q_k \neq q'$ and $\langle C, R'\rangle \not\models \phi \rightarrow [\alpha]_{q_k}\varphi_k$, which implies that $\langle C, R'\rangle \not\models \eta$, which implies that $\langle C, R'\rangle \not\models \bigwedge_{\beta\in V} \beta$. Else, if $w^- \models \phi$ and $w^+ \not\models \varphi_1 \vee \cdots \vee \varphi_n$ then $(w^-, w^+, q') \in R_\alpha$ where $q' \neq 0$. But this is a contradiction, because it is required that $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 1$, but due to the PES constraint

that the transition probabilities $q_{i1}, \ldots, q_{in}$ of any axiom i must sum to 1, $\sum_{(w^-,w^+,pr)\in R_\alpha} pr > 1$. ■

Definition 5.7.2: Note that any formula $\phi \rightarrow \Phi$ (with condition ϕ) such that $\phi \not\models f$ and $\phi \not\models \neg f$ for some $f \in \mathcal{F}$, is semantically equivalent to $(\phi \wedge f \rightarrow \Phi) \wedge (\phi \wedge \neg f \rightarrow \Phi)$. Given this observation, one can expand any formula of the form $\phi \rightarrow \Phi$ into a set of semantically equivalent formulae, each with a definitive condition. We shall refer to this process applied to a set X of formulae as *expending X by conditions*.

Proposition 5.7.1: $(\phi \rightarrow [\alpha]\varphi) \wedge (\phi' \rightarrow [\alpha]_q\varphi') \wedge (\phi' \rightarrow \phi) \models \phi' \rightarrow [\alpha]_q(\varphi \wedge \varphi')$ for all $q \in \mathbb{Q}_{[0,1]}$.

Proof:
Let $\mathcal{S}$ be an arbitrary SLAP structure and w a world in it. Suppose $\mathcal{S}, w \models (\phi \rightarrow [\alpha]\varphi) \wedge (\phi' \rightarrow [\alpha]_q\varphi') \wedge (\phi' \rightarrow \phi)$. Assume $\mathcal{S}, w \models \phi'$. Then $\mathcal{S}, w \models \phi$ (and thus $\mathcal{S}, w \models [\alpha]\varphi$) and $\mathcal{S}, w \models [\alpha]_q\varphi'$ (i.e., $\mathcal{S}, w \models \phi \wedge [\alpha]\varphi \wedge [\alpha]_q\varphi'$.

Now, $\mathcal{S}, w \models [\alpha]\varphi \wedge [\alpha]_q\varphi'$ iff

$$\sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \varphi} pr = 1 \quad \text{and} \quad \sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \varphi'} pr = q.$$

Therefore, $\{(w, w', pr) \in R_\alpha \mid w' \models \varphi'\} \subseteq \{(w, w', pr) \in R_\alpha \mid w' \models \varphi, pr > 0\}$. Hence, $\mathcal{S}, w \models$ iff $\sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \varphi \wedge \varphi'} pr = q$, and by definition, $\mathcal{S}, w \models [\alpha]_q(\varphi \wedge \varphi')$. ■

Proposition 5.7.2: A fluent cannot be effectively and invariantly underspecified under the same condition.

Proof:
By definition of $\Upsilon^{inv}(\alpha, \phi')$, whenever $\phi \wedge \phi' \not\equiv \bot$, then $\phi \in \Upsilon^{inv}(\alpha, \phi')$. Note that $\neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi$ $\equiv \neg\phi_1 \wedge \neg\phi_2 \wedge \cdots \wedge \neg\phi_n$, where $\Upsilon^{inv}(\alpha, \phi') = \{\phi_1, \phi_2, \ldots, \phi_n\}$. But $\phi \in \{\phi_1, \phi_2, \ldots, \phi_n\}$. Therefore, $\phi \wedge \neg \bigvee_{\phi \in Cond^{Inv}(\alpha)} \phi \equiv \bot$ and the proposition follows. ■

Theorem 5.7.1: For both approaches, given a PES Π for α, a set of invariance predicates INV for α, an inexecutability axiom INX for α derived from Π and INV, a set of frame axioms FA for α and a set of equiprob formulae EF for α, their union is a CTS for α.

Proof:
Suppose V is a VES for α and INX^V is an inexecutability axiom derived from V as in Lemma 5.7.1. If we can show that V and INX^V exist such that $INX \wedge \bigwedge_{\beta \in \Pi \cup INV \cup FA \cup EF} \equiv INX^V \wedge \bigwedge_{\delta \in V} \delta$, then by Lemma 5.7.1, we have proved the theorem. Hence, we show how to convert $\Pi \cup INV \cup FA \cup EF$ into a semantically equivalent VES V and we prove that $INX \equiv INX^V$.

We show how to 'enlarge' Π into a VES using four rewrite rules involving INV, IF and EF. The rewrite rules are:

1. For every $Inv(\alpha, \phi, \{f_1, \ldots, f_m\}) \in INV$, for $i \in \{1, \ldots, m\}$, add $\phi \wedge f_i \rightarrow [\alpha]f_i$ and $\phi \wedge \neg f_i \rightarrow [\alpha]\neg f_i$ to INV if $\phi \wedge f_i$, respectively, $\phi \wedge \neg f_i$ is satisfiable. Remove all invariance predicates from INV.
2. Expand Π, INV, FF and EF by conditions.

3. For every $\phi' \rightarrow [\alpha]\ell \in INV$, if there is no $\phi \rightarrow \Phi \in \Pi$ such that $\phi \equiv \phi'$, then add $\phi' \rightarrow [\alpha]\top$ to Π.

4. When assuming invariance: Note that EF is empty. For every $\phi \rightarrow \Phi \in \Pi$, for every $\phi' \rightarrow [\alpha]_q\varphi' \in FA$ such that $\phi \equiv \phi'$, for every $[\alpha]_q\varphi$ in Φ, if $\varphi' \models \varphi$, then replace $[\alpha]_q\varphi$ by $[\alpha]_q\varphi'$.

   When assuming uniform distribution: Note that FA is empty. For every $\phi \rightarrow \Phi \in \Pi$, for every $\phi \rightarrow [\alpha]_{q_1}(\varphi \wedge \gamma_1) \wedge \cdots \wedge [\alpha]_{q_m}(\varphi \wedge \gamma_m), \in EF$ such that $\phi \equiv \phi'$, for every $[\alpha]_q\varphi$ in Φ, if $\varphi_i \models \varphi$ for all $i \in \{1, \ldots, m\}$, then replace $[\alpha]_q\varphi$ by $[\alpha]_{q_1}(\varphi \wedge \gamma_1) \wedge \cdots \wedge [\alpha]_{q_m}(\varphi \wedge \gamma_m)$.

5. For every $\phi \rightarrow \Phi \in \Pi$, if $\phi' \rightarrow [\alpha]\ell \in INV$ such that $\phi \equiv \phi'$, then replace every $[\alpha]\varphi$ in Φ by $[\alpha](\varphi \wedge \ell)$.

Recalling that $Inv(\alpha, \phi, \{f_1, \ldots, f_m\})$ abbreviates: for $i \in \{1, \ldots, m\}$, $\phi \rightarrow ((f_i \rightarrow [\alpha]f_i) \wedge (\neg f_i \rightarrow [\alpha]\neg f_i))$, step 1 is sound. By Definition 5.7.2, step 2 is sound. Due to $\phi' \rightarrow [\alpha]\ell \models \phi' \rightarrow [\alpha]\top$, step 3 is sound. Note that, by Proposition 5.7.2, no formula in Π is modified/rewritten twice in step 4. Hence, the process in step 4 does not cause side-effects. And together with the two assumptions and the respective explanations of the approaches to complete the specifications, step 4 is sound. By Proposition 5.7.1, step 5 is sound.

Note that the two approaches of Sections 5.6.1 and 5.6.2 are designed exactly to deal with fluents not dealt with before, that is, to appropriately add literals corresponding to every fluent not mentioned in an effect of some effect axiom, for every effect in every axiom. Thus, by the nature of INV, FA and EF and the rewrite rules of steps 4 and 5, every effect of every effect axiom is now a definitive formula.

A VES is a PES; the conditions of all its effect axioms must thus be disjoint. It is assumed that Π is initially a PES, hence, with disjoint conditions. None of the rewrite rules causes some pair of conditions in Π to be joint: In particular, step 2: Expansion by conditions cannot cause joint conditions.

Observe that INX depends only on the axiom conditions of the original Π, which has essentially the same axiom conditions as those of V (given that condition expansion in step 2 does not add conditions), and INX^V depends only on the axiom conditions of V. Hence $INX \equiv INX^V$. ■

Example

PES_2 and INV_1 presented in Section 5.5 are repeated here, for convenience:

INV_1:

$$Inv(g, f \wedge \neg h, \{d\});$$
$$Inv(g, \neg f \wedge \neg h, \{f, d\});$$
$$Inv(di, \neg f \wedge \neg d \wedge h, \{f, h\});$$
$$Inv(di, \neg f \wedge d \wedge h, \{f, d, h\});$$
$$Inv(di, f \wedge \neg d \wedge h, \{h\});$$
$$Inv(r, h, \{f, d\}).$$

PES_2:

$$\begin{aligned} f \wedge \neg h &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}\neg h; \\ \neg f \wedge \neg h &\rightarrow [g]_{0.9}h \wedge [g]_{0.1}\neg h; \\ f \wedge \neg d \wedge h &\rightarrow [di]_{0.85}(\neg f \wedge d) \wedge [di]_{0.15}(\neg f \wedge \neg d); \\ h &\rightarrow [r]\neg h. \end{aligned}$$

When assuming invariance (AI) by default, FA:

$$\begin{aligned} f \wedge \neg h &\rightarrow [g]_{0.3}(\neg h \wedge f); \\ \neg f \wedge \neg d \wedge h &\rightarrow [di]\neg d. \end{aligned}$$

When assuming uniform distribution (AUD) by default, EF:

$$\begin{aligned} f \wedge \neg h &\rightarrow [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f); \\ \neg f \wedge \neg d \wedge h &\rightarrow [di]_{0.5}d \wedge [di]_{0.5}\neg d. \end{aligned}$$

The transformation process is now applied to these sentences, using the five rewrite rules presented in the proof of Theorem 5.7.1.

Rule 1
INV_1:

$$\begin{aligned} f \wedge \neg h \wedge d &\rightarrow [g]d; \\ f \wedge \neg h \wedge \neg d &\rightarrow [g]\neg d; \\ \neg f \wedge \neg h &\rightarrow [g]\neg f; \\ \neg f \wedge \neg h \wedge d &\rightarrow [g]d; \\ \neg f \wedge \neg h \wedge \neg d &\rightarrow [g]\neg d; \\ \neg f \wedge \neg d \wedge h &\rightarrow [di]\neg f; \\ \neg f \wedge \neg d \wedge h &\rightarrow [di]h; \\ \neg f \wedge d \wedge h &\rightarrow [di]\neg f; \\ \neg f \wedge d \wedge h &\rightarrow [di]d; \\ \neg f \wedge d \wedge h &\rightarrow [di]h; \\ f \wedge \neg d \wedge h &\rightarrow [di]h; \\ h \wedge f &\rightarrow [r]f; \\ h \wedge \neg f &\rightarrow [r]\neg f; \\ h \wedge d &\rightarrow [r]d; \\ h \wedge \neg d &\rightarrow [r]\neg d. \end{aligned}$$

Rule 2
PES_2:

$$\begin{aligned}
f \wedge \neg h \wedge d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}\neg h;\\
f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}\neg h;\\
\neg f \wedge \neg h \wedge d &\rightarrow [g]_{0.9}h \wedge [g]_{0.1}\neg h;\\
\neg f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.9}h \wedge [g]_{0.1}\neg h;\\
f \wedge \neg d \wedge h &\rightarrow [di]_{0.85}(\neg f \wedge d) \wedge [di]_{0.15}(\neg f \wedge \neg d);
\end{aligned}$$

$$\begin{aligned}
h \wedge f \wedge d &\rightarrow [r]\neg h;\\
h \wedge f \wedge \neg d &\rightarrow [r]\neg h;\\
h \wedge \neg f \wedge d &\rightarrow [r]\neg h;\\
h \wedge \neg f \wedge \neg d &\rightarrow [r]\neg h.
\end{aligned}$$

When AI, FA:

$$\begin{aligned}
f \wedge \neg h \wedge d &\rightarrow [g]_{0.3}(\neg h \wedge f);\\
f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.3}(\neg h \wedge f);\\
\neg f \wedge \neg d \wedge h &\rightarrow [di]\neg d.
\end{aligned}$$

When AUD, EF:

$$\begin{aligned}
f \wedge \neg h \wedge d &\rightarrow [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f);\\
f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f);\\
\neg f \wedge \neg d \wedge h &\rightarrow [di]_{0.5}d \wedge [di]_{0.5}\neg d.
\end{aligned}$$

INV_1:

$$\begin{aligned}
f \wedge \neg h \wedge d &\rightarrow [g]d;\\
f \wedge \neg h \wedge \neg d &\rightarrow [g]\neg d;\\
\neg f \wedge \neg h \wedge d &\rightarrow [g]\neg f;\\
\neg f \wedge \neg h \wedge \neg d &\rightarrow [g]\neg f;\\
\neg f \wedge \neg h \wedge d &\rightarrow [g]d;\\
\neg f \wedge \neg h \wedge \neg d &\rightarrow [g]\neg d;\\
\neg f \wedge \neg d \wedge h &\rightarrow [di]\neg f;\\
\neg f \wedge \neg d \wedge h &\rightarrow [di]h;\\
\neg f \wedge d \wedge h &\rightarrow [di]\neg f;\\
\neg f \wedge d \wedge h &\rightarrow [di]d;\\
\neg f \wedge d \wedge h &\rightarrow [di]h;\\
f \wedge \neg d \wedge h &\rightarrow [di]h;\\
h \wedge f \wedge d &\rightarrow [r]f;\\
h \wedge \neg f \wedge d &\rightarrow [r]\neg f;\\
h \wedge d \wedge f &\rightarrow [r]d;\\
h \wedge \neg d \wedge f &\rightarrow [r]\neg d;\\
h \wedge f \wedge \neg d &\rightarrow [r]f;\\
h \wedge \neg f \wedge \neg d &\rightarrow [r]\neg f;\\
h \wedge d \wedge \neg f &\rightarrow [r]d;\\
h \wedge \neg d \wedge \neg f &\rightarrow [r]\neg d.
\end{aligned}$$

Rule 3
The following are added to PES_2.

$$\begin{aligned}
\neg f \wedge \neg d \wedge h &\rightarrow [di]\top;\\
\neg f \wedge d \wedge h &\rightarrow [di]\top.
\end{aligned}$$

Rule 4
When AI, the changes in PES_2 are:

$$\begin{aligned}
f \wedge \neg h \wedge d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}(\neg h \wedge f);\\
f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}(\neg h \wedge f);\\
\neg f \wedge \neg d \wedge h &\rightarrow [di]\neg d.
\end{aligned}$$

When AUD, the changes in PES_2 are:

$$\begin{aligned} f \wedge \neg h \wedge d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f); \\ f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f); \\ \neg f \wedge \neg d \wedge h &\rightarrow [di]_{0.5} d \wedge [di]_{0.5} \neg d. \end{aligned}$$

Rule 5
When AI, PES_2:

$$\begin{aligned} f \wedge \neg h \wedge d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}(\neg h \wedge f); \\ f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.3}(\neg h \wedge f); \\ \neg f \wedge \neg h \wedge d &\rightarrow [g]_{0.9}(h \wedge \neg f \wedge d) \wedge [g]_{0.1}(\neg h \wedge \neg f \wedge d); \\ \neg f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.9}(h \wedge \neg f \wedge \neg d) \wedge [g]_{0.1}(\neg h \wedge \neg f \wedge \neg d); \\ f \wedge \neg d \wedge h &\rightarrow [di]_{0.85}(\neg f \wedge d \wedge h) \wedge [di]_{0.15}(\neg f \wedge \neg d \wedge h); \\ \neg f \wedge \neg d \wedge h &\rightarrow [di](\neg d \wedge \neg f \wedge h); \\ \neg f \wedge d \wedge h &\rightarrow [di](\top \wedge \neg f \wedge d \wedge h); \\ h \wedge f \wedge d &\rightarrow [r](\neg h \wedge f \wedge d); \\ h \wedge f \wedge \neg d &\rightarrow [r](\neg h \wedge \neg d \wedge f); \\ h \wedge \neg f \wedge d &\rightarrow [r](\neg h \wedge \neg f \wedge d); \\ h \wedge \neg f \wedge \neg d &\rightarrow [r](\neg h \wedge \neg f \wedge \neg d). \end{aligned}$$

When AUD, PES_2:

$$\begin{aligned} f \wedge \neg h \wedge d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f); \\ f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.7}(f \wedge h) \wedge [g]_{0.15}(\neg h \wedge f) \wedge [g]_{0.15}(\neg h \wedge \neg f); \\ \neg f \wedge \neg h \wedge d &\rightarrow [g]_{0.9}(h \wedge \neg f \wedge d) \wedge [g]_{0.1}(\neg h \wedge \neg f \wedge d); \\ \neg f \wedge \neg h \wedge \neg d &\rightarrow [g]_{0.9}(h \wedge \neg f \wedge \neg d) \wedge [g]_{0.1}(\neg h \wedge \neg f \wedge \neg d); \\ f \wedge \neg d \wedge h &\rightarrow [di]_{0.85}(\neg f \wedge d \wedge h) \wedge [di]_{0.15}(\neg f \wedge \neg d \wedge h); \\ \neg f \wedge \neg d \wedge h &\rightarrow [di]_{0.5}(d \wedge \neg f \wedge h) \wedge [di]_{0.5}(\neg d \wedge \neg f \wedge h); \\ \neg f \wedge d \wedge h &\rightarrow [di](\top \wedge \neg f \wedge d \wedge h); \\ h \wedge f \wedge d &\rightarrow [r](\neg h \wedge f \wedge d); \\ h \wedge f \wedge \neg d &\rightarrow [r](\neg h \wedge \neg d \wedge f); \\ h \wedge \neg f \wedge d &\rightarrow [r](\neg h \wedge \neg f \wedge d); \\ h \wedge \neg f \wedge \neg d &\rightarrow [r](\neg h \wedge \neg f \wedge \neg d). \end{aligned}$$

The following is the set of inexecutability axioms INX_2 derived from the new PES_2:

$$\begin{aligned} \langle g \rangle &\rightarrow (f \wedge \neg h \wedge d) \vee (f \wedge \neg h \wedge \neg d) \vee (\neg f \wedge \neg h \wedge d) \vee (\neg f \wedge \neg h \wedge \neg d); \\ \langle di \rangle &\rightarrow (f \wedge \neg d \wedge h) \vee (\neg f \wedge \neg d \wedge h) \vee (\neg f \wedge d \wedge h); \\ \langle r \rangle &\rightarrow (h \wedge f \wedge d) \vee (h \wedge f \wedge \neg d) \vee (h \wedge \neg f \wedge d) \vee (h \wedge \neg f \wedge \neg d) \end{aligned}$$

which is semantically equivalent to

$$\begin{aligned} \langle g \rangle \top &\rightarrow \neg h; \\ \langle di \rangle \top &\rightarrow h \wedge \neg(f \wedge d); \\ \langle r \rangle \top &\rightarrow h. \end{aligned}$$

which is identical to INX_1 (Formulae 5.2, page 79). Furthermore, PES_2 is a VES, and it is thus easy to see that $PES_2 \cup INX_2$ is three transition model specifications for the three actions.

5.8 Concluding Remarks

We presented a non-nesting logic (SLAP) for modeling probabilistic transition systems. The logic is based on modal logic with possible worlds semantics. We proved that determining whether an SLAP sentence is valid is decidable. We showed how entailment can be cast as a validity problem and an example domain problem was presented. A crucial part of proving SLAP decidable was reliant on the decidability of the feasibility of systems of linear inequalities.

Adding stochastic observations, that is, a means to reason about noisy sensing to SLAP will yield the logic called SLAOP, a logic for specifying partially observable Markov decision processes (POMDPs) [Monahan, 1982, Lovejoy, 1991].

The $\Box$ (necessity) operator in SLAP may not be nested. This makes the logic less expressive, in a sense, than logics which do allow nesting of necessity operators. For instance, the related logics $\mathcal{LAP}$ and $\mathcal{ESP}$ allow nesting of necessity operators. This 'deficiency' also applies to our SLAOP. However, SLAP and SLAOP were designed to be steps in the design of our Stochastic Decision Logic (SDL). SDL does allow for some form of modal operator nesting, however, not exactly the same kind as in $\mathcal{LAP}$ and $\mathcal{ESP}$. This issue will be discussed in the chapter defining SDL.

Iocchi et al. [2009] also say that one of their aims is to extend $\mathcal{E}+$ to represent POMDPs. But it seems that their extension of $\mathcal{E}+$ and our extension of SLAP to achieve POMDP specifications will result in significantly different logics, with possibly different computability and computational properties.

The work appearing in this chapter has been presented at a workshop [Rens et al., 2013]. However, Definitions 5.6.2 and 5.6.3 above, replace three definitions in the workshop paper. With the new definitions, the two approaches corresponding to the two assumptions (Secs. 5.6.1 and 5.6.2) are slightly simpler. Furthermore, these two approaches for generating full specifications, presented in this chapter, do not constrain the knowledge engineer as much: In the approach presented in the workshop paper, the following constraints were in place. For every effect axiom $\phi \rightarrow \Phi$ for α, for all $Inv(\alpha, \phi', F) \in INV$, either $\phi \wedge \phi' \equiv \bot$ or $\phi \models \phi'$. These constraints are no longer necessary with the new approach.

We proved that our two approaches to complete underspecified transition models result in complete transition model specifications.

There seems to be two issues with underspecified models. One is knowing what information is missing. The other is deciding what information to add and how to add it correctly and completely.

We have presented a systematic approach to generate complete specifications of probabilistic transition models (CTSs) with a probabilistic modal logic. For these specifications to be more compact than they would be if transition probabilities were simply written down, it is expected that a user/knowledge engineer will capture (with sentences of a logical language) some transition information from the domain of interest, and then for missing information, express the desired transition behavior of the model of the domain, and finally, for information still not provided by the user, s/he must take a stance as to what the default transition behavior should be: invariance of the truth-values of fluents not mentioned in the effect axioms, or uniform distribution of transition probabilities. In real world situations, a combination of assumptions may be more effective. For instance, in a very dynamic environment, the default should perhaps be 'variance'. That is, when information is not given about how the truth-value of a fluent should change when some action is executed, it could be assumed that the fluent's value will *always* change. Nevertheless, assuming (necessary) (in)variance is an assumption of certainty; these are 'minimum entropy'/certain information assumptions and could be studied under the topic of traditional nonmonotonic reasoning [Brewka, 2012].

The 'uniform distribution' assumption on the other hand is a kind of 'maximum entropy' approach. Wang and Schmolze [2005] have a very similar approach to ours to achieve compact representations in POMDP planning. Some researchers (see, e.g., Grove et al. [1994] and the work of Kern-Isberner [2001] and colleagues) have proposed the assignment of a unique probability distribution over a vocabulary such that information theoretic entropy is maximized while the available probabilistic information is conserved. This *principle of maximum entropy* [Jaynes, 1978] seems to be a reasonable approach, but it may also be reasonable to assume a particular *a priori* probability distribution for a given domain when no other information is forthcoming. Although "default reasoning about probabilities" [Jaeger, 1994] is usually applied to what is believed in the *current* situation, the idea is easily applied to what will be believed in the *next* situation, that is, to transition models.

Another approach to more compact specifications is via notions of conditional independence of Belief Networks. See, for example, Fierens et al. [2005] for a starting point in the area of combining belief nets with logic. We have not looked at the relationship between the notion of invariance and conditional independence in a probabilistic setting.

Bacchus et al. [1999] give an account of specifying stochastic actions in the situation calculus while retaining Reiter's solution to the frame problem [Reiter, 1991] via successor-state axioms (SSAs). In particular, §3 of their paper shows how to deal with a nondeterministic action by 'decomposing' it into a set of deterministic actions, each leading to one of the effects of the nondeterministic action. In SLAP, stochastic (nondeterministic) actions are specified 'directly', actions are not decomposed. The 'direct approach' corresponds more closely to POMDP models than the 'decomposition approach', and thus aligns better with logics with explicit POMDP semantics. We could thus not rely on Reiter's solution. A deeper study is needed to compare the pros and cons of using decomposition and SSAs, on the one hand, and using our direct approach without SSAs, on the other hand.

Finally, it might be a good idea to define a *variance* set $Var(\cdot)$ instead of the invariance predicate:

$$Var(\alpha, \phi) \stackrel{def}{=} \{f \in \mathcal{F} \mid f\text{'s truth-value varies with the execution of } \alpha \text{ under condition } \phi\}.$$

There are usually fewer variant fluents than invariant fluents, given some action. It will thus likely be more efficient to specify $Var(\cdot)$ than $Inv(\cdot)$. A decision procedure must then use the information captured by $Var(\cdot)$ appropriately. This is very close to the idea of the dependence relation $\rightsquigarrow$ of Castilho et al. [1999]. This is left for future work.

The next chapter introduces the Specification Logic of Actions and Observations with Probability (SLAOP), and how to use it. Its development was important in that it clarifies issues involving reasoning with stochastic actions and observations. We shall see in the latter part of Chapter 6that SLAOP is useful as a stand-alone logic.

6. THE SPECIFICATION LOGIC OF ACTIONS AND OBSERVATIONS WITH PROBABILITY

An early version of the logic presented in this chapter was presented at the sixth Starting Artificial Intelligence Research Symposium (STAIRS) in Montpellier, France [Rens et al., 2012]. The current version was presented at the eighth international symposium on Foundations of Information and Knowledge Systems (FoIKS) in Bordeaux, France [Rens et al., 2014b].

In order for robots and intelligent agents in stochastic domains to reason about actions and observations, they must first have a *representation* or *model* of the domain over which to reason. For example, a robot may need to represent available knowledge about its `grab` action in its current situation. It may need to represent that when 'grabbing' the oil-can, there is a 5% chance that it will knock over the oil-can. As another example, if the robot has access to information about the weight of an oil-can, it may want to represent the fact that the can weighs heavy 90% of the time in 'situation A', but that it is heavy 98% of the time in 'situation B'.

This chapter presents the Specification Logic of Actions and Observations with Probability (SLAOP). The logic is meant to facilitate the specification of domains with agents whose actions *and* observations are stochastic. With SLAOP, POMDP models can be represented compactly. However, models which provide the initial belief-state cannot be represented, because SLAOP cannot express the notion of a belief-state.

The version of SLAOP presented here is an extension of SLAP from the two preceding chapters. SLAP is extended with (i) notions of rewards and action costs, (ii) a notion of equality between actions and observations and (iii) observations for dealing with perception/sensing. To establish a correspondence between POMDPs and SLAOP, SLAOP must view observations as objects at the same semantic level as actions. We make use of ideas introduced with LAO to add observations as first-class objects.

The present version of SLAOP presented at the FoIKS symposium includes significant progress on the preliminary version presented at STAIRS. We mention only some of the major changes between the two versions. Firstly, the present version inherits the $\Box$ operator from SLAP, which is important for marking sentences as globally applicable axioms. The preliminary version of SLAOP had no $\Box$ operator. Another change is, instead of the predicate $(\varsigma \mid \alpha : q)$ used in the present version, a modal operator $[\varsigma \mid \alpha]_q\varphi$ with a slightly different definition was used in the 'old' SLAOP. $[\varsigma \mid \alpha]_q\varphi$ can be read 'The probability of perceiving ς in a world in which φ holds is equal to q, given α was performed.' It turned out that specifying φ creates unwanted interactions with the modal operator $[\alpha]_q\varphi$ for specifying transition probabilities. Moreover, we have determined that $(\varsigma \mid \alpha : q)$ (with the given meaning; cf. § 6.1.2) is sufficient for specifying perception probabilities (cf. § 6.5). Last and most importantly, the decision procedure of the preliminary version relied on many intricate tableau rules; relying on the solvability of systems of inequalities (as in the

present version) is much cleaner and the decidability of such systems carries over to help prove the decidability of SLAOP. The decision procedure for the previous version of SLAOP was not proven complete. The current version is proved complete and terminating.

Recall the oil-drinking domain with

- actions $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{weigh}, \texttt{replace}\}$,
- observations $\Omega = \{\texttt{obsNil}, \texttt{obsLight}, \texttt{obsMedium}, \texttt{obsHeavy}\}$ and
- fluents $\mathcal{F} = \{\texttt{full}, \texttt{drank}, \texttt{holding}\}$.

Intuitively, when the robot performs a `weigh` action (i.e., it activates its 'weight' sensor) it will perceive either `obsLight`, `obsMedium` or `obsHeavy`; for other actions, it will perceive `obsNil`.

Given a formalization BK of our scenario, the robot may have the following queries:

- If the oil-can is empty and I'm not holding it, is there a 0.9 probability that I'll be holding it after grabbing it, and a 0.1 probability that I'll have missed it? That is, does $(\neg\texttt{full} \wedge \neg\texttt{holding}) \rightarrow ([\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \texttt{holding}) \wedge [\texttt{grab}]_{0.1}(\neg\texttt{full} \wedge \neg\texttt{holding}))$ follow from BK?
- If the oil-can is not full, I've drunk the oil and I'm holding the can, is there a 0.7 probability of perceiving the can is light, given I weighed it? That is, does $(\neg\texttt{full} \wedge \texttt{drank} \wedge \texttt{holding}) \rightarrow (\texttt{obsLight} \mid \texttt{weigh} : 0.7)$ follow from BK?

In the second half of this chapter, we show how SLAOP can be used for specifying models of domains with stochastic actions and observations, including action rewards and costs.

Specifying the action model is very similar to how it is done with LAO, but necessarily includes techniques used with SLAP. Here, we shall not investigate the 'completion' of *proper effects specifications* (PESs) as is done in the Chapter 5, however, we shall assume that the specifications are PESs.

Specifying the probabilistic perception models must be done carefully for them to be internally coherent and coherent with the action models. Specifying rewards and costs is simple.

Section 6.1 defines SLAOP. Section 6.2 provides a decision procedure for determining entailment of sentences in SLAOP. In Section 6.3, we prove that the procedure is sound, complete and that it terminates, that is, we show that SLAOP is decidable with respect to entailment. Section 6.4 discusses the specification of action rules, perception rules and utility rules. In Section 6.5, we present two examples of how the entailment decision procedure for SLAOP can be used.

6.1 Defining the Logic

First we present the syntax of SLAOP, then we state its semantics.

6.1.1 Syntax

The vocabulary of the language contains six sorts of objects of interest:

1. a finite set of *fluents* (alias, *propositional atoms*) $\mathcal{F} = \{f_1, \ldots, f_n\}$,
2. a finite set of names of atomic *actions* $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$,
3. a finite set of names of atomic *observations* $\Omega = \{\varsigma_1, \ldots, \varsigma_n\}$,
4. all *real numbers* $\mathbb{R}$,[1]
5. a countable set of *action variables* $V_{\mathcal{A}} = \{v_1^{\alpha}, v_2^{\alpha}, \ldots\}$,
6. a countable set of *observation variables* $V_{\Omega} = \{v_1^{\varsigma}, v_2^{\varsigma}, \ldots\}$.

From now on, we denote $\mathbb{R} \cap [0, 1]$ as $[0, 1]$. We shall refer to elements of $\mathcal{A} \cup \Omega$ as *constants*. We are going to work in a multi-modal setting, in which we have modal operators $[\alpha]_q$, one for each $\alpha \in \mathcal{A}$ and $q \in [0, 1]$, and predicates $(\varsigma \mid \alpha : q)$, one for each pair in $\Omega \times \mathcal{A}$ and $q \in [0, 1]$.

Definition 6.1.1: Let $f \in \mathcal{F}$, $\alpha \in (\mathcal{A} \cup V_{\mathcal{A}})$, $\varsigma \in (\Omega \cup V_{\Omega})$, $v \in (V_{\mathcal{A}} \cup V_{\Omega})$, $q \in [0, 1]$ and $r \in \mathbb{R}$. The language of SLAOP, denoted $\mathcal{L}_{SLAOP}$, is the least set of Ψ defined by the grammar:

$$\varphi ::= f \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi.$$
$$\Phi ::= \varphi \mid \alpha = \alpha \mid \varsigma = \varsigma \mid \mathit{Reward}(r) \mid \mathit{Cost}(\alpha, r) \mid [\alpha]_q\varphi \mid (\varsigma \mid \alpha : q) \mid (\forall v)\Phi \mid \neg\Phi \mid \Phi \wedge \Phi.$$
$$\Psi ::= \Phi \mid \Box\Phi \mid \neg\Psi \mid \Psi \wedge \Psi.$$

The scope of quantifier $(\forall v)$ is determined in the same way as is done in first-order logic. A variable v appearing in a formula Ψ is said to be *bound* by quantifier $(\forall v)$ if and only if v is the same variable as v and is in the scope of $(\forall v)$. If a variable is not bound by any quantifier, it is *free*. In $\mathcal{L}_{SLAOP}$, variables are not allowed to be free; they are always bound.

Note that formulae with nested modal operators of the form $\Box\Box\Phi$, $\Box\Box\Box\Phi$, $[\alpha]_q[\alpha]_q\varphi$ and $[\alpha]_q[\alpha]_q[\alpha]_q\varphi$ et cetera are not in $\mathcal{L}_{SLAOP}$. 'Single-step' or 'flat' formulae are sufficient to *specify* action transition probabilities, that is, for specifying a transition model. To reason about the effects of sequences of actions, nesting may be appropriate, but SLAOP is not for reasoning at that level. As usual, we treat $\bot, \vee, \rightarrow$ and $\leftrightarrow$ as abbreviations. $\rightarrow$ and $\leftrightarrow$ have the weakest bindings and $\neg$ the strongest; parentheses enforce or clarify the scope of operators conventionally.

The definition of a POMDP reward function $R(a, s)$ may include not only the reward value of state s, but it may deduct the cost of performing a in s. It will be convenient for the person specifying a POMDP using SLAOP to be able to specify action costs independently from the rewards of states, because these two notions are not necessarily connected. To specify rewards and execution costs in SLAOP, we require Reward and Cost as special predicates. $\mathit{Reward}(r)$ can be read 'The reward for being in the current situation is r units' and we read $\mathit{Cost}(\alpha, c)$ as 'The cost for executing α is c units'.

$[\alpha]_q\varphi$ is read 'The probability of reaching a φ-world after executing α, is equal to q'. $[\alpha]$ abbreviates $[\alpha]_1$. $(\varsigma \mid \alpha : q)$ is read 'The probability of perceiving ς, given α was performed, is q'.

[1] In SLAP [Rens et al., 2014a] and the previous version of SLAOP [Rens et al., 2012], rational numbers were used. Due to our completeness proof relying on Tarski [1957]'s quantifier elimination method which involves real numbers, we use real numbers here.

$\langle\alpha\rangle\varphi$ abbreviates $\neg[\alpha]_0\varphi$ and is read 'It is possible to reach a world in which φ holds after executing α'. Note that $\langle\alpha\rangle\varphi$ does not mean $\neg[\alpha]\neg\varphi$. $[\alpha]_q\varphi$ and $\neg[\alpha]_q\varphi$ are referred to as *dynamic literals*. $(\varsigma \mid \alpha : q)$ and $\neg(\varsigma \mid \alpha : q)$ are referred to as *perception literals*.

One reads $\Box\Phi$ as 'Φ holds in every possible world'. We require the $\Box$ operator to mark certain information (sentences) as holding in *all* possible worlds. These sentences are, essentially, the axioms which model the domain of interest. $(\forall v^{\alpha})$ is to be read 'For all actions' and $(\forall v^{\varsigma})$ is to be read 'For all observations'. $(\forall v)\Phi$ (where $v \in (V_{\mathcal{A}} \cup V_{\Omega})$) can be thought of as a syntactic shorthand for the finite conjunction of Φ with the variables replaced by the constants of the right sort (cf. Def. 6.1.3 for the formal definition). $(\exists v)\Phi$ abbreviates $\neg(\forall v)\neg\Phi$.

6.1.2 Semantics

SLAOP structures extend SLAP structures. Recall that $w : \mathcal{F} \mapsto \{0, 1\}$ is a total function that assigns a truth-value to each fluent. And C is the set of *conceivable worlds*.

SLAP structures are comparable to Markov decision processes (MDPs) [Puterman, 1994] without reward functions, whereas SLAOP structures are comparable to POMDPs (with reward functions). A POMDP model is a tuple $\langle S, A, T, R, \Omega, O, b^0\rangle$; S is a finite set of states the agent can be in; A is a finite set of actions the agent can choose to execute; T is the function defining the probability of reaching one state from another for each action; R is a function giving the expected immediate reward gained by the agent for any state and agent action; Ω is a finite set of observations the agent can experience of its world; O is a function giving a probability distribution over observations for any state and action performed to reach that state; b^0 is the initial probability distribution over all states in S.

A SLAOP structure is a 'translation' of a POMDP model, except for the initial belief-state b^0.[2]

Definition 6.1.2: A SLAOP structure is a tuple $\mathcal{S} = \langle W, R, O, N, Q, U\rangle$ such that

1. $W \subseteq C$ a non-empty set of *possible worlds*.
2. $R : \mathcal{A} \mapsto R_{\alpha}$, where $R_{\alpha} : (W \times W) \mapsto [0, 1]$ is a total function from pairs of worlds into the reals; That is, R is a mapping that provides an accessibility relation R_{α} for each action $\alpha \in \mathcal{A}$; For every $w^{-} \in W$, it is required that either $\sum_{w^{+}\in W} R_{\alpha}(w^{-}, w^{+}) = 1$ or $\sum_{w^{+}\in W} R_{\alpha}(w^{-}, w^{+}) = 0$.
3. O is a nonempty finite set of observations;
4. $N : \Omega \mapsto O$ is a bijection that associates to each name in Ω, a unique observation in O;
5. $Q : \mathcal{A} \mapsto Q_{\alpha}$, where $Q_{\alpha} : (W \times O) \mapsto [0, 1]$ is a total function from pairs in $W \times O$ into the reals; That is, Q is a mapping that provides a perceivability relation Q_{α} for each action $\alpha \in \mathcal{A}$; For all $w^{-}, w^{+} \in W$: if $R_{\alpha}(w^{-}, w^{+}) > 0$, then $\sum_{o\in O} Q_{\alpha}(w^{+}, o) = 1$, that is, there is a probability distribution over observations in a reachable world; Else if $R_{\alpha}(w^{-}, w^{+}) = 0$, then $\sum_{o\in O} Q_{\alpha}(w^{+}, o) = 0$;

[2] Specification of the initial belief-state is required at a higher level of reasoning. SDL is designed for that level (see Chap. 7).

6. U is a pair $\langle Re, Co\rangle$, where $Re : W \mapsto \mathbb{R}$ is a reward function and Co is a mapping that provides a cost function $Co_\alpha : C \mapsto \mathbb{R}$ for each $\alpha \in \mathcal{A}$.

Note that the set of possible worlds may be the whole set of conceivable worlds.

R_α defines the transition probability $pr \in [0,1]$ between worlds w^+ and world w^- via action α. If $R_\alpha(w^-, w^+) = 0$, then w^+ is said to be *inaccessible* or *not reachable* via α performed in w^-, else if $R_\alpha(w^-, w^+) > 0$, then w^+ is said to be *accessible* or *reachable* via action α performed in w^-. If for some w^-, $\sum_{w^+ \in W} R_\alpha(w^-, w^+) = 0$, we say that α is *inexecutable* in w^-.

Q_α defines the observation probability $pr \in [0,1]$ of observation o perceived in world w^+ after the execution of action α. Assuming w^+ is accessible, if $Q_\alpha(w^+, o) > 0$, then o is said to be *perceivable* in w^+, given α, else if $Q_\alpha(w^+, o) = 0$, then o is said to be *unperceivable* in w^+, given α. The definition of perceivability relations implies that there is always at least one possible observation in any world reached due to an action.

Because N is a bijection, it follows that $|O| = |\Omega|$. (We take $|X|$ to be the cardinality of set X.) The value of the reward function $Re(w)$ is a real number representing the reward an agent gets for being in or getting to the world w. It must be defined for each $w \in W$. The value of the cost function $Co(\alpha, w)$ is a real number representing the cost of executing α in the world w. It must be defined for each action $\alpha \in \mathcal{A}$ and each $w \in W$.

Definition 6.1.3 (Truth Conditions): Let $\mathcal{S}$ be a SLAOP structure, with $\alpha, \alpha' \in \mathcal{A}$, $q, pr \in [0,1]$ and $r \in \mathbb{R}$. Let $f \in \mathcal{F}$ and let Φ be any sentence in $\mathcal{L}_{SLAOP}$. We say Φ is *satisfied* at world w in structure $\mathcal{S}$ (written $\mathcal{S}, w \models \Phi$) if and only if the following holds:

$\mathcal{S}, w \models \top$ for all $w \in W$;

$\mathcal{S}, w \models f \iff w(f) = 1$ for $w \in W$;

$\mathcal{S}, w \models \neg\Psi \iff \mathcal{S}, w \not\models \Psi$;

$\mathcal{S}, w \models \Psi \wedge \Psi' \iff \mathcal{S}, w \models \Psi$ and $\mathcal{S}, w \models \Psi'$;

$\mathcal{S}, w \models (\alpha = \alpha') \iff \alpha, \alpha' \in \mathcal{A}$ are the same element;

$\mathcal{S}, w \models (\varsigma = \varsigma') \iff \varsigma, \varsigma' \in \Omega$ are the same element;

$\mathcal{S}, w \models Reward(r) \iff Re(w) = r$;

$\mathcal{S}, w \models Cost(\alpha, r) \iff Co_\alpha(w) = r$;

$\mathcal{S}, w \models [\alpha]_q\varphi \iff \sum_{w' \in W, \mathcal{S}, w' \models \varphi} R_\alpha(w, w') = q$;

$\mathcal{S}, w \models (\varsigma \mid \alpha : q) \iff Q_\alpha(w, N(\varsigma)) = q$;

$\mathcal{S}, w \models \Box\Phi \iff$ for all $w' \in W, \mathcal{S}, w' \models \Phi$;

$\mathcal{S}, w \models (\forall v^\alpha)\Phi \iff \mathcal{S}, w \models \Phi|^{v^\alpha}_{\alpha_1} \wedge \ldots \wedge \Phi|^{v^\alpha}_{\alpha_n}$;

$\mathcal{S}, w \models (\forall v^\varsigma)\Phi \iff \mathcal{S}, w \models \Phi|^{v^\varsigma}_{\varsigma_1} \wedge \ldots \wedge \Phi|^{v^\varsigma}_{\varsigma_n}$,

where we write $\Phi|^v_c$ to mean the formula Φ with all variables $v \in (V_\mathcal{A} \cup V_\Omega)$ appearing in it replaced by constant $c \in \mathcal{A} \cup \Omega$ of the right sort.

A formula Ψ is *valid* in a SLAOP structure (denoted $\mathcal{S} \models \Psi$) if $\mathcal{S}, w \models \Psi$ for every $w \in W$. Ψ is *SLAOP-valid* (denoted $\models \Psi$) if Ψ is true in every structure $\mathcal{S}$. If $\models \theta \leftrightarrow \psi$, we say θ and ψ are *semantically equivalent* (abbreviated $\theta \equiv \psi$).

Ψ is *satisfiable* if $\mathcal{S}, w \models \Psi$ for some $\mathcal{S}$ and $w \in W$. A formula that is not satisfiable is *unsatisfiable* or a *contradiction*. The truth of a propositional formula depends only on the world in which it is evaluated. We may thus write $w \models \Psi$ instead of $\mathcal{S}, w \models \Psi$ when Ψ is a propositional formula.

Let $\mathcal{K} \subseteq \mathcal{L}_{SLAOP}$ and $\Psi \in \mathcal{L}_{SLAOP}$. We say that Ψ is a *local semantic consequence* of $\mathcal{K}$ (denoted $\mathcal{K} \models \Psi$) if for all structures $\mathcal{S}$, and all $w \in W$ of $\mathcal{S}$, if $\mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ then $\mathcal{S}, w \models \Psi$. We shall also say that $\mathcal{K}$ *entails* Ψ whenever $\mathcal{K} \models \Psi$. In fact, $\mathcal{K} \models \Psi$ if and only if $\models \bigwedge_{\kappa \in \mathcal{K}} \kappa \rightarrow \Psi$ (i.e., $\mathcal{K}$ entails Ψ iff $\bigwedge_{\kappa \in \mathcal{K}} \kappa \rightarrow \Psi$ is SLAOP-valid).

If there exists a world $w \in C$ such that $w \models \delta$, where δ is a propositional formula, and for all $w' \in C$, if $w' \neq w$ then $w' \not\models \delta$, we say that δ is *definitive* (then, δ defines a world; δ is a *complete propositional theory*). Let $cpt(\varphi)$ be all the definitive formulae which entail φ, that is, $cpt(\varphi) = \{\delta \in \mathcal{L}_{SLAOP} \mid \delta \text{ is definitive and } \delta \models \varphi\}$.

6.2 Decision Procedure for Semantic Consequence

In this section we describe a decision procedure which has two phases: creation of a tableau tree (the *tableau* phase) which essentially eliminates propositional connectives, then a phase which checks for inconsistencies given possible mappings from 'labels' (of the tableau calculus) to worlds (the *label assignment* phase). Particularly, in the label assignment phase, solutions for systems of inequalities (equations and disequalities) are sought.

We point out that the label assignment phase corresponds to the SLI phase of the decision procedure for SLAP, in the sense that for both logics, the second phase proceeds after the tableau phase and it deals with systems of equations. The difference however, is that for SLAOP the meaning of labels are explicitly addressed (see § 6.2.3). In fact, a label assignment approach could also have been taken for SLAP. However, due to SLAOP being more expressive and thus more complicated than SLAP, we thought it prudent—for the sake of clarity—to address the meaning of labels more directly in SLAOP. The proof of completeness for SLAOP's decision procedure is then easier to understand. Nevertheless, SLAOP's decision procedure is potentially computationally complexer. Finally, in SLAP, sentences of the form $\neg\Box\Phi$ are not in the language, but due to the label assignment approach, sentences of this form can be dealt with in SLAOP.

6.2.1 The Tableau Phase

A preprocessing step occurs, where all (sub)formulae of the form $(\forall v^{\alpha})\Phi$ and $(\forall v^{\varsigma})\Phi$ are replaced by, respectively, $(\Phi|^{v^{\alpha}}_{\alpha_1} \wedge \ldots \wedge \Phi|^{v^{\alpha}}_{\alpha_n})$ and $(\Phi|^{v^{\varsigma}}_{\varsigma_1} \wedge \ldots \wedge \Phi|^{v^{\varsigma}}_{\varsigma_n})$. The occurrence of $(\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)$ in rule obs (below) is only an abbreviation for the semantically equivalent formula without a quantifier and variables.

The tableau rules for SLAOP follow. Let Γ^j_k be a leaf node.

- rule $\perp$: If Γ^j_k contains (x, Φ) and $(x, \neg\Phi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \perp)\}$.

- rule $\neg$: If Γ^j_k contains $(x, \neg\neg\Phi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Phi)\}$.
- rule $\wedge$: If Γ^j_k contains $(x, \Phi \wedge \Phi')$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Phi), (x, \Phi')\}$.
- rule $\vee$: If Γ^j_k contains $(x, \neg(\Phi \wedge \Phi'))$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \neg\Phi)\}$ and node $\Gamma^{j'}_0 = \Gamma^j_k \cup \{(x, \neg\Phi')\}$, where j' is a fresh integer.
- rule $=$: If Γ^j_k contains $(x, c = c')$ and c and c' are distinct constants, or if Γ^j_k contains $(x, \neg(c = c'))$ and c and c' are identical constants, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \bot)\}$.
- rule $\Diamond\varphi$: If Γ^j_k contains $(0, \neg[\alpha]_0\varphi)$ or $(0, [\alpha]_q\varphi)$ for $q > 0$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \varphi)\}$, where x is a fresh integer.
- rule obs: If Γ^j_k contains $(x, \neg[\alpha]_0\varphi)$ or $(x, [\alpha]_q\varphi)$ for $q > 0$ and some x, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(0, \Box(\delta_1 \rightarrow (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \Box(\delta_2 \rightarrow (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \cdots \vee \Box(\delta_n \rightarrow (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)))\}$, where $\delta_i \in cpt(\varphi)$.
- rule $\Box$: If Γ^j_k contains $(0, \Box\Phi)$ and (x, Φ') for any $x \geq 0$, and if it does not yet contain (x, Φ), then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Phi)\}$.
- rule $\Diamond$: If Γ^j_k contains $(0, \neg\Box\Phi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \neg\Phi)\}$, where x is a fresh integer.

6.2.2 *Systems of Inequalities*

We first need to explain how a system of inequalities (SI) can be generated from a set of dynamic and perception literals, before the label assignment phase can be explained.

Definition 6.2.1: $W(\Gamma, x) \stackrel{def}{=} \{w \in C \mid w \models \ell$ for all $(x, \ell) \in \Gamma$ where ℓ is a propositional literal$\}$.

Definition 6.2.2: $X(\Gamma) \stackrel{def}{=} \{0, 1, \ldots, x'\}$ are all the labels mentioned in Γ.

Definition 6.2.3: $W(\Gamma) \stackrel{def}{=} \bigcup_{x \in \{0,1,\ldots,x'\}} W(\Gamma, x)$, where x' is the largest label in $X(\Gamma)$.

Let $n = |W(\Gamma)|$. Let $W(\Gamma)^\# = (w_1, w_2, \ldots, w_n)$ be an ordering of the worlds in $W(\Gamma)$. With each world $w_k \in W(\Gamma)^\#$, we associate a real variable $pr^\alpha_k \in \mathbb{Q}_{[0,1]}$. One can generate

$$c_{i,1}pr^\alpha_1 + c_{i,2}pr^\alpha_2 + \cdots + c_{i,n}pr^\alpha_n = q_i,$$

and

$$c_{i,1}pr^\alpha_1 + c_{i,2}pr^\alpha_2 + \cdots + c_{i,n}pr^\alpha_n \neq q_i,$$

for a formulae $(x, [\alpha]_{q_i}\varphi_i) \in \Gamma$, respectively $(x, \neg[\alpha]_{q_i}\varphi_i) \in \Gamma$ such that $c_{i,k} = 1$ if $w_k \models \varphi_i$, else $c_{i,k} = 0$, where x represents a label.

Let $\Delta(\alpha)$ be a set of dynamic literals mentioning α, and let

$$\Delta(\alpha)^\# = \{[\alpha]_{q_1}\varphi_1, [\alpha]_{q_2}\varphi_2, \ldots, [\alpha]_{q_g}\varphi_g, \neg[\alpha]_{q_{g+1}}\varphi_{g+1}, \neg[\alpha]_{q_{g+2}}\varphi_{g+2}, \ldots, \neg[\alpha]_{q_{g+h}}\varphi_{g+h}\}$$

be an ordering of the members of $\Delta(\alpha)$. With this notation in hand, given some α, we define the system

$$\begin{array}{rl}
c_{1,1}pr_1^\alpha + c_{1,2}pr_2^\alpha + \cdots + c_{1,n}pr_n^\alpha & = q_1 \\
c_{2,1}pr_1^\alpha + c_{2,2}pr_2^\alpha + \cdots + c_{2,n}pr_n^\alpha & = q_2 \\
\vdots & \\
c_{g,1}pr_1^\alpha + c_{g,2}pr_2^\alpha + \cdots + c_{g,n}pr_n^\alpha & = q_g \\
c_{g+1,1}pr_1^\alpha + c_{g+1,2}pr_2^\alpha + \cdots + c_{g+1,n}pr_n^\alpha & \neq q_{g+1} \\
c_{g+2,1}pr_1^\alpha + c_{g+2,2}pr_2^\alpha + \cdots + c_{g+2,n}pr_n^\alpha & \neq q_{g+2} \\
\vdots & \\
c_{g+h,1}pr_1^\alpha + c_{g+h,2}pr_2^\alpha + \cdots + c_{g+h,n}pr_n^\alpha & \neq q_{g+h} \\
pr_1^\alpha + pr_2^\alpha + \cdots + pr_n^\alpha & = \lceil pr_1^\alpha + pr_2^\alpha + \cdots + pr_n^\alpha \rceil,
\end{array} \tag{6.1}$$

where each of the first $g + h$ (in)equalities represents a member in $\Delta(\alpha)^\#$. The equation

$$pr_1^\alpha + pr_2^\alpha + \cdots + pr_n^\alpha = \lceil pr_1^\alpha + pr_2^\alpha + \cdots + pr_n^\alpha \rceil$$

is to ensure that either $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 1$ or $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 0$, as stated in Definition 6.1.2 on page 98.

Let $m = |\Omega|$. Let $\Omega^\# = (\varsigma_1, \varsigma_2, \ldots, \varsigma_m)$ be an ordering of the observations in Ω. With each observation in $\varsigma_j \in \Omega^\#$, we associate a real variable pr_j^ς.

One can generate

$$pr_j^\sigma = q_j \quad \text{and} \quad pr_j^\sigma \neq q_j$$

for a formula $(x, (\sigma_j \mid \alpha : q_j)) \in \Gamma$, respectively, $(x, \neg(\sigma_j \mid \alpha : q_j)) \in \Gamma$, where $\sigma_j \in \Omega^\#$ and $pr_j^\sigma \in \{pr_1^\varsigma, \ldots, pr_2^\varsigma, \ldots, pr_m^\varsigma\}$.

Let $\Omega(\alpha)$ be a set of perception literals involving α, and let

$$\Omega(\alpha)^\# = \{(\varsigma_1 \mid \alpha : q_1), \ldots, (\varsigma_t \mid \alpha : q_t), \neg(\varsigma_{t+1} \mid \alpha : q_{t+1}), \ldots, \neg(\varsigma_{t+v} \mid \alpha : q_{t+v})\}$$

be an ordering of the members of $\Omega(\alpha)$.

Then given some α, one can induce the following system.

$$\begin{array}{rl}
pr_1^\sigma & = q_1 \\
pr_2^\sigma & = q_2 \\
\vdots & \\
pr_t^\sigma & = q_t \\
pr_{t+1}^\sigma & \neq q_{t+1} \\
pr_{t+2}^\sigma & \neq q_{t+2} \\
\vdots & \\
pr_{t+v}^\sigma & \neq q_{t+v} \\
pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_2^\varsigma + \cdots + pr_m^\varsigma & = \lceil pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_2^\varsigma + \cdots + pr_m^\varsigma \rceil,
\end{array} \tag{6.2}$$

where each of the first $t + v$ (in)equalities represents a member in $\Omega(\alpha)^{\#}$. The equation

$$pr_1^{\varsigma} + pr_2^{\varsigma} + \cdots + pr_2^{\varsigma} + \cdots + pr_m^{\varsigma} = \lceil pr_1^{\varsigma} + pr_2^{\varsigma} + \cdots + pr_2^{\varsigma} + \cdots + pr_m^{\varsigma} \rceil$$

is to ensure that either $\sum_{o \in O, (o,w^+,q) \in Q_\alpha} q = 1$ or $\sum_{o \in O, (o,w^+,q) \in Q_\alpha} q = 0$, as stated in Definition 6.1.2 on page 98.

Let $S(\Delta(\alpha))$ and $S(\Omega(\alpha))$ be the systems formed from $\Delta(\alpha)$ (System 6.1), respectively, $\Omega(\alpha)$ (System 6.2). Let V be the set of all variables mentioned in $S(\Delta(\alpha))$ or $S(\Omega(\alpha))$.

Definition 6.2.4: $Z(\Delta(\alpha))$ and $Z(\Omega(\alpha))$ denote the solution set for $S(\Delta(\alpha))$, respectively, $S(\Omega(\alpha))$. It is the set of all solutions of the form $(s_1^{\alpha}, s_2^{\alpha}, \ldots, s_n^{\alpha})$, respectively, $(s_1^{\varsigma}, s_2^{\varsigma}, \ldots, s_m^{\varsigma})$, where assigning s_i^{α} to $pr_i^{\alpha} \in V$ for $i = 1, 2, \ldots, n$, respectively, assigning s_j^{ς} to $pr_j^{\varsigma} \in V$ for $j = 1, 2, \ldots, m$ solves all the (in)equalities in $S(\Delta(\alpha))$, respectively, $S(\Omega(\alpha))$ simultaneously. An SI is *feasible* if and only if its solution set is not empty.

Suppose $\Delta(\texttt{replace})$ contains

$$[\texttt{replace}]_{0.43}(\texttt{full} \wedge \neg\texttt{holding}) \quad \text{and} \quad \neg[\texttt{replace}]_{0.43}(\texttt{full} \wedge \neg\texttt{holding}).$$

Then $S(\Delta(\texttt{replace}))$ will contain

$$\begin{array}{ccccccccccccccc} 0 & + & pr_2^{\alpha} & + & 0 & + & pr_4^{\alpha} & + & 0 & + & 0 & + & 0 & = & 0.43 \\ 0 & + & pr_2^{\alpha} & + & 0 & + & pr_4^{\alpha} & + & 0 & + & 0 & + & 0 & \neq & 0.43. \end{array}$$

This system is clearly infeasible, and the whole system $S(\Delta(\texttt{replace}))$, of which this one is a subsystem, is, by extension, also infeasible. As will be seen in the next subsection, a node for which an infeasible system can be generated will be recognized as closed.

Suppose $\Omega(\texttt{weigh})$ contains $(\texttt{obsHeavy}|\texttt{weigh} : 0.56)$ and $(\texttt{obsHeavy}|\texttt{weigh} : 0.55)$. Then $S(\Omega(\texttt{weigh}))$ will contain

$$\begin{aligned} pr_4^{\varsigma} &= 0.56 \\ pr_4^{\varsigma} &= 0.55, \end{aligned}$$

where $\Omega^{\#} = \{\texttt{obsNil}, \texttt{obsLight}, \texttt{obsMedium}, \texttt{obsHeavy}\}$. This system is clearly infeasible, and thus also $S(\Omega(\texttt{weigh}))$.

Lemma 6.2.1: Determining whether an SI (as defined in this thesis) is feasible, is decidable.

Please find the proof in appendix Section A.3.

6.2.3 The Label-Assignment Phase

Given two formulae $(x, \Phi), (x', \Phi') \in \Gamma$ such that Φ contradicts Φ', if x and x' represent the same world, then Γ should close. But if $x \neq x'$, one must determine whether x and x' can be made to represent different worlds. In other words, one must check whether there is a 'proper' assignment of worlds to labels such that no contradictions occur.

Informally, $x \in X(\Gamma)$ could represent any one of the worlds in $W(\Gamma, x)$. Now suppose (x, Φ), $(x', \Phi') \in \Gamma$ such that Φ contradicts Φ' and $W(\Gamma, x) = \{w_1, w_2\}$ and $W(\Gamma, x') = \{w_2, w_3\}$. Assuming that Φ and Φ' do not involve the $\Box$ operator, it is conceivable that there exists a structure $\mathcal{S}$ such that (i) $\mathcal{S}, w_1 \models \Phi$ and $\mathcal{S}, w_2 \models \Phi'$, (ii) $\mathcal{S}, w_1 \models \Phi$ and $\mathcal{S}, w_3 \models \Phi'$ or (iii) $\mathcal{S}, w_2 \models \Phi$ and $\mathcal{S}, w_3 \models \Phi'$. But to have $\mathcal{S}, w_2 \models \Phi$ and $\mathcal{S}, w_2 \models \Phi'$ is inconceivable. Hence, if it were the case that, for example, $W(\Gamma, x) = \{w_2\}$ and $W(\Gamma, x') = \{w_2\}$, then we would have found a contradiction and Γ should be made closed.

To formalize the process, some more definitions are required:

Definition 6.2.5: $SoLA(\Gamma) \stackrel{def}{=} \{(0{:}w^0, 1{:}w^1, \ldots, x'{:}w^{x'}) \mid w^x \in W(\Gamma, x)$ and $x \in \{0, 1, \ldots, x'\} = X(\Gamma)\}$. We shall call an element of $SoLA(\Gamma)$ a *label assignment*. $LA(\Gamma)$ denotes an element of $SoLA(\Gamma)$. When it is clear that we are talking about an element of $SoLA(\Gamma)$, we simply write LA.

Definition 6.2.6: $E(\Gamma, x) \stackrel{def}{=} \{(x, \Phi) \in \Gamma \mid \Phi$ is $Reward(r)$ or $\neg Reward(r)$ or $Cost(\alpha, c)$ or $\neg Cost(\alpha, c)$ for some/any constants r and c and some/any action $\alpha\}$.

Definition 6.2.7: $E(\Gamma, LA, w) \stackrel{def}{=} \bigcup_{x:w \in LA(\Gamma)} E(\Gamma, x)$.

In natural language, $E(\Gamma, LA, w)$ is a set of formulae (as defined) in Γ with labels x such that the labels could logically represent world w, that is, such that $w \models \ell$ for all $(x, \ell) \in \Gamma$, and LA is one of the ways in which worlds can be assigned to labels mentioned in Γ.

Definition 6.2.8: $F(\Gamma, \alpha, x) \stackrel{def}{=} \{[\alpha]_q \varphi \mid (x, [\alpha]_q \varphi) \in \Gamma\} \cup \{\neg[\alpha]_q \varphi \mid (x, \neg[\alpha]_q \varphi) \in \Gamma\}$.

Definition 6.2.9: $F(\Gamma, \alpha, LA, w) \stackrel{def}{=} \bigcup_{x:w \in LA(\Gamma)} F(\Gamma, \alpha, x)$.

In natural language, $F(\Gamma, \alpha, LA, w)$ is the set of dynamic literals mentioning α in Γ with labels x such that the labels could logically represent world w.

Definition 6.2.10: $G(\Gamma, \alpha, x) \stackrel{def}{=} \{(\varsigma \mid \alpha : q) \mid (x, (\varsigma \mid \alpha : q)) \in \Gamma\} \cup \{\neg(\varsigma \mid \alpha : q) \mid (x, \neg(\varsigma \mid \alpha : q)) \in \Gamma\}$.

Definition 6.2.11: $G(\Gamma, \alpha, LA, w) \stackrel{def}{=} \bigcup_{x:w \in LA(\Gamma)} G(\Gamma, \alpha, x)$.

In natural language, $G(\Gamma, \alpha, LA, w)$ is the set of perception literals mentioning α in Γ with labels x such that the labels could logically represent world w.

After the tableau phase has completed, the label assignment phase begins. For each leaf node Γ^j_k of an open branch, do the following.

For every $LA \in SoLA(\Gamma^j_k)$, if one of the following two cases holds, then mark LA as UNSAT.

- For some $w \in W(\Gamma^j_k)$, $E(\Gamma^j_k, LA, w)$ contains
  - $Reward(r)$ and $Reward(r')$ such that $r \neq r'$, or
  - $Reward(r)$ and $\neg Reward(r)$, or
  - $Cost(\alpha, c)$ and $Cost(\alpha, c')$ (same action α) such that $c \neq c'$, or
  - $Cost(\alpha, c)$ and $\neg Cost(\alpha, c)$ (same action α).

- For some action $\alpha \in \mathcal{A}$ and some $w \in W(\Gamma_k^j)$, $Z(F(\Gamma_k^j, \alpha, LA, w)) = \emptyset$ or $Z(G(\Gamma_k^j, \alpha, LA, w)) = \emptyset$.

If every $LA \in SoLA(\Gamma_k^j)$ is marked as UNSAT, then create new leaf node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(0, \bot)\}$.

That is, if for all logically correct ways of assigning possible worlds to labels (i.e., for all the label assignments in $SoLA(\Gamma_k^j)$), no assignment (LA) satisfies all formulae in Γ_k^j, then Γ_k^j is unsatisfiable.

Definition 6.2.12: A tree is called *finished* after the label assignment phase is completed.

Note that all branches of a finished tree are saturated.

Definition 6.2.13: If a tree for $\neg\Psi$ is closed, we write $\vdash \Psi$. If there is a finished tree for $\neg\Psi$ with an open branch, we write $\nvdash \Psi$.

6.3 Properties of the Decision Procedure

All proofs not given here can be found in the appendix Section A.3.

6.3.1 Soundness

Lemma 6.3.1: Let Γ be the leaf node of a saturated tree. Suppose there exists a structure $\mathcal{S} = \langle W, R, O, N, Q, U\rangle$ such that $W = W(\Gamma)$, and for all $(x, \delta\omega) \in \Gamma$, where $\delta\omega$ is a dynamic or perception literal involving α, there exists a $w \in W(\Gamma)$ such that $\mathcal{S}, w \models \delta\omega$. Then there exists an $LA \in SoLA(\Gamma)$ such that for all $w \in W(\Gamma)$, $Z(F(\Gamma, \alpha, LA, w)) \neq \emptyset$ and $Z(G(\Gamma, \alpha, LA, w)) \neq \emptyset$.

Lemma 6.3.2: Let T be a finished tree. For every node Γ in T: If there exists a structure $\mathcal{S}$ such that for all $(x, \Phi) \in \Gamma$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$, then the (sub)tree rooted at Γ is open.

Theorem 6.3.1: (Soundness) If $\vdash \Psi$ then $\models \Psi$. (Contrapositively, if $\not\models \Psi$ then $\nvdash \Psi$.)

6.3.2 Completeness

We start with the description of the construction of a SLAOP structure, given the leaf node Γ of some open branch of a finished tree. $\mathcal{S} = \langle W, R, O, N, Q, U\rangle$ can be constructed as follows:

- Let $W = W(\Gamma)$.
- For every action $\alpha \in \mathcal{A}$, the accessibility relation R_α can be constructed as follows. Let $R_\alpha(w, w_j) = s_j^\alpha \iff$
  - $w \in W$,

– $w_j \in W(\Gamma)^{\#}$,

– $(s_1^{\alpha}, s_2^{\alpha}, \ldots, s_n^{\alpha}) \in Z(F(\Gamma, \alpha, LA, w))$.

In other words, for every $w \in W$, determine which labels represent it according to the label assignment LA and then use any solution $(s_1^{\alpha}, s_2^{\alpha}, \ldots, s_n^{\alpha})$ for the SI generated from $F(\Gamma, \alpha, LA, w)$ to assign the transition probabilities, where s_1^{α} is the probability of reaching world w_1 from world w, s_2^{α} is the probability of reaching world w_2 from world w, and so on until s_n^{α}/w_n.

- Let $O = \Omega$.
- Let $N = \{(o, o) \mid o \in O\}$.
- For every action $\alpha \in \mathcal{A}$, the perceivability relation Q_{α} can be constructed as follows. Recall that $\Omega^{\#} = (\varsigma_1, \varsigma_2, \ldots, \varsigma_m)$ is an ordering of Ω. Let $Q_{\alpha}(w_j, N(\varsigma_j)) = s_j^{\varsigma} \iff$

  – $\varsigma_j \in \Omega^{\#}$,

  – $w_j \in W(\Gamma)^{\#}$,

  – $(s_1^{\varsigma}, s_2^{\varsigma}, \ldots, s_m^{\varsigma}) \in Z(G(\Gamma_k, \alpha, LA, w_j))$,

  In other words, for every $w \in W$, determine which labels represent it according to the label assignment LA and then use any solution $(s_1^{\varsigma}, s_2^{\varsigma}, \ldots, s_m^{\varsigma})$ for the SI generated from $G(\Gamma, \alpha, LA, w)$ to assign the perception probabilities, where s_1^{ς} is the probability of perceiving ς_1 in world w, s_2^{ς} is the probability of perceiving ς_2 in w, and so on until s_m^{ς}.
- If there is $(x, \neg Reward(r)) \in \Gamma$ for some x and r, then let $maxRew(\Gamma) = \max_r\{(x, \neg Reward(r)) \in \Gamma\}$, else, let $maxRew(\Gamma) = 0$. For each $(x, Reward(r)) \in \Gamma$, let $Re(w) = r$, where $x{:}w \in X(\Gamma)$. For all $w \in W(\Gamma)$, if it is not the case that $(x, Reward(r)) \in \Gamma$, where $x{:}w \in X(\Gamma)$, then let $Re(w) = maxRew(\Gamma) + 1$. If there is $(x, \neg Cost(\alpha, c)) \in \Gamma$ for some x, α and c, then let $maxCost(\Gamma) = \max_c\{(x, \neg Cost(\alpha, c)) \in \Gamma\}$, else, let $maxCost(\Gamma) = 0$. For each $(x, Cost(\alpha, c)) \in \Gamma$, let $Co_{\alpha}(w) = c$, where $x{:}w \in X(\Gamma)$. For all $w \in W(\Gamma)$, if it is not the case that $(x, Cost(\alpha, c)) \in \Gamma$, where $x{:}w \in X(\Gamma)$, then let $Co_{\alpha}(w) = maxCost(\Gamma) + 1$. Let $U = \langle Re, Co\rangle$ such that $Co = \{(\alpha, Co_{\alpha}) \mid \alpha \in \mathcal{A}\}$.

Lemma 6.3.3: $\mathcal{S}$ is a SLAOP structure.

W.l.o.g., one can assume that, for every $(x, \Box\Phi) \in \Gamma$, Φ is in DNF.

Lemma 6.3.4: Let Γ be the leaf node of a finished tree, where $(0, \Box\Phi) \in \Gamma$, for some $\Box\Phi \in \mathcal{L}_{SLAOP}$. For every label $x \in X(\Gamma)$, there exists a term $(\Phi_{k1} \wedge \Phi_{k2} \wedge \cdots \wedge \Phi_{km_k})$ of Φ such that (x, Φ_{k1}), (x, Φ_{k2}), …, $(x, \Phi_{km_k}) \in \Gamma$.

Proof:
The same as the proof of Lemma 5.3.3. ■

Lemma 6.3.5: Let Γ be the leaf node of an open branch of a finished tree. We know that there exists a label assignment $LA \in SoLA(\Gamma)$ such that $Z(F(\Gamma, \alpha, LA, w))$ and $Z(G(\Gamma, \alpha, LA, w))$ are not empty, for all $w \in W(\Gamma)$ and all $\alpha \in \mathcal{A}$. If $\mathcal{S}$ is constructed as described above, then for all $(x, \Psi) \in \Gamma$, $\mathcal{S}, w \models \Psi$ for $x{:}w \in LA$.

Theorem 6.3.2: (Completeness) If $\models \Psi$ then $\vdash \Psi$. (Contrapositively, if $\nvdash \Psi$ then $\not\models \Psi$.)

6.3.3 Termination

Lemma 6.3.6: A tree for any formula $\Phi \in \mathcal{L}_{SLAOP}$ becomes saturated. That is, the tableau phase terminates.

Proof:
We can divide all the tableau rules into three categories: (i) those which add $\perp$ to the new node, (ii) those with the subformula property and (iii) rule obs. Category-(i) rules never cause rules to become applicable later. As a direct consequence of sentences being finite and the subformula property, every category-(ii) rule must eventually become inapplicable. Rule obs is reproduced here: If Γ^j_k contains $(x, \neg[\alpha]_0\varphi)$ or $(x, [\alpha]_q\varphi)$ for $q > 0$ and some x, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(0, \Box(\delta_1 \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)) \vee \Box(\delta_2 \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)) \vee \cdots \vee \Box(\delta_n \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)))\}$, where $\delta_i \in cpt(\varphi)$. Note that $\Box(\delta_1 \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)) \vee \Box(\delta_2 \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)) \vee \cdots \vee \Box(\delta_n \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0))$ is not dynamic; it can thus not make rule obs applicable. That is, rule obs can only cause category-(i) and category-(ii) rules to become applicable.

Therefore, all rules eventually become inapplicable, and it follows that any tree (for any formula) would become saturated. ■

Theorem 6.3.3: The decision procedure for SLAOP terminates.

Proof:
Due to Lemma 6.3.6, the tableau phase terminates (with a finite number of branches).

In the label assignment phase: For every leaf node Γ of an open branch, for every $LA \in SoLA(\Gamma)$, two cases are checked: (i) whether $E(\Gamma, LA, w)$ is satisfiable for all $w \in W(\Gamma)$ and (ii) whether there exists a solution set for an SI; once for each action in $\mathcal{A}$ for each world in $W(\Gamma)$.

$W(\Gamma)$ and $X(\Gamma)$ are both finite and thus $SoLA(\Gamma)$ is finite. $E(\Gamma, LA, w)$ is finite and so is $\mathcal{A}$. By Lemma 6.2.1, finding the solution set for an SI is decidable as used in the label assignment phase and the process thus terminates in this phase. ■

Corollary 6.3.1: The entailment problem for SLAOP is decidable.

Because the procedure is sound (Th. 6.3.1), complete (Th. 6.3.2) and terminating (Th. 6.3.3), entailment is decidable.

6.4 Specifying Domains with SLAOP

A straightforward approach to specifying a stochastic, partially observable domain or environment will be presented in this section. In each case, the general approach will be presented, followed by an example based on the oil-drinking scenario. A static law is any sentence which does not contain reward, cost, dynamic or perception literals. Otherwise, there is no particular structure to a static law; they will thus not be discussed further in this chapter.

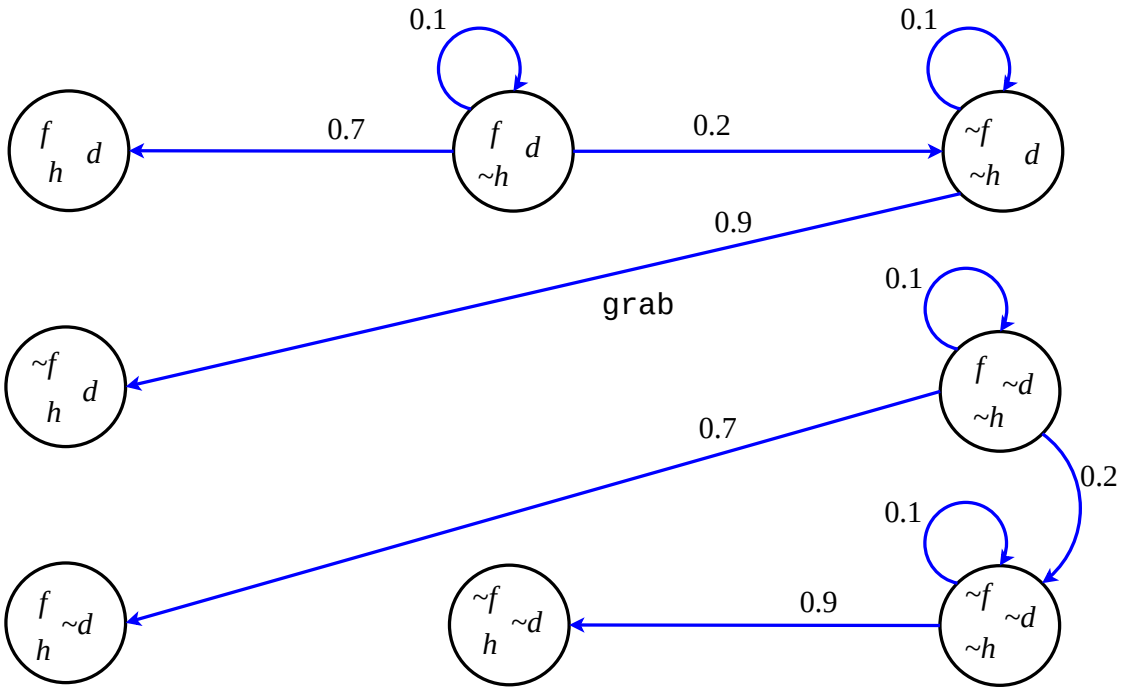


Fig. 6.1: A transition diagram for the `grab` action.

6.4.1 Action Rules

Action rules correspond to R of SLAOP structures and T of POMDP models. As also for SLAP, three kinds of axioms make up the action rules: For every action $\alpha \in \mathcal{A}$, one requires *effect axioms* and *inexecutability axioms*.

Figures 6.1, 6.2, 6.3 and 6.4 are pictorial representations of transitions and their probabilities for the actions `grab` (g), `drink` (di), `replace` (r) and `weigh` (w) of the oil-drinking scenario. The eight circles represent the eight conceivable worlds with their valuations. The letters f, d and h represent, respectively, propositional literals `full`, `drank` and `holding`. And $\sim$ reads 'not'.

Effect Axioms

Effect axioms should take the form

$$\begin{aligned}
\phi_1 &\to [\alpha]_{p_{11}}\varphi_{11} \wedge \cdots \wedge [\alpha]_{p_{1n}}\varphi_{1n} \\
\phi_2 &\to [\alpha]_{p_{21}}\varphi_{21} \wedge \cdots \wedge [\alpha]_{p_{2n}}\varphi_{2n} \\
&\vdots \\
\phi_j &\to [\alpha]_{p_{j1}}\varphi_{j1} \wedge \cdots \wedge [\alpha]_{p_{jn}}\varphi_{jn},
\end{aligned}$$

where (i) no $p_{ik} = 0$, (ii) the transition probabilities $p_{i1}, \ldots, p_{in}$ of any axiom i must sum to 1, (iii) for every i, for any pair of effects φ_{ik} and $\varphi_{ik'}$, $\varphi_{ik} \wedge \varphi_{ik'} \equiv \bot$ and (iv) for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$.

Effect axioms of actions `grab`, `drink`, `replace` and `weigh` follow.

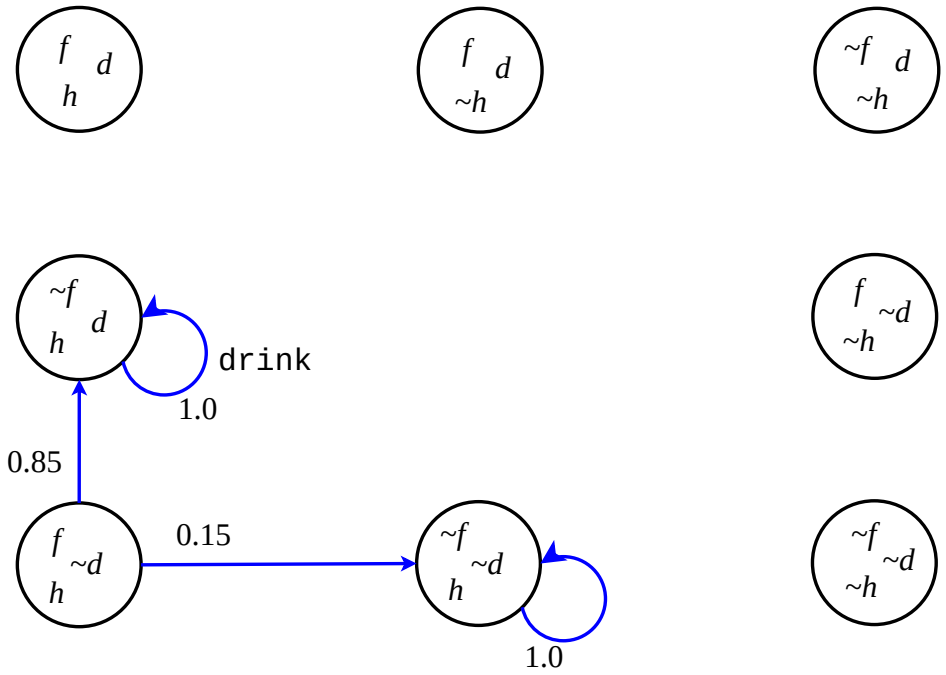


Fig. 6.2: A transition diagram for the `drink` action.

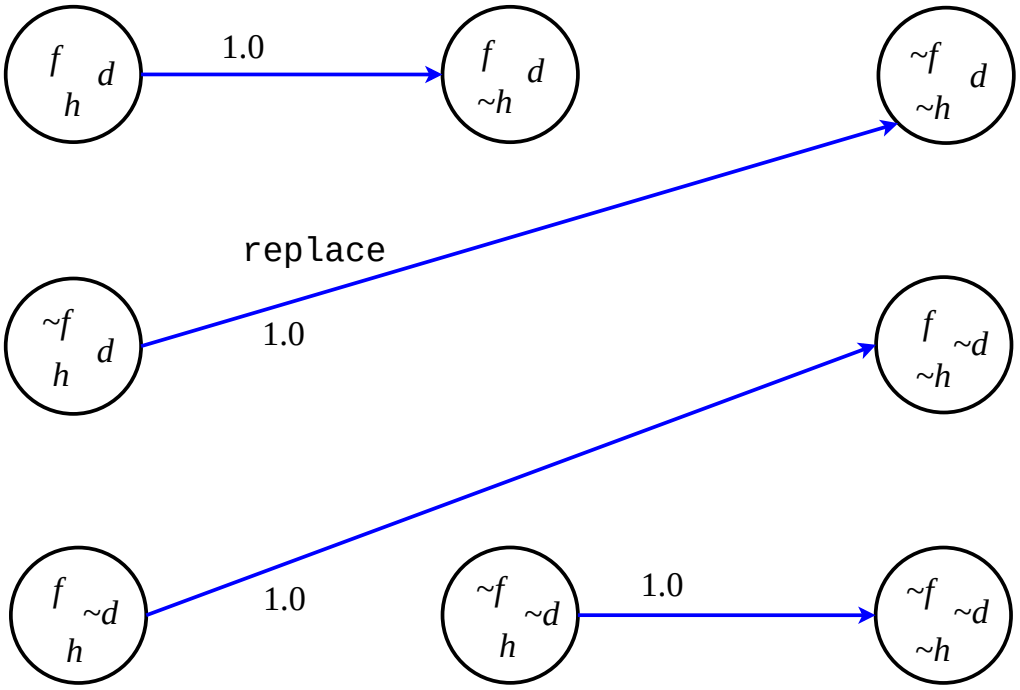


Fig. 6.3: A transition diagram for the `replace` action.

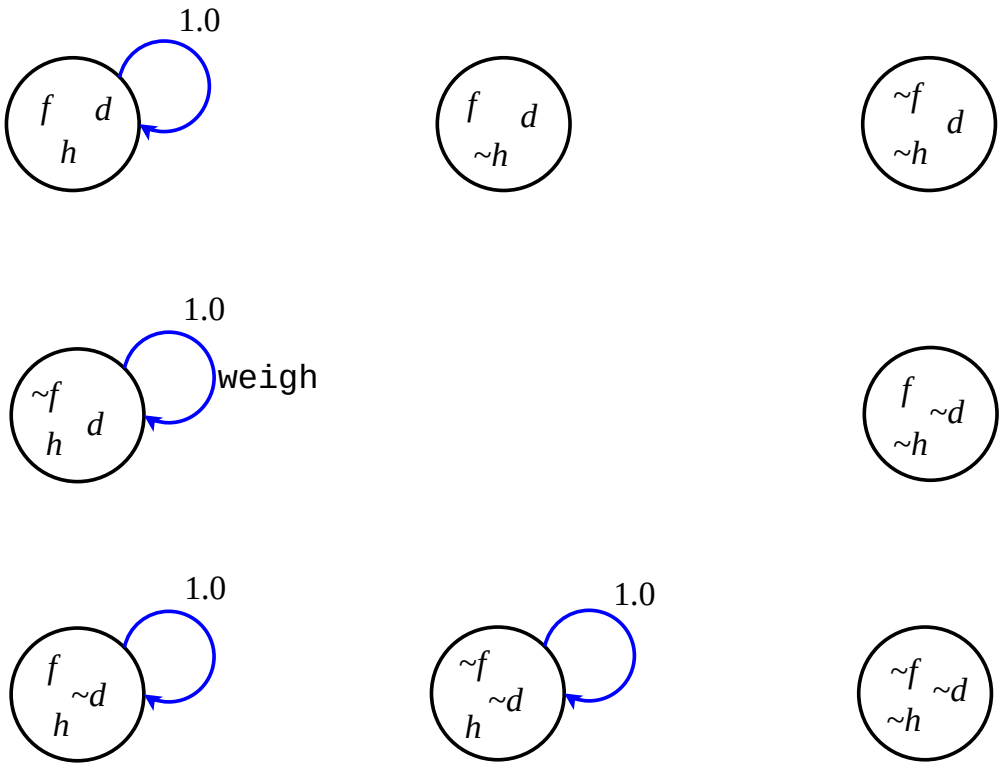


Fig. 6.4: A transition diagram for the `weigh` action.

$$
\begin{aligned}
f \wedge d \wedge \neg h &\rightarrow [g]_{0.7}(f \wedge d \wedge h) \wedge [g]_{0.2}(\neg f \wedge d \wedge \neg h) \wedge [g]_{0.1}(f \wedge d \wedge \neg h);\\
f \wedge \neg d \wedge \neg h &\rightarrow [g]_{0.7}(f \wedge \neg d \wedge h) \wedge [g]_{0.2}(\neg f \wedge \neg d \wedge \neg h) \wedge [g]_{0.1}(f \wedge \neg d \wedge \neg h);\\
\neg f \wedge d \wedge \neg h &\rightarrow [g]_{0.9}(\neg f \wedge d \wedge h) \wedge [g]_{0.1}(\neg f \wedge d \wedge \neg h);\\
\neg f \wedge \neg d \wedge \neg h &\rightarrow [g]_{0.9}(\neg f \wedge \neg d \wedge h) \wedge [g]_{0.1}(\neg f \wedge \neg d \wedge \neg h).
\end{aligned}
$$

$$
\begin{aligned}
f \wedge \neg d \wedge h &\rightarrow [di]_{0.85}(\neg f \wedge d \wedge h) \wedge [di]_{0.15}(\neg f \wedge \neg d \wedge h);\\
\neg f \wedge d \wedge h &\rightarrow [di](\neg f \wedge d \wedge h);\\
\neg f \wedge \neg d \wedge h &\rightarrow [di](\neg f \wedge h).
\end{aligned}
$$

$$
\begin{aligned}
f \wedge d \wedge h &\rightarrow [r](f \wedge d \wedge \neg h);\\
f \wedge \neg d \wedge h &\rightarrow [r](f \wedge \neg d \wedge \neg h);\\
\neg f \wedge d \wedge h &\rightarrow [r](\neg f \wedge d \wedge \neg h);\\
\neg f \wedge \neg d \wedge h &\rightarrow [r](\neg f \wedge \neg d \wedge \neg h).
\end{aligned}
$$

$$\begin{aligned} f \wedge d \wedge h &\rightarrow [w](f \wedge d \wedge h); \\ f \wedge \neg d \wedge h &\rightarrow [w](f \wedge \neg d \wedge h); \\ \neg f \wedge d \wedge h &\rightarrow [w](\neg f \wedge d \wedge h); \\ \neg f \wedge \neg d \wedge h &\rightarrow [w](\neg f \wedge \neg d \wedge h). \end{aligned}$$

Inexecutability Axioms

Suppose there are j effect axioms for action α, with conditions $\phi_1, \phi_2, \ldots, \phi_j$. Then, assuming that effect axioms are meant to say all there is to be said about actions (the completeness assumption), we want to express that if a world does not satisfy one of the j conditions, then it is not possible to execute. This can be written as

$$\neg(\phi_1 \vee \phi_2 \vee \cdots \vee \phi_j) \rightarrow \neg\langle\alpha\rangle\top$$

or

$$\langle\alpha\rangle\top \rightarrow (\phi_1 \vee \phi_2 \vee \cdots \vee \phi_j).$$

Often $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_j$ has a compact equivalent form.

The inexecutability axioms of the four actions follow.

$$\begin{aligned} \langle\texttt{grab}\rangle\top &\rightarrow \neg\texttt{holding}; \\ \langle\texttt{drink}\rangle\top &\rightarrow \texttt{holding} \wedge \neg(\texttt{full} \wedge \texttt{drank}); \\ \langle\texttt{replace}\rangle\top &\rightarrow \texttt{holding}; \\ \langle\texttt{weigh}\rangle\top &\rightarrow \texttt{holding}. \end{aligned}$$

Recall that inexecutability axioms can also be called *condition closure axioms* because inexecutability axioms are derived from the *closure* of the *conditions* of actions' executability.

6.4.2 Perception Rules

Perception rules correspond to Q of SDL structures and O of POMDP models.

Let $E(\alpha) = \{\varphi_{11}, \varphi_{12}, \ldots, \varphi_{21}, \varphi_{22}, \ldots, \varphi_{jn}\}$ be the set of all effects of action α executed under all executable conditions. For every action α, perception rules typically take the form

$$\begin{aligned} \phi_1 &\rightarrow (\varsigma_{11} \mid \alpha : p_{11}) \wedge \cdots \wedge (\varsigma_{1m} \mid \alpha : p_{1m}) \\ \phi_2 &\rightarrow (\varsigma_{21} \mid \alpha : p_{21}) \wedge \cdots \wedge (\varsigma_{2m} \mid \alpha : p_{2m}) \\ &\vdots \\ \phi_k &\rightarrow (\varsigma_{k1} \mid \alpha : p_{k1}) \wedge \cdots \wedge (\varsigma_{km} \mid \alpha : p_{km}), \end{aligned}$$

where (i) the sum of perception probabilities $p_{i1}, \ldots, p_{im}$ of any rule i must sum to 1, (ii) for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$ and (iii) $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_k \equiv \bigvee_{\varphi \in E(\alpha)} \varphi$.

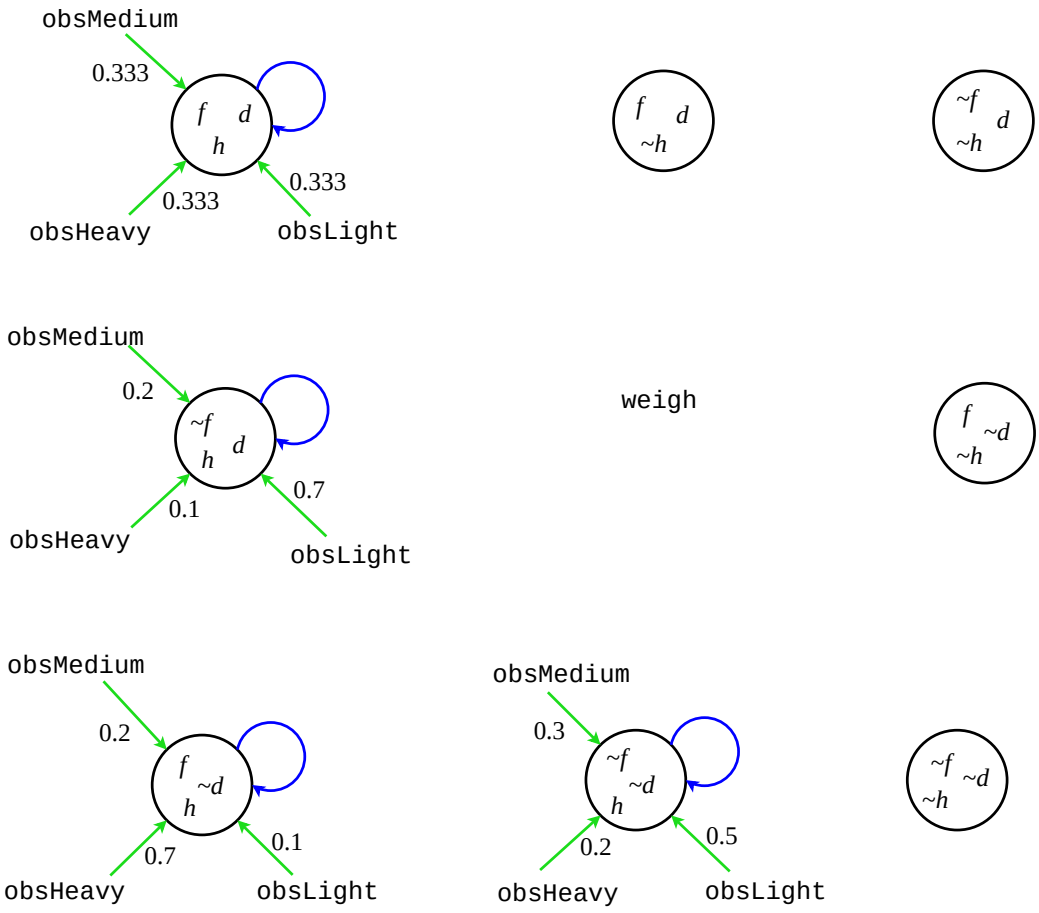


Fig. 6.5: A transition diagram for the `weigh` action.

In the oil-drinking scenario, the robot perceives only `obsNil` when it executes `grab`, `drink` or `replace`, because they are ontic actions. Their perception rules are simply

$$\begin{aligned} \top &\rightarrow (\texttt{obsNil} \mid \texttt{grab} : 1) \\ h \wedge \neg(f \wedge d) &\rightarrow (\texttt{obsNil} \mid \texttt{drink} : 1) \\ \neg h &\rightarrow (\texttt{obsNil} \mid \texttt{replace} : 1). \end{aligned}$$

Action `weigh` is a sensory action. It does not have the simple form of ontic actions. The following abbreviations for observation constants will be used: $\texttt{obsHeavy} := oH$, $\texttt{obsMedium} := oM$ and $\texttt{obsLight} := oL$. The perception rules capturing the meaning of Figure 6.5 are

$$\begin{aligned} f \wedge d \wedge h &\rightarrow (oL \mid w : 0.\bar{3}) \wedge (oM \mid w : 0.\bar{3}) \wedge (oH \mid w : 0.\bar{3}) \\ f \wedge \neg d \wedge h &\rightarrow (oL \mid w : 0.1) \wedge (oM \mid w : 0.2) \wedge (oH \mid w : 0.7) \\ \neg f \wedge d \wedge h &\rightarrow (oL \mid w : 0.7) \wedge (oM \mid w : 0.2) \wedge (oH \mid w : 0.1) \\ \neg f \wedge \neg d \wedge h &\rightarrow (oL \mid w : 0.5) \wedge (oM \mid w : 0.3) \wedge (oH \mid w : 0.2) \end{aligned}$$

A perception axiom/rule of the form

$$\phi_i \rightarrow \cdots \wedge (\varsigma \mid \alpha : 0) \wedge \cdots$$

implies that ς is unperceivable in a ϕ_i-world given that the world is reachable via α. Likewise, if $\varsigma \notin \{\varsigma_{i1}, \ldots, \varsigma_{im}\}$ and

$$\phi_i \rightarrow (\varsigma_{i1} \mid \alpha : p_{i1}) \wedge \cdots \wedge (\varsigma_{im} \mid \alpha : p_{im})$$

is a perception rule, then ς is unperceivable in a ϕ_i-world given that the world is reachable via α.

6.4.3 Utility Rules

The rewards and costs of actions in different states must be specified. Refer to these *utility rules* as UR. Utility rules correspond to U of SLAOP structures and R of POMDP models. Utility rules typically take the form

$$\begin{aligned} \phi_1 &\rightarrow Reward(r_1) \\ \phi_2 &\rightarrow Reward(r_2) \\ &\vdots \\ \phi_j &\rightarrow Reward(r_j), \end{aligned}$$

meaning that in all worlds where ϕ_i is satisfied, the agent gets r_i units of reward. And for every action α,

$$\begin{aligned} \phi_1 &\rightarrow Cost(\alpha, r_1) \\ \phi_2 &\rightarrow Cost(\alpha, r_2) \\ &\vdots \\ \phi_j &\rightarrow Cost(\alpha, r_j), \end{aligned}$$

meaning that the cost for performing α in a world where ϕ_i is satisfied is r_i units. The conditions are disjoint as for action and perception rules.

6.5 Using Entailment in SLAOP

Recall that we denote the action rules as AR, perception rules as PR, utility rules as UR and static laws as SL. Let the agent's background knowledge be denoted BK. We define BK to be $AR \cup PR \cup UR \cup SL$.

Given some initial condition IC, an agent can ask whether some arbitrary sentence $\Psi \in \mathcal{L}_{SLAOP}$ follows from or is entailed by its background knowledge in the initial condition. That is, an agent can query whether

$$\{\Box\beta \mid \beta \in BK\} \models IC \rightarrow \Psi$$

holds, where IC is a propositional sentence, the sentences in BK contain no $\Box$ operator, and there is no restriction on Ψ.

Two examples based on the oil-drinking scenario are presented next. Suppose the following *domain axioms*[3] are part of the robot's background knowledge BK for the oil-drinking scenario. For brevity, we use a condensed subset of the full specification given in the previous section.

$$
\begin{aligned}
&f \wedge d \wedge h \rightarrow (\forall v^{\varsigma})(v^{\varsigma} \mid w : 0.\bar{3}) \\
&f \wedge \neg d \wedge h \rightarrow (oL \mid w : 0.1) \wedge (oH \mid w : 0.7) \\
&((f \wedge \neg d) \vee (\neg f \wedge d)) \wedge h \rightarrow (oM \mid w : 0.2) \\
&\neg f \wedge d \wedge h \rightarrow (oL \mid w : 0.7) \wedge (oH \mid w : 0.1) \\
&\neg f \wedge \neg d \wedge h \rightarrow (oL \mid w : 0.5) \wedge (oM \mid w : 0.3) \wedge (oH \mid w : 0.2) \\
&\neg f \wedge \neg h \rightarrow [g]_{0.9} h \wedge [g]_{0.1} \neg h \wedge [g] \neg f.
\end{aligned}
$$

The fact that one can do meaningful reasoning with an incomplete specification, also shows the flexibility of the logic.

In Figures 6.6 and 6.7, the vertices represent nodes and the arcs represent the application of tableau rules. Arcs are labeled with the rule they represent, except when branching occurs, in which case, the $\vee$ rule was applied. The figures show how the vertices relate to the corresponding nodes. The reader should keep in mind that the node corresponding to a vertex v contains all the labeled formulae in vertices above v on the same branch—the vertices show only the elements of nodes which are 'added' to a node due to the application of some rule. An exception is the top vertex of a tree, which is the trunk and not the result of any rule application.

In order to show the development of the tree, some liberties were taken with respect to rule application: In some cases, rule application is not shown, that is, from parent node to child node, a formula may be 'processed' more than is possible by the application of the rule represented by the arc from parent to child in the figure. The arc labeled "nf" denotes *normal forming*: translating abbreviations into symbols in the language.

For the first example, we claim that

$$\{\Box\beta \mid \beta \in BK\} \models \neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h). \tag{6.3}$$

Figure 6.6 shows only one branch of a tree for

$$\bigwedge_{\beta \in BK} \Box\beta \wedge \neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h)). \tag{6.4}$$

For the claim (Statement 6.3) to hold, the tree for (6.4) must close. We shall only show that the branch in Figure 6.6 closes. The leaf node of the branch is open and must thus be considered in the label assignment phase.

For clarity, denote w_1 as 111 where $w_1 \models f \wedge d \wedge h$, w_2 as 110 where $w_2 \models f \wedge d \wedge \neg h$, $\ldots$, w_8

[3] Only the last of these sentences can be expressed in SLAP. Notice the compact representation of the perception probabilities in the first sentence, due to quantification.

$$\Gamma_0^0 = \{(0, \bigwedge_{\beta \in BK} \Box\beta \wedge \neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h))\}$$

$\downarrow$ nf

$$(0, \neg\neg(\neg f \wedge d \wedge \neg h) \wedge \neg[g]_{0.9}(\neg f \wedge h)) \in \Gamma_1^0$$

$\downarrow \wedge$

$$(0, \neg\neg(\neg f \wedge d \wedge \neg h)), (0, \neg[g]_{0.9}(\neg f \wedge h)) \in \Gamma_2^0$$

$\downarrow \neg$

$$(0, \neg f \wedge d \wedge \neg h) \in \Gamma_3^0$$

$\downarrow \wedge$

$$(0, \neg f), (0, d), (0, \neg h) \in \Gamma_4^0$$

$\downarrow \Box$

$$(0, \neg(\neg f \wedge \neg h) \vee ([g]_{0.9}h \wedge [g]_{0.1}\neg h \wedge [g]\neg f)) \in \Gamma_5^0$$

$\downarrow \vee$

$$(0, ([g]_{0.9}h \wedge [g]_{0.1}\neg h \wedge [g]\neg f) \in \Gamma_0^3$$

$\downarrow \wedge$

$$(0, [g]_{0.9}h), (0, [g]_{0.1}\neg h), (0, [g]\neg f) \in \Gamma_1^3$$

$\downarrow \Diamond\varphi$

$$(1, h), (2, \neg h), (3, \neg f) \in \Gamma_2^3$$

$\downarrow \Box$

$$\begin{gathered}(1, f \vee h \vee ([g]_{0.9}h \wedge [g]_{0.1}\neg h)), (1, f \vee h \vee [g]\neg f),\\ (2, f \vee h \vee ([g]_{0.9}h \wedge [g]_{0.1}\neg h)), (2, f \vee h \vee [g]\neg f),\\ (3, f \vee h \vee ([g]_{0.9}h \wedge [g]_{0.1}\neg h)), (3, f \vee h \vee [g]\neg f) \in \Gamma_3^3\end{gathered}$$

$\downarrow \vee$

$$\begin{gathered}(1, [g]_{0.9}h), (1, [g]_{0.1}\neg h), (1, [g]\neg f), (2, [g]_{0.9}h),\\ (2, [g]_{0.1}\neg h), (2, [g]\neg f), (3, [g]_{0.9}h), (3, [g]_{0.1}\neg h),\\ (3, [g]\neg f) \in \Gamma_0^8\end{gathered}$$

Fig. 6.6: One branch of a tree for proving that $\{\Box\beta \mid \beta \in BK\}$ entails $\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h)$.

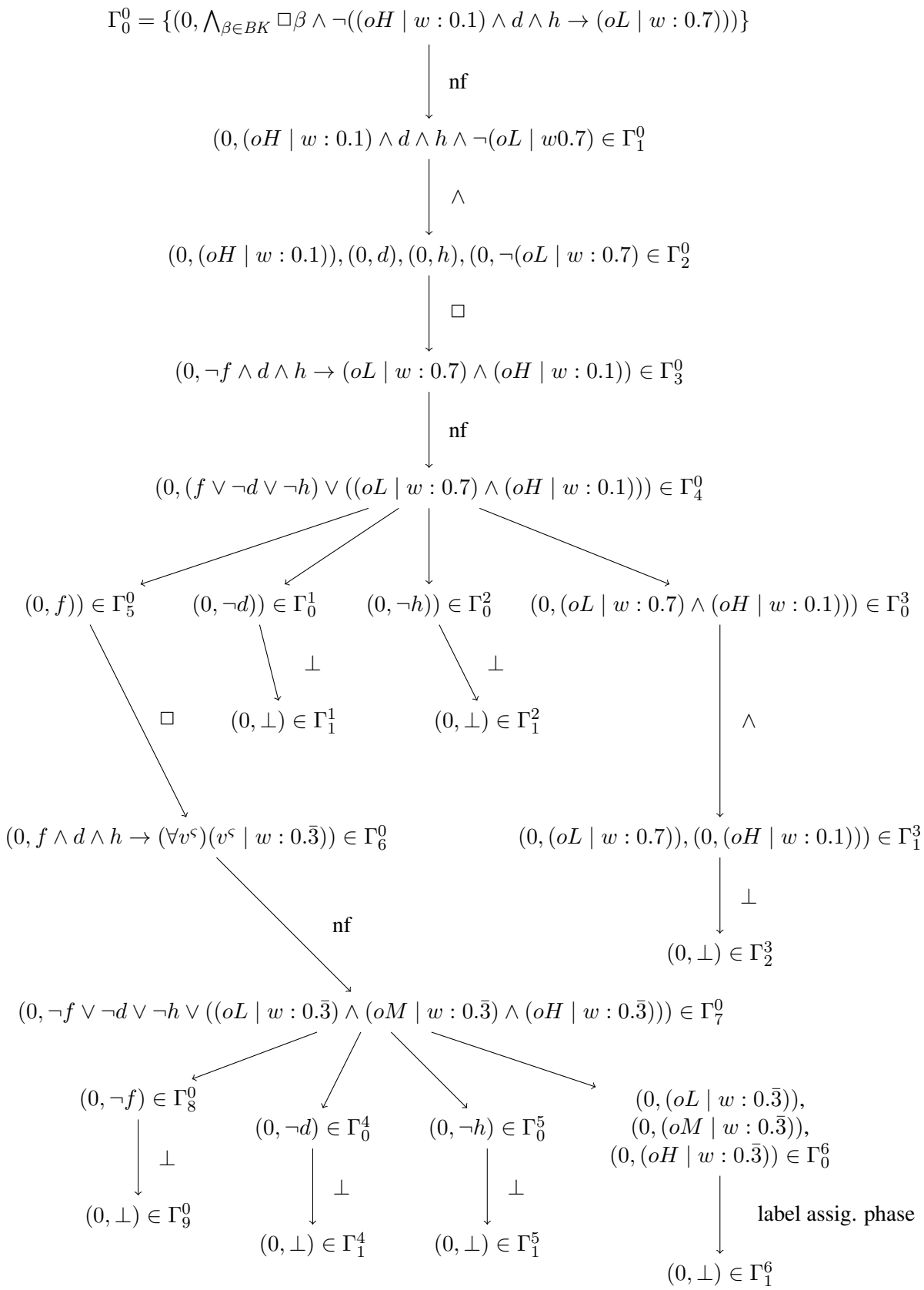


Fig. 6.7: A tree for proving that $\{\Box\beta \mid \beta \in BK\}$ entails $(oH \mid w : 0.1) \wedge d \wedge h \rightarrow (oL \mid w : 0.7)$.

as 000 where $w_8 \models \neg f \wedge \neg d \wedge \neg h$. We shall refer to the leaf node as Γ. Observe that

- $W(\Gamma, 0) = \{010\}$,
- $W(\Gamma, 1) = \{111, 101, 011, 001\}$,
- $W(\Gamma, 2) = \{110, 100, 010, 000\}$,
- $W(\Gamma, 3) = \{011, 010, 001, 000\}$.

and that $W(\Gamma) = \{111, 101, 011, 001, 110, 100, 010, 000\} = C$. Observe that 0:010 is in every label assignment in $SoLA(\Gamma)$.

Note that $F(\Gamma, \texttt{grab}, 0) \subseteq F(\Gamma, \texttt{grab}, LA, 010)$ for all $LA \in SoLA(\Gamma)$. And note that $F(\Gamma, \texttt{grab}, 0)$ equals

$$\{[\texttt{grab}]_{0.9}\texttt{holding}, [\texttt{grab}]_{0.1}\neg\texttt{holding}, [\texttt{grab}]\neg\texttt{full}, \neg[\texttt{grab}]_{0.9}(\neg\texttt{full} \wedge \texttt{holding})\}.$$

The system generated from $F(\Gamma, \texttt{grab}, 0)$ is

$$\begin{array}{ccccccccccccccc}
0 & + & 0 & + & 0 & + & 0 & + & pr_5^\alpha & + & 0 & + & pr_7^\alpha & + & 0 & = & 0.9 \\
0 & + & pr_2^\alpha & + & 0 & + & pr_4^\alpha & + & 0 & + & pr_6^\alpha & + & 0 & + & pr_8^\alpha & = & 0.1 \\
0 & + & 0 & + & 0 & + & 0 & + & pr_5^\alpha & + & pr_6^\alpha & + & pr_7^\alpha & + & pr_8^\alpha & = & 1 \\
pr_1^\alpha & + & 0 & + & pr_3^\alpha & + & 0 & + & pr_5^\alpha & + & 0 & + & pr_7^\alpha & + & 0 & \neq & 0.9 \\
pr_1^\alpha & + & pr_2^\alpha & + & pr_3^\alpha & + & pr_4^\alpha & + & pr_5^\alpha & + & pr_6^\alpha & + & pr_7^\alpha & + & pr_8^\alpha & = & 1.
\end{array}$$

Due to $pr_5^\alpha + pr_6^\alpha + pr_7^\alpha + pr_8^\alpha = 1$ (3rd equation), it must be the case that $pr_5^\alpha + pr_7^\alpha \neq 0.9$ (4th inequation). But it is required by the first equation that $pr_5^\alpha + pr_7^\alpha = 0.9$, which forms a contradiction. Thus, for every label assignment, there exists an action and a world w—that is, 010—for which $Z(F(\Gamma, \texttt{grab}, LA, w) = \emptyset$ and the branch closes.

For the second example, we claim that $\{\Box\beta \mid \beta \in BK\} \models (oH \mid w : 0.1) \wedge d \wedge h \rightarrow (oL \mid w : 0.7)$. Figure 6.7 shows the closed tree for

$$\bigwedge_{\beta \in BK} \Box\beta \wedge \neg((oH \mid w : 0.1) \wedge d \wedge h \rightarrow (oL \mid w : 0.7)).$$

The arc labelled "label assig. phase" means that for all label assignments, the SI generated for a set of formulae will include $(oH \mid w : 0.1)$ and $(oH \mid w : 0.\bar{3})$, which will cause all SIs to be infeasible. Hence, the label assignment phase will create a new node containing $(0, \bot)$ at the end of the branch.

6.6 Concluding Remarks

A decidable logic with a semantics closely related to partially observable Markov decision processes (POMDPs) was presented. The logic is a step towards the definition of a logic for reasoning about an agent's belief-states and expected future rewards, where the agent's actions and observations are stochastic.

Predicate $(\varsigma \mid \alpha : q)$ is useful for specifying the probability of perceiving an observation in the 'current' world. However, it would be useful to query the probability q of ending in a φ-world after executing action α in the 'current' world and then perceiving ς in the φ-world. To make such queries possible, one could add a modal operator with the following definition.
$\mathcal{S}, w \models [\alpha + \varsigma : q]\varphi \iff \sum_{w' \in W, \mathcal{S}, w' \models \varphi} R_\alpha(w, w') \times Q_\alpha(w', N(\varsigma)) \geq q.$

Informally, sentences of the form $[\alpha]_q\varphi$ and $(\varsigma \mid \alpha : q)$ have a meaning 'probability is exactly q.' In our Stochastic Decision Logic (SDL), to make the language more expressive, there are sentences with the meaning 'probability is less than, less than or equal to, etc. q.'

For specifying a domain in SLAOP, the question of what world an agent is in does not arise. But due to partial observability, after the agent has executed a few actions, the agent will only have an (uncertain) *belief* about which world it is in, as opposed to (certain) *knowledge* of where it is. For an agent to reason with beliefs, the notion of an epistemic or belief state needs to be added to SLAOP. We would also like to add a notion of the expected value of a sequence of actions, and then be able to determine whether the expected value is less than, less than or equal to, etc. some given value. As in SLAP, the $\Box$ (necessity) operator in SLAOP may not be nested. All this is dealt with by SDL.

The complexity of the decision procedure has not been analysed. Our focus for SLAOP is mainly decidability. Evaluation of the systems of equations in the label assignment phase has the potential for being very expensive. These are linear systems of equations; one could thus investigate Linear Programming methods [Dantzig, 1963 & 1998, Murty, 1983, Gass, 2010] to optimize the evaluation of the systems.

We feel that presenting a decidability result for a new class of logics is not trivial. Even though the entailment problem in SLAOP—as presented in this chapter—may be intractable, it is important to have a decision procedure as a launchpad for tackling the computational complexity.

SLAOP is a decidable logic for specifying stochastic actions, stochastic observation, and rewards and costs of actions. The introduction of equality between actions and observations and the definition of finite quantification allows one to express some kinds of sentences more compactly.

A sound and complete procedure for deciding whether entailment queries hold was presented. One of the main contributions of the work is the formulation of the decision procedure with a tableau phase and a label assignment phase which appeals to solving systems of equations.

Due to the approach of using label assignments in the entailment decision procedure, sentences of the form $\neg\Box\Phi$ are part of the language of SLAOP; entailment queries involving this form of sentence can be dealt with. The language of SLAP does not include sentences of this form because its entailment decision procedure cannot deal with them.

Both approaches have their pros and cons, so we decided to document them both. Please refer to the discussion at the beginning of Section 6.2.

The logic cannot be used directly for reasoning about what is true after some sequence of actions and observations, and the utility of a sequence of actions cannot be determined. Moreover, we must add a notion of belief-state if we want a logic which simulates POMDPs. These issues will be tackled in the logic defined in the next chapter.

7. THE STOCHASTIC DECISION LOGIC

A paper about the work presented in this chapter has been accepted for presentation at the Seventh International Conference on Agents and Artificial Intelligence (ICAART), held in Lisbon, Portugal in January 2015.

In this chapter, we propose the *Stochastic Decision Logic* (SDL), combining the benefits of POMDP theory and logic for posing entailment queries about POMDP models. Traditionally, to make any deductions in POMDP theory, a POMDP model must be completely specified. *A major contribution of this work is that it allows the user to determine whether or not a set of sentences is entailed by an arbitrarily precise specification of a POMDP model.* By "arbitrarily precise specification" we mean that the transition function, the perception function, the reward function or the initial belief-state may not be completely defined by the logical specification provided. Another view is that the logic allows for the (precise) specification of and reasoning over *classes* of POMDP models.

Full-scale planning will not be considered here. However, as a preliminary step, projections concerning epistemic situations and expected rewards will be possible. That is, at this stage, the logic cannot be used to produce a reward-maximizing policy conditioned on observations. The logic can, however, track beliefs, given a sequence of executed actions or a sequence of actions being considered for execution. More precisely, with SDL, an agent can determine (i) the degree of belief in a propositional sentence after an arbitrary finite number of actions and observations and (ii) the utility of a finite sequence of actions after a number of actions and observations. We provide a procedure to determine whether some hypothesised situation follows from a knowledge base of the system and some beliefs corresponding to or possibly in conflict with the real system state.

In SDL, a particular kind of entailment is important and we shall provide an entailment checking procedure. Soundness, completeness and termination of the proposed entailment decision procedure is proved. Moreover, SDL is *decidable* with respect to entailment.

Most syntactic and semantic elements of SDL have their foundations in the Specification Logic of Actions with Probability (SLAP) and the Specification Logic of Actions and Observations with Probability (SLAOP). However, one cannot reason about degrees of belief or expected rewards (utility) in those logics. [1]

Here is a flavor of the language and use of SDL referring to the 'oil-drinking' scenario: The syntactic elements mentioned in the following examples are formally defined in Section 7.1.1. $\mathbf{B}\varphi \geq p$ is read 'The degree of belief in φ is greater than or equal to p' and $\mathbf{U}\Lambda > r$ is read 'The utility of performing Λ is greater than r'. Given a complete formalization $\mathcal{K}$ of the oil-drinking scenario, a robot may have the following queries:

[1] SDL is not, formally speaking, an extension of either of these.

- Is the degree of belief that I'll have the oil-can in my gripper greater than or equal to 0.9, after I attempt grabbing it twice in a row? That is, does

$$[\![\texttt{grab}, \texttt{obsNil}]\!]\,[\![\texttt{grab}, \texttt{obsNil}]\!]\mathbf{B}(\texttt{holding}) \geq 0.9$$

  follow from $\mathcal{K}$?

- After grabbing the can, then perceiving that it has medium weight, is the utility of drinking the contents of the oil-can, then placing it on the floor, more than 6 units? That is, does

$$[\![\texttt{grab}, \texttt{obsNil}]\!]\,[\![\texttt{weigh}, \texttt{obsMedium}]\!]\mathbf{U}[\![\texttt{drink}]\!]\,[\![\texttt{replace}]\!] > 6$$

  follow from $\mathcal{K}$?

In SDL, the specification of action and perception rules follow the same recipe as in SLAOP. Nevertheless, the specification framework will be presented in terms of SDL. What needs to be addressed in particular is how to specify initial belief-states. The following correspondence is noted. Let

$$\phi \rightarrow \Phi \in \mathcal{L}_{SLAOP}$$

be an effect, effect closure or inexecutability axiom, or a perception or utility rule. Then $\Box(\phi \rightarrow \Phi)$ in SLAOP corresponds to

$$\phi \Rightarrow \Phi$$

in SDL.

Section 7.1 defines SDL. Section 7.2 provides a decision procedure for determining entailment of sentences in SDL. In Section 7.3, we prove that the procedure is sound, complete and that it terminates, that is, we show that SDL is decidable with respect to entailment. Section 7.4 explicates the framework for specifying domains with SDL, including how to write information concerning belief-states and utilities of sequences of actions. Section 7.5 presents several examples of entailment queries, using the oil-drinking scenario.

7.1 Defining the Logic

First, the syntax of the logic is presented, then its semantics. The last subsection discusses the correspondence between POMDPs and SDL.

7.1.1 Syntax

The vocabulary of our language contains six sorts of objects of interest:

1. a finite set of *fluents* $\mathcal{F} = \{f_1, \ldots, f_n\}$,
2. a finite set of names of atomic *actions* $\mathcal{A} = \{\alpha_1, \ldots, \alpha_n\}$,
3. a countable set of *action variables* $V_{\mathcal{A}} = \{v^a_1, v^a_2, \ldots\}$,
4. a finite set of names of atomic *observations* $\Omega = \{\varsigma_1, \ldots, \varsigma_n\}$,
5. a countable set of *observation variables* $V_\Omega = \{v^o_1, v^o_2, \ldots\}$,

6. all *real numbers* $\mathbb{R}$.

We denote $\mathbb{R} \cap [0,1]$ as $[0,1]$ and we refer to elements of $\mathcal{A} \cup \Omega$ as *constants*. We work in a multi-modal setting, in which we have modal operators $[\alpha]$, one for each $\alpha \in \mathcal{A}$, and a *belief update* modal operator (or *update operator* for short) $[\![\alpha + \varsigma]\!]$, one for each pair in $\mathcal{A} \times \Omega$. Intuitively, $[\![\alpha+\varsigma]\!]\Psi$ means 'Ψ holds in the belief-state resulting from performing action α and then perceiving ς'. For instance, $[\![\alpha_1 + \varsigma_1]\!]\ [\![\alpha_2 + \varsigma_2]\!]$ expresses that the agent executes α_1 then perceives ς_1 then executes α_2 then perceives ς_2. $\mathbf{B}$ is a modal operator for belief and $\mathbf{U}$ is a modal operator for utility.

We first define a language $\mathcal{L}$, then a useful sublanguage $\mathcal{L}_{SDL} \subset \mathcal{L}$. The reason why we define $\mathcal{L}$ is because it is easier to define the truth condition for $\mathcal{L}$; the truth conditions for $\mathcal{L}_{SDL}$ then follow directly.

Definition 7.1.1: First the propositional fragment:

$$\varphi ::= f \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi,$$

where $f \in \mathcal{F}$. Then the fragment Φ used in formulae of the form $\varphi \Rightarrow \Phi$ (see the definition of Θ below). Let $\alpha \in (V_{\mathcal{A}} \cup \mathcal{A})$, $\varsigma \in (V_{\Omega} \cup \Omega)$, $p \in [0,1]$, $r \in \mathbb{R}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$.

$$\Phi ::= \varphi \mid \alpha = \alpha \mid \varsigma = \varsigma \mid \mathit{Reward}(r) \mid \mathit{Cost}(\alpha, r) \mid [\alpha]\varphi \bowtie p \mid (\alpha|\varsigma) \bowtie p \mid \neg\Phi \mid \Phi \wedge \Phi,$$

where φ is defined above. Let $v^a \in V_{\mathcal{A}}$ and $v^o \in V_{\Omega}$. The language of $\mathcal{L}$ is defined as Θ:

$$\begin{aligned}
&\Lambda ::= [\![\alpha]\!] \mid \Lambda[\![\alpha]\!] \\
&\Theta ::= \top \mid \alpha = \alpha \mid \varsigma = \varsigma \mid \mathit{Cont}(\alpha, \varsigma) \mid \mathbf{B}\varphi \ \bowtie\ p \mid \mathbf{U}\Lambda \ \bowtie\ r \mid \\
&\quad \varphi \Rightarrow \Phi \mid [\![\alpha + \varsigma]\!]\Theta \mid (\forall v^a)\Theta \mid (\forall v^o)\Theta \mid \neg\Theta \mid \Theta \wedge \Theta \mid \Theta \vee \Theta,
\end{aligned}$$

where φ and Φ are defined above.

The language of SDL, denoted $\mathcal{L}_{SDL}$, is the subset of formulae of $\mathcal{L}$ excluding formulae containing subformulae of the form $\neg(\varphi \Rightarrow \Phi)$.

The scope of quantifier $(\forall v')$ is determined in the same way as is done in first-order logic. A variable v appearing in a formula Θ is said to be *bound* by quantifier $(\forall v')$ if and only if v is the same variable as v' and is in the scope of $(\forall v')$. If a variable is not bound by any quantifier, it is *free*. In $\mathcal{L}$, variables are not allowed to be free; they are always bound. That ends the definition.

$[\alpha]\varphi \bowtie p$ is read 'The probability x of reaching a φ-world after executing α is such that $x \bowtie p$'. Whereas $[\alpha]$ is a modal operator, $(\varsigma|\alpha)$ is a predicate; $(\varsigma|\alpha) \bowtie p$ is read 'The probability x of perceiving ς, given α was performed is such that $x \bowtie p$'.

$\mathit{Cont}(\alpha, \varsigma)$ is read 'Consciousness continues after executing α and then perceiving ς'. $\mathbf{B}\varphi \ \bowtie\ p$ is read 'The degree of belief x in φ is such that $x \bowtie p$'. Performing $\Lambda = [\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_z]\!]$ means that α_1 is performed, then α_2 then ... then α_z. $\mathbf{U}\Lambda \ \bowtie\ r$ is thus read 'The utility x of performing Λ is such that $x \bowtie r$'. Evaluating some sentence Ψ after a sequence of z update operations, means

that Ψ will be evaluated after the agent's belief-state has been updated according to the sequence

$$\underbrace{[\![\alpha + \varsigma]\!] \cdots [\![\alpha' + \varsigma']\!]}_{z \text{ times}}$$

of actions and observations. $\varphi \Rightarrow \Phi$ is read 'It is a general law of the domain that Φ holds in all situations (worlds) which satisfy φ'.

Note that, for instance, $\neg(\varphi \Rightarrow \Phi) \wedge (\varphi' \Rightarrow \Phi') \wedge \neg Cont(\alpha, \varsigma) \notin \mathcal{L}_{SDL}$, but $(\varphi \Rightarrow \Phi) \wedge (\varphi' \Rightarrow \Phi') \wedge \neg Cont(\alpha, \varsigma) \in \mathcal{L}_{SDL}$. And, for instance, $\neg(\forall v')(\varphi \Rightarrow \Phi) \vee (\varphi' \Rightarrow \Phi') \vee \neg(\forall v'')\, Cont(\alpha, \varsigma) \notin \mathcal{L}_{SDL}$, but $(\forall v')(\varphi \Rightarrow \Phi) \vee (\varphi' \Rightarrow \Phi') \vee \neg(\forall v'')\, Cont(\alpha, \varsigma) \in \mathcal{L}_{SDL}$.

$\perp$ abbreviates $\neg\top$, $\theta \rightarrow \theta'$ abbreviates $\neg\theta \vee \theta'$ and $\leftrightarrow$ abbreviates $(\theta \rightarrow \theta') \wedge (\theta' \rightarrow \theta)$. In grammars φ and Φ, $\theta \vee \theta'$ abbreviates $\neg(\neg\theta \wedge \neg\theta')$, but in grammar Θ, $\vee$ is defined directly. $\rightarrow$ and $\leftrightarrow$ have the weakest bindings, with $\Rightarrow$ just stronger; and $\neg$ the strongest. Parentheses enforce or clarify the scope of operators conventionally.

$c = c'$ is an equality literal, $Reward(r)$ is a reward literal, $Cost(\alpha, r)$ is a cost literal, $[\alpha]\varphi \bowtie p$ is a dynamic literal, $(\varsigma|\alpha) \bowtie p$ is a perception literal, and $\varphi \Rightarrow \Phi$ is a law literal. $Cont(\alpha, \varsigma)$ is an executability literal, $\mathbf{B}\varphi \bowtie p$ is a belief literal and $\mathbf{U}\Lambda \bowtie r$ is a utility literal. The negation of all these literals are also literals with the associated names.

Note that formulae with nested modal operators of the form $\mathbf{BB}\varphi$, $\mathbf{BBB}\varphi$, et cetera, $\mathbf{UU}\Lambda$, $\mathbf{UUU}\Lambda$, et cetera, $[\alpha][\alpha]\varphi$ and $[\alpha][\alpha][\alpha]\varphi$ et cetera are not in $\mathcal{L}_{SDL}$. However, formulae of the form $[\![\alpha + \varsigma]\!]([\![\alpha' + \varsigma']\!]\Psi' \wedge [\![\alpha'' + \varsigma'']\!]\Psi'')$, $[\![\alpha + \varsigma]\!](\Psi \wedge [\![\alpha' + \varsigma']\!][\![\alpha'' + \varsigma'']\!]\Psi')$ for instance, are in $\mathcal{L}_{SDL}$.

7.1.2 Semantics

Let $w : \mathcal{F} \mapsto \{0, 1\}$ be a total function that assigns a truth value to each fluent. We call w a *world*. Let C be the set of $2^{|\mathcal{F}|}$ *conceivable worlds*, that is, all possible functions w.

Definition 7.1.2: An SDL structure is a tuple $\mathcal{D} = \langle R, Q, U \rangle$ such that[2]

- $R : \mathcal{A} \mapsto R_\alpha$, where $R_\alpha : (C \times C) \mapsto [0, 1]$ is a total function from pairs of worlds into the reals; That is, R is a mapping that provides an accessibility relation R_α for each action $\alpha \in \mathcal{A}$; For every $w^- \in C$, it is required that either $\sum_{w^+ \in C} R_\alpha(w^-, w^+) = 1$ or $\sum_{w^+ \in C} R_\alpha(w^-, w^+) = 0$.
- $Q : \mathcal{A} \mapsto Q_\alpha$, where $Q_\alpha : (C \times \Omega) \mapsto [0, 1]$ is a total function from pairs in $C \times \Omega$ into the reals; That is, Q is a mapping that provides a perceivability relation Q_α for each action $\alpha \in \mathcal{A}$; For all $w^+ \in C$, if there exists a $w^- \in C$ such that $R_\alpha(w^-, w^+) > 0$, then $\sum_{\varsigma \in \Omega} Q_\alpha(w^+, \varsigma) = 1$, else $\sum_{\varsigma \in \Omega} Q_\alpha(w^+, \varsigma) = 0$.
- U is a pair $\langle Re, Co \rangle$, where $Re : C \mapsto \mathbb{R}$ is a reward function and Co is a mapping that provides a cost function $Co_\alpha : C \mapsto \mathbb{R}$ for each $\alpha \in \mathcal{A}$.

[2] The set of possible worlds W is not part of SDL structures because the whole set of conceivable worlds will always be referenced. And the set O and associated naming function N found in LAO and SLAOP structures are left out, because N is a bijection, meaning that O actually adds nothing. In retrospect, LAO and SLAOP structures could also work without O and N.

As in partially observable Markov decision processes (POMDPs), in SDL, an agent typically does not know in which world $w \in C$ it actually is, but for each w it has a degree of belief that it is in that world. Let $b : C \mapsto [0, 1]$ be a probability distribution over C, referred to as a *belief-state*. The degree of belief in w is denoted by the probability measure $b(w)$. We refer to all probability mass functions b over C as the set P.

As an agent acts and evolves, what it believes changes, that is, b changes for each agent activity. We assume the agent remains in one physical environment for its lifetime, and we assume that its sets $\mathcal{A}$ and Ω remain the same. The fact that the agent does not change environments has the consequence that the structure modeling the agent and its environment also remain static.

Definition 7.1.3: The probability of reaching the next belief-state after executing α and perceiving ς is

$$P_{NB}(\alpha, \varsigma, b) = \sum_{w' \in C} Q_\alpha(\varsigma, w') \sum_{w \in C} R_\alpha(w, w') b(w).$$

$P_{NB}(\cdot)$ has the same intuitive meaning as $Pr(z \mid a, b)$ (Eq. 3.2).

Definition 7.1.4: We define a *belief update* function $BU(\alpha, \varsigma, b) = b'$:

$$b'(w') = \frac{Q_\alpha(w', \varsigma) \sum_{w \in C} R_\alpha(w, w') b(w)}{P_{NB}(\alpha, \varsigma, b)},$$

for $P_{NB}(\alpha, \varsigma, b) \neq 0$.

b' is the belief-state attained by executing α in b and perceiving ς in the resulting state. Recall that $b'(w')$ is the probability of being in world w' while in belief-state b'. $BU(\cdot)$ has the same intuitive meaning as the state estimation function (Eq. 3.1).

Let a be a POMDP action and s a POMDP state. Given the opportunity to be slightly more clear about the specification of rewards in SDL, we interpret $R(a, s)$ from the chapter reviewing POMDPs as $R(s) - C(a, s)$, where $R(s)$ provides the positive reward portion of $R(a, s)$ and $C(a, s)$ provides the punishment or cost portion. By this interpretation, we assume that simply being in a state has an intrinsic reward (independent of an action), however, that punishment is conditional on actions *and* the states in which they are executed. There are many other ways to interpret $R(a, s)$, and $R(a, s)$ is not even the most general reward function possible; a more general function is $R(s, a, s')$ meaning that rewards depend on a state s, an action executed in s and a state s' reached due to performing a in s. SDL adopts one of several reasonable approaches. In the semantics of SDL, we equate state s with world $w \in C$ and action a with $\alpha \in \mathcal{A}$, and interpret $R(a, s)$ as $Re(w) - Co_\alpha(w)$.

We derive a reward function over belief-states for SDL in a similar fashion as we did with Equation 3.3, however, including the notion of cost.

$$RC(\alpha, b) = \sum_{w \in C} (Re(w) - Co_\alpha(w)) b(w).$$

Definition 7.1.5: Let $\alpha, \alpha' \in \mathcal{A}$, $\varsigma, \varsigma' \in \Omega$, $p \in [0, 1]$ and $r \in \mathbb{R}$. Let $f \in \mathcal{F}$ and let Θ be any sentence in $\mathcal{L}$. Let $\bowtie \in \{<, \leq, =, \geq, >\}$. We say $\Theta \in \mathcal{L}$ is *satisfied* at world w and belief-state b

in SDL structure $\mathcal{D}$ (written $\mathcal{D}bw \models \Theta$) if and only if the following holds:[3]

$\mathcal{D}bw \models \top$ for all $w \in C$;

$\mathcal{D}bw \models f \iff w(f) = 1$;

$\mathcal{D}bw \models \neg\varphi \iff \mathcal{D}bw \not\models \varphi$;

$\mathcal{D}bw \models \varphi \wedge \varphi' \iff \mathcal{D}bw \models \varphi$ and $\mathcal{D}bw \models \varphi'$;

$\mathcal{D}bw \models \alpha = \alpha' \iff \alpha$ and α' are the same element;

$\mathcal{D}bw \models \varsigma = \varsigma' \iff \varsigma$ and ς' are the same element;

$\mathcal{D}bw \models Reward(r) \iff Re(w) = r$;

$\mathcal{D}bw \models Cost(\alpha, c) \iff Co_\alpha(w) = c$;

$\mathcal{D}bw \models [\alpha]\varphi \bowtie p \iff \sum_{\substack{w' \in C \\ \mathcal{D}bw' \models \varphi}} R_\alpha(w, w') \bowtie p$;

$\mathcal{D}bw \models (\varsigma|\alpha) \bowtie p \iff Q_\alpha(w, \varsigma) \bowtie p$;

$\mathcal{D}bw \models \neg\Phi \iff \mathcal{D}bw \not\models \Phi$;

$\mathcal{D}bw \models \Phi \wedge \Phi' \iff \mathcal{D}bw \models \Phi$ and $\mathcal{D}bw \models \Phi'$;

$\mathcal{D}bw \models Cont(\alpha, \varsigma) \iff P_{NB}(\alpha, \varsigma, b) \neq 0$;

$\mathcal{D}bw \models \mathbf{B}\varphi \bowtie p \iff \sum_{\substack{w' \in C \\ \mathcal{D}bw' \models \varphi}} b(w') \bowtie p$;

$\mathcal{D}bw \models \mathbf{U}[\![\alpha]\!] \bowtie r \iff RC(\alpha, b) \bowtie r$;

$\mathcal{D}bw \models \mathbf{U}[\![\alpha]\!]\Lambda \bowtie r \iff \left(RC(\alpha, b) + \sum_{\substack{\varsigma \in \Omega \\ b' = BU(\alpha,\varsigma,b) \\ \mathcal{D}b'w \models \mathbf{U}\Lambda = r'}} P_{NB}(\alpha, \varsigma, b) \cdot r'\right) \bowtie r$;

$\mathcal{D}bw \models \varphi \Rightarrow \Theta \iff$ for all $w' \in C$, $\mathcal{D}bw' \not\models \varphi$ or $\mathcal{D}bw' \models \Theta$;

$\mathcal{D}bw \models [\![\alpha + \varsigma]\!]\Theta \iff P_{NB}(\alpha, \varsigma, b) \neq 0$ and $\mathcal{D}b'w \models \Theta$, where $b' = BU(\alpha, \varsigma, b)$;

$\mathcal{D}bw \models \neg\Theta \iff \mathcal{D}bw \not\models \Theta$;

$\mathcal{D}bw \models \Theta \wedge \Theta' \iff \mathcal{D}bw \models \Theta$ and $\mathcal{D}bw \models \Theta'$;

$\mathcal{D}bw \models \Theta \vee \Theta' \iff \mathcal{D}bw \models \Theta$ or $\mathcal{D}bw \models \Theta'$;

$\mathcal{D}bw \models (\forall v^a)\Theta \iff \mathcal{D}bw \models \Theta|^{v^a}_{\alpha_1} \wedge \ldots \wedge \Theta|^{v^a}_{\alpha_n}$;

$\mathcal{D}bw \models (\forall v^o)\Theta \iff \mathcal{D}bw \models \Theta|^{v^o}_{\varsigma_1} \wedge \ldots \wedge \Theta|^{v^o}_{\varsigma_n}$,

where we write $\Theta|^v_c$ to mean the formula Θ with all variables $v \in (V_\mathcal{A} \cup V_\Omega)$ appearing in it replaced by constant $c \in \mathcal{A} \cup \Omega$ of the right sort.

A sentence $\Psi \in \mathcal{L}_{SDL}$ is *satisfiable* if there exists a structure $\mathcal{D}$, a belief-state b and a world w such that $\mathcal{D}bw \models \Psi$, else Ψ is *unsatisfiable*.

[3] As is convention in this thesis, $\iff$ means 'if and only if'. Where $\Rightarrow$ appears in the definition of $\varphi \Rightarrow \Theta$ below, it is clear that the symbol is used at the object level.

Let $\mathcal{K} \subseteq \mathcal{L}_{SDL}$ and $\Psi \in \mathcal{L}_{SDL}$. We say that Ψ is a *local semantic consequence* of $\mathcal{K}$ (or $\mathcal{K}$ *entails* Ψ; denoted $\mathcal{K} \models \Psi$) if for all structures $\mathcal{D}$, all belief-states b, all $w \in C$: if $\mathcal{D}bw \models \kappa$ for every $\kappa \in \mathcal{K}$, then $\mathcal{D}bw \models \Psi$. When $\mathcal{K}$ is a finite subset of $\mathcal{L}_{SDL}$, it is easy to show that $\mathcal{K} \models \Psi \iff \bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \neg\Psi$ is unsatisfiable. SDL decision procedure for entailment is based on this latter correspondence.

7.1.3 The Correspondence Between SDL and POMDPs

In this section we investigate the theoretical correspondence between SDL and POMDPs. We start with some preliminary remarks to align SDL structures and POMDP models. Theorem 7.1.1 is the main result.

In much literature on POMDPs, a model or tuple of components is not defined; the components of a POMDP are simply presented outside of any particular structure (e.g., [Geffner and Bonet, 1998, Smith and Simmons, 2005]). Kaelbling et al. [1998] define a POMDP model as $\langle S, A, T, R, \Omega, O\rangle$ without the initial belief-state, and Virin et al. [2007] define a POMDP model as $\langle S, A, tr, R, \Omega, O, b^0\rangle$, where tr is used instead of T. Pineau et al. [2003] define a POMDP model as $\langle S, A, O, b^0, T, \Omega, R, \gamma\rangle$, even including the discount factor, but switching the use of O and Ω so that for them, O is the set of observations and Ω is the observation function. We use Z for the name of the observation function, as in the book Probabilistic Robotics [Thrun et al., 2005]. Clearly, the convention for describing a POMDP has not yet been settled.

We have chosen to ignore the discount factor γ in SDL; equivalently, we consider only POMDP models with $\gamma = 1$. This is done for two reasons, (i) we wish to introduce our logic as simply as possible (without being trivial) and (ii) finite horizon problems do not require a discount factor for convergence when seeking a solution policy.

Let $\mathcal{L}^{\not\Rightarrow}_{SDL}$ be the subset of formulae of $\mathcal{L}_{SDL}$ excluding formulae containing subformulae of the form $\varphi \Rightarrow \Phi$ (i.e., excluding law literals). Fix a set of fluents $\mathcal{F}$ and let C be the conceivable worlds induced from $\mathcal{F}$. Let δ^w be a complete propositional theory such that $w \models \delta^w$ and for all other $w' \in C$, $w' \not\models \delta^w$. Let $M = \langle S, A, T, R, Z, O, b^0\rangle$ be any POMDP model such that $S \subseteq C$, $S \neq \emptyset$, $\gamma = 1$ in the POMDP value function, $\mathcal{A} = A$ and $\Omega = Z$. Let $b^0 = \{(w_1, p_1), (w_2, p_2), \ldots, (w_n, p_n)\}$ and $S = \{w_1, w_2, \ldots, w_n\}$.

Definition 7.1.6: Let $BK \subset \mathcal{L}_{SDL}$ and $IB \in \mathcal{L}_{SDL}$. $\langle BK, IB\rangle$ is a *specification* of POMDP M if and only if

- For every $w^\times \in C$, if $w^\times \notin S$, then $BK \models \delta^{w^\times} \Rightarrow \bot$.
- For all $w, w' \in S$ and all $\alpha \in \mathcal{A}$, $BK \models \delta^w \Rightarrow ([\alpha]\delta^{w'} = p)$, such that $p = T(w, \alpha, w')$.
- For all $w \in S$, all $\varsigma \in \Omega$ and all $\alpha \in \mathcal{A}$, $BK \models \delta^w \Rightarrow ((\varsigma \mid \alpha) = p)$, such that $p = O(w, \alpha, \varsigma)$.
- For all $w \in S$ and all $\alpha \in \mathcal{A}$, $BK \models \delta^w \Rightarrow (Reward(r) \wedge Cost(\alpha, 0))$, such that $r = R(\alpha, w)$.
- $IB \models \mathbf{B}\delta^{w_1} = p_1 \wedge \mathbf{B}\delta^{w_2} = p_2 \wedge \cdots \wedge \mathbf{B}\delta^{w_n} = p_n$.

What the following lemma essentially says is that Θ^- is satisfied at $\mathcal{D}$, b^0 and w if and only if $IB \rightarrow \Theta^-$ is satisfied at $\mathcal{D}$, w and any arbitrary belief-state, where IB specifies b^0.

Lemma 7.1.1: Let $\langle BK, IB\rangle$ be a *specification* of POMDP $M = \langle S, A, T, R, Z, O, b^0\rangle$. Let Θ^- be an element of $\mathcal{L}^{\not\Rightarrow}_{SDL}$, $\mathcal{D}$ an arbitrary SDL structure and w an arbitrary element of C. Then $\mathcal{D}b^0w \models \Theta^-$ if and only if for all belief-states $b' \in P$, $\mathcal{D}b'w \models IB \rightarrow \Theta^-$.

Proof:
Clearly, if $\mathcal{D}b^0w \models IB$ then for all $b' \in P$, if $b' \neq b^0$, then $\mathcal{D}b'w \not\models IB$. Therefore, $\mathcal{D}b^0w \models \Theta^-$
$\iff$ for all $b' \in P$, if $b' \neq b^0$, then $\mathcal{D}b'w \not\models IB$, else $\mathcal{D}b'w \models \Theta^-$
$\iff$ for all $b' \in P$, $\mathcal{D}b'w \not\models IB$ or $\mathcal{D}b'w \models \Theta^-$
$\iff$ for all $b' \in P$, $\mathcal{D}b'w \models IB \rightarrow \Theta^-$. ■

Informally, the following lemma sets up correspondences between three forms of SDL formulae and the three corresponding notions defined in the chapter on POMDPs (Chap. 3).

Lemma 7.1.2: Let $S^\varphi = \{w \in C \mid w \models \varphi\}$ such that $S^\varphi \subseteq S$, where $\varphi \in \mathcal{L}_{SDL}$ is a propositional sentence.

For all $b \in P$ and all $w \in C$:
(i) $\mathcal{D}bw \models \mathbf{U}\Lambda = r$ iff $U(\Lambda, b) = r$,
(ii) $\mathcal{D}bw \models \mathbf{B}\varphi = p$ iff $B(S^\varphi, b) = p$ and
(iii) given sequence $[\![\alpha^1 + \varsigma^1]\!] \cdots [\![\alpha^y + \varsigma^y]\!]$, $SE(\alpha^y, \varsigma^y, \cdots\ SE(\alpha^1, \varsigma^1, b^0) \cdots) = BU(\alpha^y, \varsigma^y, \cdots\ BU(\alpha^1, \varsigma^1, b^0) \cdots)$.

Proof:
Let $\mathcal{D} = \langle R, Q, U\rangle$ be an SDL structure such that $\mathcal{D}bw \models \bigwedge_{\beta \in BK} \beta$ for all $b \in P$ and all $w \in C$. Due to Definition 7.1.6, it must be that for all $w, w' \in S$, all $\varsigma \in \Omega$ and all $\alpha \in \mathcal{A}$, $R_\alpha(w, w') = T(w, \alpha, w')$, $Q_\alpha(w, \varsigma) = O(w, \alpha, \varsigma)$ and $Re(w) - Co_\alpha(w) = R(\alpha, w)$, where $U = \langle Re, Co\rangle$.

Then, for all $\alpha \in \mathcal{A}$, $\varsigma \in \Omega$ and $b \in P$, by Definition 7.1.3, $P_{NB}(\alpha, \varsigma, b) = Pr(\varsigma \mid \alpha, b)$, by Definition 7.1.4, $BU(\alpha, \varsigma, b) = SE(\alpha, \varsigma, b)$ and by definitions of $RC(\cdot)$ and $\rho(\cdot)$, $RC(\alpha, b) = \rho(\alpha, b)$.

Now, given the equations above, the symmetries in the following definitions should be obvious: (i) $\mathcal{D}bw \models \mathbf{U}\Lambda = r$ and $U(\Lambda, b) = r$, (ii) $\mathcal{D}bw \models \mathbf{B}\varphi = p$ and $B(S^\varphi, b) = p$ and (iii) $SE(\cdot)$ and $BU(\cdot)$.

The lemma follows. ■

The theorem below relates POMDP models with SDL structures by relating functions $U(\cdot)$ and $B(\cdot)$ defined in Chapter 3 to SDL modal operators $\mathbf{U}$ and $\mathbf{B}$.

Theorem 7.1.1: Let $\langle BK, IB\rangle$ be a *specification* of POMDP $M = \langle S, A, T, R, Z, O, b^0\rangle$. Let $S^\varphi = \{w \in C \mid w \models \varphi\}$ such that $S^\varphi \subseteq S$, where $\varphi \in \mathcal{L}_{SDL}$ is a propositional sentence.

Given any sequence of actions and observations $[\![\alpha^1 + \varsigma^1]\!] \cdots [\![\alpha^y + \varsigma^y]\!]$, $U(\Lambda, b^y) \bowtie r$ and $B(S^\varphi, b^y) \bowtie' p$, where $b^y = SE(\alpha^y, \varsigma^y, \cdots SE(\alpha^1, \varsigma^1, b^0) \cdots)$ if and only if $BK \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!] \cdots [\![\alpha^y + \varsigma^y]\!](\mathbf{U}\Lambda \bowtie r \wedge \mathbf{B}\varphi \bowtie' p)$.

Proof:
Let $\mathcal{D} = \langle R, Q, U\rangle$ be an SDL structure such that $\mathcal{D}bw \models \bigwedge_{\beta\in BK}\beta$ for all $b \in P$ and all $w \in C$.

Let $b^y = SE(\alpha^y, \varsigma^y, \cdots SE(\alpha^1, \varsigma^1, b^0)\cdots)$.

($\Rightarrow$) Suppose $U(\Lambda, b^y) \bowtie r$ and $B(S^\varphi, b^y) \bowtie' p$. Then by items (i) and (ii) of Lemma 7.1.2, for all $w \in C$: $\mathcal{D}b^y w \models \mathbf{U}\Lambda \bowtie r$ and $\mathcal{D}b^y w \models \mathbf{B}\varphi \bowtie' p$. Thus, by item (iii) of Lemma 7.1.2 and the definition of $[\![\varsigma + \alpha]\!]\Phi$, for all $w \in C$: $\mathcal{D}b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{U}\Lambda = r$ and $\mathcal{D}b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{B}\varphi = p$. Hence, by construction of $\mathcal{D}$ for all structures $\mathcal{D}'$ and all $w \in C$: if $\mathcal{D}'b^0 w \models \beta$ for every $\beta \in BK$, then $\mathcal{D}'b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{U}\Lambda \bowtie r$ and $\mathcal{D}'b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{B}\varphi \bowtie' p$. By Lemma 7.1.1, this implies that for all structures $\mathcal{D}'$, all belief-states b and all $w \in C$: if $\mathcal{D}'bw \models \beta$ for every $\beta \in BK$, then $\mathcal{D}'bw \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{U}\Lambda \bowtie r$ and $\mathcal{D}'bw \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{B}\varphi \bowtie' p$. Referring to item (iii) of Lemma 7.1.2 and using the definition of $\wedge$, this implies that for all structures $\mathcal{D}'$, all belief-states b, all $w \in C$: if $\mathcal{D}'bw \models \beta$ for every $\beta \in BK$, then $\mathcal{D}'bw \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!](\mathbf{U}\Lambda \bowtie r \wedge \mathbf{B}\varphi \bowtie' p)$. Therefore, due to the definition of entailment, $BK \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!](\mathbf{U}\Lambda \bowtie r \wedge \mathbf{B}\varphi \bowtie' p)$.

($\Leftarrow$) On the other hand, suppose $BK \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!](\mathbf{U}\Lambda \bowtie r \wedge \mathbf{B}\varphi \bowtie' p)$. Then, due to the definition of entailment, for all structures $\mathcal{D}'$, all belief-states b and all $w \in C$: if $\mathcal{D}'bw \models \beta$ for every $\beta \in BK$, then $\mathcal{D}'bw \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!](\mathbf{U}\Lambda \bowtie r \wedge \mathbf{B}\varphi \bowtie' p)$. Thus, by item (iii) of Lemma 7.1.2 and the definition of $\wedge$, for all structures $\mathcal{D}'$, all belief-states b, all $w \in C$: if $\mathcal{D}'bw \models \beta$ for every $\beta \in BK$, then $\mathcal{D}'bw \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{U}\Lambda \bowtie r$ and $\mathcal{D}'bw \models IB \rightarrow [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{B}\varphi \bowtie' p$. Hence, by Lemma 7.1.1, for all structures $\mathcal{D}'$ and all $w \in C$: if $\mathcal{D}'b^0 w \models \beta$ for every $\beta \in BK$, then $\mathcal{D}'b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{U}\Lambda \bowtie r$ and $\mathcal{D}'b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{B}\varphi \bowtie' p$. Due to the way in which $\mathcal{D}$ is constructed, it follows that for all $w \in C$: $\mathcal{D}b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{U}\Lambda = r$ and $\mathcal{D}b^0 w \models [\![\alpha^1 + \varsigma^1]\!]\cdots[\![\alpha^y + \varsigma^y]\!]\mathbf{B}\varphi = p$. Using item (iii) of Lemma 7.1.2 and the definition of $[\![\varsigma + \alpha]\!]\Phi$, the former statement implies that for all $w \in C$: $\mathcal{D}b^y w \models \mathbf{U}\Lambda \bowtie r$ and $\mathcal{D}b^y w \models \mathbf{B}\varphi \bowtie' p$. Therefore, by items (i) and (ii) of Lemma 7.1.2, it must be the case that $U(\Lambda, b^y) \bowtie r$ and $B(S^\varphi, b^y) \bowtie' p$. ■

7.2 Decision Procedure for Semantic Consequence

We provide a decision procedure for checking the validity of sentences in $\mathcal{L}_{SDL}$. There are two phases in the decision procedure. First comes the *tableau* phase, which essentially eliminates propositional connectives, and processes sentences in preparation for the second phase. Second, the *systems of inequalities* (SI) phase creates systems of inequalities from literals, checking the systems' feasibility.

The labels of the labeled formulae defined for the decision procedure of SDL are not integers, but so-called *activity sequences*. By labeling formulae with activity sequences, one is able to track the 'movement' between sets of possible worlds. Certain machinery is required to process activity sequences. The formalities follow.

7.2.1 The Tableau Phase

The necessary definitions and terminology are given next.

Definition 7.2.1: A *labeled formula* is a pair (Σ, Ψ), where $\Psi \in \mathcal{L}_{SDL}$ is any formula and Σ is either 0 or a sequence of the form $0 \xrightarrow{\alpha_1,\varsigma_1} e_1 \xrightarrow{\alpha_2,\varsigma_2} e_2 \cdots \xrightarrow{\alpha_z,\varsigma_z} e_z$ called an *activity sequence*. The e_i represent belief-states. If Σ is $0 \xrightarrow{\alpha_1,\varsigma_1} e_1 \cdots \xrightarrow{\alpha_z,\varsigma_z} e_z$, then the concatenation of Σ and $\xrightarrow{\alpha',\varsigma'} e'$, denoted as $\Sigma \xrightarrow{\alpha',\varsigma'} e'$ is the sequence $0 \xrightarrow{\alpha_1,\varsigma_1} e_1 \cdots \xrightarrow{\alpha_z,\varsigma_z} e_z \xrightarrow{\alpha',\varsigma'} e'$.

Definition 7.2.2: If one has a tree with trunk $\Gamma^0_0 = \{(0, \Psi)\}$, we shall say one has *a tree for* Ψ.

A node Γ is *closed* if $(\Sigma, \bot) \in \Gamma$ for any Σ. It is *open* if it is not closed. A tree is closed if all of its leaf nodes are closed, else it is open.

Recall that δ^w is a complete propositional theory such that $w \models \delta^w$ and for all other $w' \in C$, $w' \not\models \delta^w$. We denote the smallest set of complete propositional theories entailing some propositional formula φ as $cpt(\varphi)$. We call elements of $cpt(\varphi)$ *definitive*.

A preprocessing step occurs, where all (sub)formulae of the form $(\forall v^a)\Psi$ and $(\forall v^o)\Psi$ are replaced by, respectively, $(\Psi|^{v^a}_{\alpha_1} \wedge \ldots \wedge \Psi|^{v^a}_{\alpha_n})$ and $(\Psi|^{v^o}_{\varsigma_1} \wedge \ldots \wedge \Psi|^{v^o}_{\varsigma_n})$.

Another preprocessing step occurs, where all (sub)formulae of the form $\varphi \Rightarrow \Phi$ are replaced by $\varphi \Rightarrow \Phi'$, where Φ' is the CNF of Φ.

The tableau rules for SDL follow. Let Γ^j_k be a leaf node.

- rule $\neg$: If Γ^j_k contains a formula (Σ, Ψ) with a double negation somewhere in it, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \Psi')\}$, where Ψ' is Ψ with the double negation removed.
- rule $\wedge$: If Γ^j_k contains $(\Sigma, \Psi \wedge \Psi')$ or $(\Sigma, \neg(\Psi \vee \Psi'))$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \Psi), (\Sigma, \Psi')\}$, respectively, $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \neg\Psi), (\Sigma, \neg\Psi')\}$.
- rule $\vee$: If Γ^j_k contains $(\Sigma, \Psi \vee \Psi')$ or $(\Sigma, \neg(\Psi \wedge \Psi'))$, then create nodes $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \Psi)\}$ and $\Gamma^{j'}_0 = \Gamma^j_k \cup \{(\Sigma, \Psi')\}$, or respectively, nodes $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \neg\Psi)\}$ and $\Gamma^{j'}_0 = \Gamma^j_k \cup \{(\Sigma, \neg\Psi')\}$, where j' is a fresh integer.
- rule $=$: If Γ^j_k contains $(\Sigma, c = c')$ or $(\Sigma, \varphi \Rightarrow c = c')$ where $\varphi \not\equiv \bot$ and c and c' are distinct constants, or if Γ^j_k contains $(\Sigma, \neg(c = c'))$ or $(\Sigma, \varphi \Rightarrow \neg(c = c'))$ where $\varphi \not\equiv \bot$ and c and c' are identical constants, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \bot)\}$.
- rule $\Rightarrow \wedge$: If Γ^j_k contains $(\Sigma, \varphi \Rightarrow \Phi \wedge \Phi')$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \varphi \Rightarrow \Phi), (\Sigma, \varphi \Rightarrow \Phi'))\}$.
- rule $\delta \Rightarrow$: If Γ^j_k contains $(\Sigma, \varphi \Rightarrow \Phi)$ where Φ is a disjunction of literals, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \delta_1 \Rightarrow \Phi), (\Sigma, \delta_2 \Rightarrow \Phi), \ldots, (\Sigma, \delta_n \Rightarrow \Phi)\}$, where $cpt(\varphi) = \{\delta_1, \delta_2, \ldots, \delta_n\}$.
- rule $\Rightarrow \vee$: If Γ^j_k contains $(\Sigma, \varphi \Rightarrow \Phi \vee \Phi')$ where φ is definitive, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, (\varphi \Rightarrow \Phi) \vee (\varphi \Rightarrow \Phi'))\}$.
- rule Ξ: If Γ^j_k contains $(\Sigma, [\![\alpha + \varsigma]\!]\Psi)$ then: if Γ^j_k contains (Σ', Ψ') such that $\Sigma' = \Sigma \xrightarrow{\alpha,\varsigma} e$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma', \Psi)\}$, else create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma \xrightarrow{\alpha,\varsigma} e', \Psi)\}$,

where e' is a fresh integer.

- rule $\neg\Xi$: If Γ^j_k contains $(\Sigma, \neg[\![\alpha+\varsigma]\!]\Psi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \neg Cont(\alpha,\varsigma) \vee [\![\alpha+\varsigma]\!]\neg\Psi)\}$.
- rule $\neg\mathbf{B}$: If Γ^j_k contains $(\Sigma, \neg\mathbf{B}\varphi \bowtie q)$, then
  - if $\bowtie$ is $<$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{B}\varphi \geq q)\}$.
  - if $\bowtie$ is $\leq$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{B}\varphi > q)\}$.
  - if $\bowtie$ is $=$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{B}\varphi < q \vee \mathbf{B}\varphi > q)\}$.
  - if $\bowtie$ is $\geq$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{B}\varphi < q)\}$.
  - if $\bowtie$ is $>$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{B}\varphi \leq q)\}$.
- rule $\neg\mathbf{U}$: If Γ^j_k contains $(\Sigma, \neg\mathbf{U}\Lambda \bowtie q)$, then
  - if $\bowtie$ is $<$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{U}\Lambda \geq q)\}$.
  - if $\bowtie$ is $\leq$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{U}\Lambda > q)\}$.
  - if $\bowtie$ is $=$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{U}\Lambda < q \vee \mathbf{U}\Lambda > q)\}$.
  - if $\bowtie$ is $\geq$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{U}\Lambda < q)\}$.
  - if $\bowtie$ is $>$, create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(\Sigma, \mathbf{U}\Lambda \leq q)\}$.

7.2.2 The SI Phase

Let $SI(\Gamma)$ be the system of inequalities generated from the formulae in Γ (explained later).

After the tableau phase is completed, the SI phase begins. Let T be a saturated tree.

> For each open leaf node Γ^j_k of T, do the following. If $SI(\Gamma^j_k)$ is infeasible, then create new leaf node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(0, \bot)\}$.

Definition 7.2.3: A tree is called *finished* after the SI phase is completed.

Definition 7.2.4: If a tree for $\neg\Psi$ is closed, we write $\vdash \Psi$. If there is a finished tree for $\neg\Psi$ with an open leaf node, we write $\nvdash \Psi$.

The generation of $SI(\Gamma)$ from the formulae in Γ is explained in the rest of this section. All variables are assumed implicitly non-negative. Let $C^{\#} = \{w_1, w_2, \ldots, w_n\}$ be an ordering of the worlds in C. Let ω^e_k be a variable representing the probability of being in world w_k at activity-point e (after a number of activity updates). We may denote an activity sequence as $\Sigma \xrightarrow{\alpha,\varsigma} e$ to refer to the last action α, observation ς and activity-point e in the sequence, where Σ may be the empty sequence. We may also denote an activity sequence as Σe to refer only to the last activity-point in the sequence; if Σ is the empty sequence, then e is the initial activity-point 0.

In the next four subsections, we deal with (i) law literals involving dynamic and perception literals, (ii) activity sequences, (iii) belief literals and (iv) laws involving reward and cost literals, and utility literals.

Action and Perception Laws

For every formulae of the form $(\Sigma, \phi \Rightarrow [\alpha]\varphi \bowtie q) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg[\alpha]\varphi \bowtie q) \in \Gamma$, for every j such that $w_j \models \phi$ (where j represents the world in which α is executed),

$$c_1 pr^{\alpha}_{j,1} + c_2 pr^{\alpha}_{j,2} + \cdots + c_n pr^{\alpha}_{j,n} \bowtie q, \text{ respectively, } c_1 pr^{\alpha}_{j,1} + c_2 pr^{\alpha}_{j,2} + \cdots + c_n pr^{\alpha}_{j,n} \not\bowtie q$$

is in $SI(\Gamma)$, such that $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$, and the $pr^{\alpha}_{j,k}$ are variables. Adding an equation

$$pr^{\alpha}_{j,1} + pr^{\alpha}_{j,2} + \cdots + pr^{\alpha}_{j,n} = \lceil pr^{\alpha}_{j,1} + pr^{\alpha}_{j,2} + \cdots + pr^{\alpha}_{j,n} \rceil$$

for every j such that $w_j \models \phi$, will ensure that either $\sum_{w' \in W} R_\alpha(w_j, w') = 1$ or $\sum_{w' \in W} R_\alpha(w_j, w') = 0$, for every $w_j \in C$, as stated in Definition 7.1.2.

Let $m = |\Omega|$. Let $\Omega^{\#} = (\varsigma_1, \varsigma_2, \ldots, \varsigma_m)$ be an ordering of the observations in Ω. With each observation in $\varsigma \in \Omega^{\#}$, we associate a variable pr^{ς}_j, where j represents the world in which ς is perceived. For every formulae of the form $(\Sigma, \phi \Rightarrow (\varsigma|\alpha) \bowtie q) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q) \in \Gamma$, for every j such that $w_j \models \phi$,

$$pr^{\varsigma|\alpha}_j \bowtie q, \text{ respectively, } pr^{\varsigma|\alpha}_j \not\bowtie q$$

is in $SI(\Gamma)$. Adding an equation

$$pr^{\varsigma_1|\alpha}_j + pr^{\varsigma_2|\alpha}_j + \cdots + pr^{\varsigma_m|\alpha}_j = \lceil (pr^{\alpha}_{1,j} + pr^{\alpha}_{2,j} + \cdots + pr^{\alpha}_{n,j})/n \rceil$$

for every j such that $w_j \models \phi$, ensures that for all $w_j \in C$, if there exists a $w_i \in C$ such that $R_\alpha(w_i, w_j) > 0$, then $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 1$, else $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 0$, as stated in Definition 7.1.2.

Belief Update

Let $\Pi(e_h, \alpha, \varsigma)$ be the abbreviation for the term

$$\sum_{j=1}^{n} pr^{\varsigma|\alpha}_j \sum_{i=1}^{n} pr^{\alpha}_{i,j} \omega^{e_h}_i,$$

which is the probability of reaching the belief-state after performing belief update $[\![\alpha + \varsigma]\!]$ at activity-point e_h. And let $BT(e_h, k, \alpha, \varsigma)$ be the abbreviation for the term

$$\frac{pr^{\varsigma|\alpha}_k \sum_{i=1}^{n} pr^{\alpha}_{i,k} \omega^{e_h}_i}{\Pi(e_h, \alpha, \varsigma)},$$

which is the probability of being in world w_k after performing belief update $[\![\alpha + \varsigma]\!]$ at activity-point e_h, where $n = |C|$.

Suppose Σ is $0 \stackrel{\alpha_0,\varsigma_0}{\longrightarrow} e_1 \stackrel{\alpha_1,\varsigma_1}{\longrightarrow} e_2 \cdots \stackrel{\alpha_{z-1},\varsigma_{z-1}}{\longrightarrow} e_z$ and $\Sigma \neq 0$. For every formulae of the form $(\Sigma, \Psi) \in \Gamma$, the following equations are in $SI(\Gamma)$.

$$\omega_k^{e_{h+1}} = BT(e_h, k, \alpha_h, \varsigma_h) \text{ (for } k = 1, 2, \ldots, n \text{ and } h = 0, 1, \ldots, z-1),$$

$$\Pi(e_h, \alpha_h, \varsigma_h) \neq 0 \text{ (for } h = 0, 1, \ldots, z-1)$$

and

$$\omega_1^{e_h} + \omega_2^{e_h} + \cdots + \omega_n^{e_h} = 1 \text{ (for } h = 0, 1, \ldots, z),$$

where e_0 is 0. Observe that the e_h are integers and we enforce the constraint that $e_i < e_j$ iff $i < j$.

Continuity and Belief Literals

For every formula of the form $(\Sigma e, Cont(\alpha, \varsigma)) \in \Gamma$ or $(\Sigma e, \neg Cont(\alpha, \varsigma)) \in \Gamma$,

$$\Pi(e, \alpha, \varsigma) \neq 0, \text{ respectively, } \Pi(e, \alpha, \varsigma) = 0$$

is in $SI(\Gamma)$.

For every formula of the form $(\Sigma e, \mathbf{B}\varphi \bowtie p) \in \Gamma$,

$$c_1\omega_1^e + c_2\omega_2^e + \cdots + c_n\omega_n^e \bowtie p,$$

is in $SI(\Gamma)$, where $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$.

Rewards, Costs and Utilities

For every formula of the form $(\Sigma, \phi \Rightarrow Reward(r)) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg Reward(r)) \in \Gamma$, for every j such that $w_j \models \phi$,

$$R_j = r, \text{ respectively, } R_j \neq r$$

is in $SI(\Gamma)$.

For every formula of the form $(\Sigma, \phi \Rightarrow Cost(\alpha, r)) \in \Gamma$ and $(\Sigma, \phi \Rightarrow \neg Cost(\alpha, r)) \in \Gamma$, for every j such that $w_j \models \phi$,

$$C_j^\alpha = r, \text{ respectively, } C_j^\alpha \neq r$$

is in $SI(\Gamma)$.

Let $RC(\alpha, e) \stackrel{def}{=} \omega_1^e(R_1 - C_1^\alpha) + \omega_2^e(R_2 - C_2^\alpha) + \cdots + \omega_n^e(R_n - C_n^\alpha)$. For every formula of the form $(\Sigma e, \mathbf{U}[\![\alpha]\!] \bowtie q) \in \Gamma$,

$$RC(\alpha, e) \bowtie q$$

is in $SI(\Gamma)$.

To keep track of dependencies between variables in inequalities derived from utility literals of the

form $(\Sigma, \mathbf{U}[\![\alpha]\!]\Lambda \bowtie q)$, we define a *utility tree*. A set of utility trees is induced from a set Δ which is defined as follows (examples follow the formal description). For every formula of the form $(\Sigma e, \mathbf{U}[\![\alpha]\!]\Lambda \bowtie q) \in \Gamma$, let $(e \xrightarrow{\alpha,\varsigma} e^{\varsigma}, \Lambda) \in \Delta$, for every $\varsigma \in \Omega$, where e^{ς} is a fresh integer. Then, for every $(\xi, [\![\alpha]\!]\Lambda) \in \Delta$ (where Λ is not empty), for every $\varsigma \in \Omega$, if $(\xi', \Psi) \in \Delta$ such that $\xi' = \xi \xrightarrow{\alpha,\varsigma} e^{\varsigma'}$, then $(\xi', \Lambda) \in \Delta$, else $(\xi \xrightarrow{\alpha,\varsigma} e^{\varsigma}, \Lambda) \in \Delta$, where e^{ς} is a fresh integer. This finishes the definition of Δ.

Suppose $\Omega = \{\varsigma_1, \varsigma_2\}$ and

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_5]\!] = 88)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] > 61)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_3]\!][\![\alpha_2]\!] < 62)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_4]\!] = 63)$,

$(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] \geq 64)$ and

$(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}[\![\alpha_2]\!][\![\alpha_1]\!] = 65)$

are in some leaf node Γ'. Then $(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_5]\!] = 88)$ is not involved in the definition of Δ', nevertheless, $RC(\alpha_5, 13) = 88$ is in $SI(\Gamma')$.

With respect to the other utility literals,

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24, [\![\alpha_2]\!])$, $(13 \xrightarrow{\alpha_1,\varsigma_2} 25, [\![\alpha_2]\!])$,

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24, [\![\alpha_3]\!][\![\alpha_2]\!])$, $(13 \xrightarrow{\alpha_1,\varsigma_2} 25, [\![\alpha_3]\!][\![\alpha_2]\!])$,

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24, [\![\alpha_4]\!])$, $(13 \xrightarrow{\alpha_1,\varsigma_2} 25, [\![\alpha_4]\!])$,

$(23 \xrightarrow{\alpha_1,\varsigma_1} 26, [\![\alpha_2]\!])$, $(23 \xrightarrow{\alpha_1,\varsigma_2} 27, [\![\alpha_2]\!])$,

$(23 \xrightarrow{\alpha_2,\varsigma_1} 28, [\![\alpha_1]\!])$ and $(23 \xrightarrow{\alpha_2,\varsigma_2} 29, [\![\alpha_1]\!])$

are in Δ'. And due to $(13 \xrightarrow{\alpha_1,\varsigma_1} 24, [\![\alpha_3]\!][\![\alpha_2]\!]), (13 \xrightarrow{\alpha_1,\varsigma_2} 25, [\![\alpha_3]\!][\![\alpha_2]\!]) \in \Delta'$, the following are also in Δ'.

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24 \xrightarrow{\alpha_3,\varsigma_1} 30, [\![\alpha_2]\!])$,

$(13 \xrightarrow{\alpha_1,\varsigma_1} 24 \xrightarrow{\alpha_3,\varsigma_2} 31, [\![\alpha_2]\!])$,

$(13 \xrightarrow{\alpha_1,\varsigma_2} 25 \xrightarrow{\alpha_3,\varsigma_1} 32, [\![\alpha_2]\!])$ and

$(13 \xrightarrow{\alpha_1,\varsigma_2} 25 \xrightarrow{\alpha_3,\varsigma_2} 33, [\![\alpha_2]\!])$.

Note how an activity-point is represented by the same integer (for instance, 24) if and only if it is reached via the same sequence of actions and observations (for instance, $13 \xrightarrow{\alpha_1,\varsigma_1}$).

The set of utility trees is generated from Δ as follows. Δ is partitioned such that $(e \xrightarrow{\alpha,\varsigma} e', \Lambda)$, $(e'' \xrightarrow{\alpha',\varsigma'} e''', \Lambda') \in \Delta$ are in the same partitioning if and only if $e = e''$. Each partitioning represents a unique utility tree with the first activity-point as the root of the tree. For example, one can generate two utility trees from Δ'; one with root 13 and one with root 23. Each activity

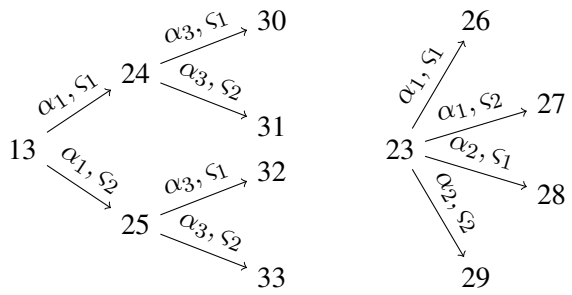


Fig. 7.1: The two utility trees generated from Δ'.

sequence of the members of Δ represents a (sub)path starting at the root of its corresponding tree. Figure 7.1 depicts the two utility trees generated from Δ'.

Before considering the general case, we illustrate the method of generating, from the utility trees in Figure 7.1, the required inequalities which must be in $SI(\Gamma')$.

The formula $(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] > 61) \in \Gamma'$ is represented by

$$RC(\alpha_1, 13) + \Pi(13, \alpha_1, \varsigma_1)RC(\alpha_2, 24) + \Pi(13, \alpha_1, \varsigma_2)RC(\alpha_2, 25) > 61$$

in $SI(\Gamma')$. To generate this inequality, the utility tree rooted at 13 is used: See that α_1 is executed at activity-point 13, α_2 is executed at activity-point 24 if ς_1 is perceived and α_2 is executed at activity-point 25 if ς_2 is perceived. Moreover, the latter two rewards must be weighted by the probabilities of reaching the respective new belief-states/activity-points.

The formula $(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_4]\!] = 63) \in \Gamma'$ is represented by

$$RC(\alpha_1, 13) + \Pi(13, \alpha_1, \varsigma_1)RC(\alpha_4, 24) + \Pi(13, \alpha_1, \varsigma_2)RC(\alpha_4, 25) = 63.$$

in $SI(\Gamma')$. This time, $\underline{\alpha_4}$ is executed at the activity-points 24 and 25.

Next, the utility tree rooted at 23 is used to find the representation of $(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] \geq 64) \in \Gamma'$. Looking at the utility tree, one can work out that

$$RC(\alpha_1, 23) + \Pi(23, \alpha_1, \varsigma_1)RC(\alpha_2, 26) + \Pi(23, \alpha_1, \varsigma_2)RC(\alpha_2, 27) \geq 64$$

must be in $SI(\Gamma')$.

For $(\Sigma \xrightarrow{\alpha',\varsigma'} 23, \mathbf{U}[\![\alpha_2]\!][\![\alpha_1]\!] = 65) \in \Gamma'$,

$$RC(\alpha_2, 23) + \Pi(23, \alpha_2, \varsigma_1)RC(\alpha_1, 28) + \Pi(23, \alpha_1, \varsigma_2)RC(\alpha_1, 29) \geq 64$$

is in $SI(\Gamma')$.

Formula

$$(\Sigma \xrightarrow{\alpha',\varsigma'} 13, \mathbf{U}[\![\alpha_1]\!][\![\alpha_3]\!][\![\alpha_2]\!] < 62) \in \Gamma', \tag{7.1}$$

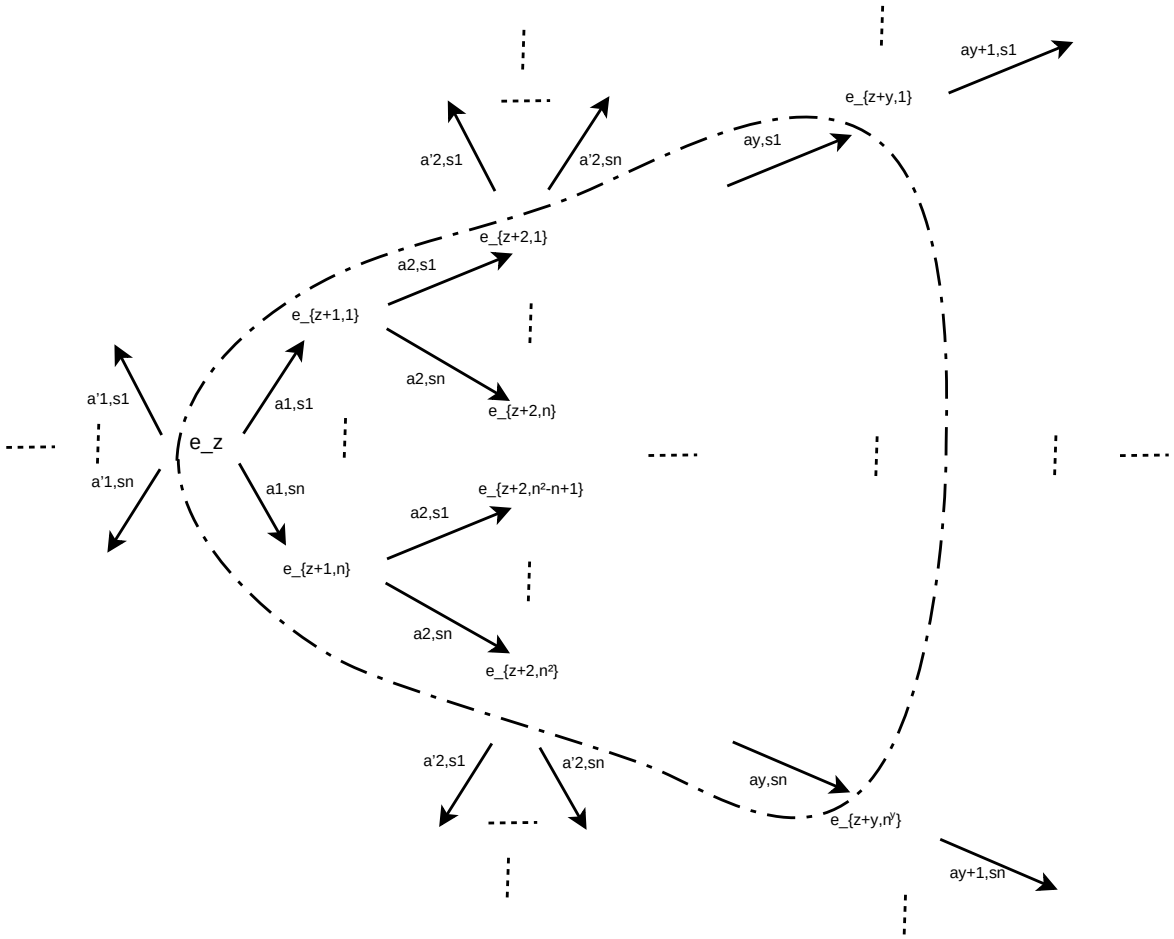


Fig. 7.2: The general form of a utility tree. The enclosed area indicates the subtree corresponding to the general utility literal of (7.2). The root of the tree, e_z, is situated towards the left of the diagram.

is represented by the following inequality.

$$RC(\alpha_1, 13) \begin{array}{l} + \Pi(13, \alpha_1, \varsigma_1)\left(\begin{array}{ll} RC(\alpha_3, 24) & + \Pi(24, \alpha_3, \varsigma_1)\ RC(\alpha_2, 30) \\ & + \Pi(24, \alpha_3, \varsigma_2)\ RC(\alpha_2, 31) \end{array} \right) \\ + \Pi(13, \alpha_1, \varsigma_2)\left(\begin{array}{ll} RC(\alpha_3, 25) & + \Pi(25, \alpha_3, \varsigma_1)\ RC(\alpha_2, 32) \\ & + \Pi(25, \alpha_3, \varsigma_2)\ RC(\alpha_2, 33) \end{array} \right) \end{array} < 62$$

The size of the utility tree rooted at 13 is due to (7.1). Hence, the whole tree is employed to generate the inequality.

Figure 7.2 depicts the general form of a utility tree generated from a utility literal with the form

$$(\Sigma e_z, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_y]\!] \bowtie q). \tag{7.2}$$

In general, for every utility literal of the form (7.2) in leaf node Γ, an inequality of the form shown

below must be in $SI(\Gamma)$ and can be generated from a utility tree of the form depicted in Figure 7.2.

$$
\begin{array}{l}
RC(\alpha_1, e_z) \\
\quad + \Pi(e_z, \alpha_1, \varsigma_1)\left(RC(\alpha_2, e_{z+1,1}) \quad \begin{array}{l} +\Pi(e_{z+2,1}\alpha_2, \varsigma_1) \quad \left(\cdots \right. \\ +\Pi(e_{z+n,n}, \alpha_2, \varsigma_n) \quad \left(\cdots \right. \end{array} \right. \\
\quad + \Pi(e_z, \alpha_1, \varsigma_n)\left(RC(\alpha_2, e_{z+1,n}) \quad \begin{array}{l} +\Pi(e_{z+2,n^2-n+1}, \alpha_2, \varsigma_1) \quad \left(\cdots \right. \\ +\Pi(e_{z+n,n^2}, \alpha_2, \varsigma_n) \quad \left(\cdots \right. \end{array} \right. \\
\quad\quad + \Pi(e_{z+y-2,1}, \alpha_{y-1}, \varsigma_1) RC(\alpha_y, e_{z+y-1,1}) \cdots \Big) \Big) \\
\cdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \bowtie q \\
\quad\quad + \Pi(e_{z+y-2,n^{y-2}}, \alpha_{y-1}, \varsigma_n) RC(\alpha_y, e_{z+y-1,n^{y-1}}) \cdots \Big) \Big)
\end{array}
$$

The value to the left of the $\bowtie$ of this general inequality can be written in a compact form as follows. We define the value of a sequence Λ of y actions, starting at activity-point e_z as

$$
\begin{aligned}
&U([\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_y]\!], e_{z+h-1,x}) \stackrel{def}{=} \\
&\quad RC([\![\alpha_1]\!], e_{z+h-1,x}) + \sum_{\varsigma_i \in \Omega^{\#}} \Pi(e_{z+h-1,x}, \alpha_1, \varsigma) U([\![\alpha_2]\!] \cdots [\![\alpha_y]\!], e_{z+h,i}), \\
&U([\![\alpha_y]\!], e_{z+y-1,x}) \stackrel{def}{=} RC(\alpha_y, e_{z+y-1,x}).
\end{aligned} \tag{7.3}
$$

The inequality which must be in $SI(\Gamma)$ can thus be written as

$$U([\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_y]\!], e_{z,\_}) \bowtie q,$$

where $e_{z,\_} = e_z$ ($\_$ is a dummy value).

Finally (almost), for every activity-point/node e in every utility tree,

$$\omega_1^e + \cdots + \omega_n^e = 1 \in SI(\Gamma),$$

and for every $e \xrightarrow{\alpha,\varsigma} e'$ in every utility tree,

$$\Pi(e, \alpha, \varsigma) = 0 \;||\; \Pi(e, \alpha, \varsigma) \neq 0, \omega_1^{e'} = BT(e, 1, \alpha, \varsigma), \ldots, \omega_n^{e'} = BT(e, n, \alpha, \varsigma) \in SI(\Gamma) \tag{7.4}$$

Statement (7.4) needs explaining:

Until now, whenever it was stated that $\omega_k^{e'} = BT(e, k, \alpha, \varsigma) \in SI(\Gamma)$, the equation matched the occurrence of an update operator $[\![\alpha + \varsigma]\!]$ in a sentence. According to the semantics of $[\![\alpha + \varsigma]\!]$, the new belief-state due to performing α and perceiving ς is reachable; that is, $\Pi(e, \alpha, \varsigma) \neq 0$, where e represents the belief-state before the update. Therefore, matching every $[\![\alpha + \varsigma]\!]$ with $\omega_k^{e'} =$

$BT(e,k,\alpha,\varsigma)$ is appropriate, because the definition of $\omega_k^{e'} = BT(e,k,\alpha,\varsigma)$ implies $\Pi(e,\alpha,\varsigma) \neq 0$. But the occurrence of $e \xrightarrow{\alpha,\varsigma}$ in a utility tree should not necessarily imply the reachability of a belief-state via α,ς from the belief-state represented by e. Nevertheless, if a new belief-state is reachable, then that new belief-state needs to be consistent with the rest of the constraints in the system. Hence, in essence, we are stating that, in the case of utility literals, with respect to the expected utility of the sequence of actions in the literal, IF $\Pi(e,\alpha,\varsigma) \neq 0$ THEN $\omega_k^{e'} = BT(e,k,\alpha,\varsigma) \in SI(\Gamma)$. Unfortunately, one cannot check $\Pi(e,\alpha,\varsigma) \neq 0$ outside the context of the rest of the system of inequalities. So (7.4) is made part of the system in the form $\neg A \vee B$, where $||$ represents $\vee$.

Because (7.4) is not an equation or inequality, it requires a special interpretation to determine the feasibility of the system $SI(\Gamma)$ in which it appears. Let $SI^{-}(\Gamma)$ be $SI(\Gamma)$ without (7.4). Quite simply, $SI(\Gamma)$ is feasible if and only if $SI^{-}(\Gamma) \cup \{\Pi(e,\alpha,\varsigma) = 0\}$ is feasible *or* $SI^{-}(\Gamma) \cup \{\Pi(e,\alpha,\varsigma) \neq 0, \omega_1^{e'} = BT(e,1,\alpha,\varsigma), \ldots, \omega_n^{e'} = BT(e,n,\alpha,\varsigma)\}$ is feasible.

Decidability of Feasibility of Systems of Inequalities

Lemma 7.2.1: Determining whether an SI (as defined in this thesis) is feasible, is decidable.

Proof:
The proof extends the proof of Lemma 6.2.1 with respect to SLAOP. The proof of Lemma 7.2.1 is in the appendix. ■

7.3 Properties of the Decision Procedure

All proofs not given here can be found in the appendix Section A.4.

7.3.1 Soundness

Lemma 7.3.1: Let T be a finished tree. For every node Γ in T: If there exists a structure $\mathcal{D}$ such that for all $(\Sigma,\Phi) \in \Gamma$ there exists a belief-state $b \in P$ and a world $w \in C$ such that $\mathcal{D}bw \models \Phi$, then the (sub)tree rooted at Γ is open.

Theorem 7.3.1: (Soundness) If $\vdash \Psi$ then $\models \Psi$. (Contrapositively, if $\not\models \Psi$ then $\not\vdash \Psi$.)

7.3.2 Completeness

We start with the description of the construction of an SDL structure, given the leaf node Γ of some open leaf node of a finished tree.

Definition 7.3.1: A solution for $SI(\Gamma)$ is a set of values $\{s_{i,j}^{\alpha}, s_j^{\varsigma|\alpha}, sR_i, sC_i^{\alpha}, s\omega_i^{e} \mid \alpha \in \mathcal{A}, \varsigma \in \Omega,\ i,j = 1,2,\ldots,|C|, e = 0$ or e an integer introduced by rule Ξ to a formula appearing in $\Gamma\}$ such that the assignments of these values to the variables (as follows) solves every equation and inequality in $SI(\Gamma)$ simultaneously: $pr_{i,j}^{\alpha} \leftarrow s_{i,j}^{\alpha}, pr_j^{\varsigma|\alpha} \leftarrow s_j^{\varsigma|\alpha}, R_i \leftarrow sR_i, C_i^{\alpha} \leftarrow sC_i^{\alpha}$ and

$\omega_i^e \leftarrow s\omega_i^e$ for all $\alpha \in \mathcal{A}, \varsigma \in \Omega,\ i,j \in \{1,2,\ldots,|C|\}, e = 0$ and e an integer introduced by rule Ξ to a formula appearing in $\Gamma\}$. Let $Z(\Gamma)$ be the set of solutions for $SI(\Gamma)$.

$\mathcal{D} = \langle R, Q, U\rangle$ can be constructed as follows. Let sln be a solution in $Z(\Gamma)$.

- For every action $\alpha \in \mathcal{A}$, the accessibility relation R_α can be constructed as follows. For $i,j = 1,2,\ldots,|C|$, let $R_\alpha(w_i, w_j) = s_{i,j}^\alpha$, where $w_i, w_j \in C^\#$ and $s_{i,j}^\alpha \in sln$.
- For every action $\alpha \in \mathcal{A}$, the perceivability relation Q_α can be constructed as follows. For $j = 1,2,\ldots,|C|$, let $Q_\alpha(w_j, \varsigma) = s_j^{\varsigma|\alpha}$, where $w_j \in C^\#, \varsigma \in \Omega$ and $s_j^{\varsigma|\alpha} \in sln$.
- For $i = 1,2,\ldots,|C|$, let $Re(w_i) = sR_i$, where $w_i \in C^\#$ and $sR_i \in sln$. For every action $\alpha \in \mathcal{A}$ and $i = 1,2,\ldots,|C|$, let $Co_\alpha(w_i) = sC_i^\alpha$, where $w_i \in C^\#$ and $sC_i^\alpha \in sln$. Let $U = \langle Re, Co\rangle$ such that $Co = \{(\alpha, Co_\alpha) \mid \alpha \in \mathcal{A}\}$.

Lemma 7.3.2: $\mathcal{S}$ is an SDL structure.

Lemma 7.3.3: Let Γ be an open leaf node of a finished tree. We know that $Z(\Gamma)$ is not empty. If $\mathcal{D}$ is constructed as described above, then for all $(\Sigma, \Psi) \in \Gamma$, there exists a b and a w such that $\mathcal{D}bw \models \Psi$.

Theorem 7.3.2: (Completeness) If $\models \Psi$ then $\vdash \Psi$. (Contrapositively, if $\nvdash \Psi$ then $\not\models \Psi$.)

7.3.3 Termination

Definition 7.3.2: Let Ψ' be a strict sub-part of Ψ and let $(\Sigma, \Psi) \in \Gamma$. A tableau rule has the *sub-formula property* if and only if the new node(s) Γ' created by the application of the rule, contains (Σ', Ψ') or $(\Sigma', \neg\Psi')$ for some Σ', where $(\Sigma', \Psi') \notin \Gamma$, respectively, $(\Sigma', \neg\Psi') \notin \Gamma$.

Lemma 7.3.4: A formula of the form $(\Sigma, \varphi \Rightarrow \Phi)$ can cause only a finite number of tableau rule applications.

Proof:
Recall that due to a preprocessing step, Φ is in CNF. Let $(\Sigma, \varphi \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m)$ be in some node Γ, where Φ_i (for $i = 1,2,\ldots,m$) is a disjunction of literals (i.e., $\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m$ is in CNF). After successive applications of rule $\Rightarrow \wedge$, $(\Sigma, \varphi \Rightarrow \Phi_1), (\Sigma, \varphi \Rightarrow \Phi_2), \ldots, (\Sigma, \varphi \Rightarrow \Phi_m) \in \Gamma'$, where Γ' is a descendant of Γ. After successive applications of rule $\delta \Rightarrow$,

$$
\begin{gathered}
(\Sigma, \delta_1 \Rightarrow \Phi_1), (\Sigma, \delta_2 \Rightarrow \Phi_1), \ldots, (\Sigma, \delta_n \Rightarrow \Phi_1),\\
(\Sigma, \delta_1 \Rightarrow \Phi_2), (\Sigma, \delta_2 \Rightarrow \Phi_2), \ldots, (\Sigma, \delta_n \Rightarrow \Phi_2),\\
\vdots\\
(\Sigma, \delta_1 \Rightarrow \Phi_m), (\Sigma, \delta_2 \Rightarrow \Phi_m), \ldots, (\Sigma, \delta_n \Rightarrow \Phi_m) \in \Gamma'',
\end{gathered}
$$

where Γ'' is a descendant of Γ' and $\delta_i \in cpt(\varphi)$. If Φ_j for some $j \in \{1,2,\ldots,m\}$ is a literal, then no tableau rule is applicable to $(\Sigma, \delta_1 \Rightarrow \Phi_j), (\Sigma, \delta_2 \Rightarrow \Phi_j), \ldots,$ nor $(\Sigma, \delta_n \Rightarrow \Phi_j)$. Else, (for all Φ_j which are not literals, $j \in \{1,2,\ldots,m\}$), after successive applications of rule $\Rightarrow \vee$ and rule $\vee$, new decendent nodes are created containing formulae of the form $(\Sigma, \delta \Rightarrow \Phi)$, where δ is definitive and Φ is a literal. No tableau rule is applicable to formulae of the form $(\Sigma, \delta \Rightarrow \Phi)$.

Clearly, no matter the order in which rules $\Rightarrow \wedge, \delta \Rightarrow, \Rightarrow \vee$ and $\vee$ are applied, the rules can be applied only a finite number of times due to $(\Sigma, \varphi \Rightarrow \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m) \in \Gamma$. ■

Lemma 7.3.5: A tree for any formula $\Psi \in \mathcal{L}_{SDL}$ becomes saturated. That is, the tableau phase terminates.

Proof:
We can divide all the tableau rules into three categories: (i) those which add $(\Sigma, \bot)$ to the new node, (ii) those with the subformula property and (iii) those from the list $\langle \Rightarrow \wedge, \delta \Rightarrow, \Rightarrow \vee, \neg\Xi, \neg\mathbf{B}, \neg\mathbf{U} \rangle$. Category-(i) rules never cause rules to become applicable later. As a direct consequence of the subformula property and sentences being finite, every category-(ii) rule must eventually become inapplicable.

Rules $\neg\mathbf{B}$ and $\neg\mathbf{U}$ are applied only once to any formula and never to a formula added to the new node due to the rule.

Rule $\neg\Xi$ adds a formula of the form $(\Sigma, \neg Cont(\alpha, \varsigma) \vee [\![\alpha + \varsigma]\!] \neg\Psi)$ to the new node. After applying rule $\vee$, $(\Sigma, \neg Cont(\alpha, \varsigma))$ causes no rule application and $(\Sigma, [\![\alpha + \varsigma]\!] \neg\Psi)$ is a category-(ii) rule.

Rules $\Rightarrow \wedge, \delta \Rightarrow$ and $\Rightarrow \vee$ are only applied to formulae of the form $(\Sigma, \varphi \Rightarrow \Phi)$. By Lemma 7.3.4, there will be a finite number of rule applications to formulae of the form $(\Sigma, \varphi \Rightarrow \Phi)$.

Therefore, all rules eventually become inapplicable, and it follows that any tree (for any formula) would become saturated. ■

Theorem 7.3.3: The decision procedure for SDL terminates.

Proof:
Due to Lemma 7.3.5, the tableau phase terminates (with a finite number of branches).

In the SI phase: for each open leaf node of a tree for some $\Psi \in \mathcal{L}_{SDL}$, the feasibility of an SI is sought once for Γ, where Γ is the leaf node of the branch. Hence, the feasibility of an SI is sought a finite number of times in the SI phase.

By Lemma 7.2.1, determining the feasibility of an SI terminates and the SI phase thus terminates. ■

Corollary 7.3.1: The validity problem for SDL is decidable.

Proof:
Because the procedure is sound and complete, it will be decidable if it always terminates, which, by Theorem 7.3.3, it does. ■

7.4 Specifying Domains with SDL

The framework presented here should be viewed as providing guidance; the knowledge engineer should adapt the framework as necessary for the particular domain being modeled. On the practical side, in the context of SDL, the domain of interest can be divided into five parts:

7.4.1 Static Laws

Static laws (denoted as the set SL) have the form $\phi \Rightarrow \varphi$, where ϕ and φ are propositional sentences, and ϕ is the condition under which φ is always satisfied. They are the basic laws and facts of the domain. For instance, "A roof is above a floor", "A full battery allows me at most four hours of operation", "I sink in liquids" and "The charging station is in sector 14". Such static laws cannot be explicitly stated in POMDPs; the information must be encoded in each relevant state.

7.4.2 Action Rules

Action rules (denoted as the set AR) must be specified. We identify three kinds of action rules.

The basic kind is the *effect axiom*. For every action α, effect axioms typically take the form

$$\begin{aligned}
\phi_1 &\Rightarrow [\alpha]\varphi_{11} = p_{11} \wedge \cdots \wedge [\alpha]\varphi_{1n} = p_{1n} \\
\phi_2 &\Rightarrow [\alpha]\varphi_{21} = p_{21} \wedge \cdots \wedge [\alpha]\varphi_{2n} = p_{2n} \\
&\vdots \\
\phi_j &\Rightarrow [\alpha]\varphi_{j1} = p_{j1} \wedge \cdots \wedge [\alpha]\varphi_{jn} = p_{jn},
\end{aligned}$$

where (i) the sum of transition probabilities $p_{i1}, \ldots, p_{in}$ of any rule i must lie in the range $[0, 1]$ (preferably 1), (ii) for every i, for any pair of effects φ_{ik} and $\varphi_{ik'}$, $\varphi_{ik} \wedge \varphi_{ik'} \equiv \bot$ and (iii) for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$.

Allowing the probabilities (mentioned in point (i)) to sum up to less than 1 is a generalization of the similar constraint for SLAOP specifications of effect axioms. The knowledge engineer must keep in mind that if the transition probabilities do not sum to 1, the specification is incomplete. Suppose, for instance, that for rule i, $p_{i1} + \cdots + p_{in} < 1$. Then one or more transitions from a ϕ_i-world has not been mentioned and some logical inferences will not be possible.

The second kind of action rule is the *frame axiom*. The knowledge engineer may know that under certain conditions, an action will not change the truth value of certain fluents. A frame axiom has the form

$$\phi \Rightarrow (f \rightarrow [\alpha]f) \wedge (\neg f \rightarrow [\alpha]\neg f), \tag{7.5}$$

where ϕ is the condition under which the value of fluent f never changes due to the execution of action α. The benefit of having a frame axiom like (7.5) is that the knowledge engineer need not mention f anywhere on the RHS of effect axiom

$$\phi_i \Rightarrow [\alpha]\varphi_{i1} = p_{i1} \wedge \cdots \wedge [\alpha]\varphi_{in} = p_{in}$$

if ϕ_i entails ϕ. This is only one suggestion to deal with *invariance* or *inertia* of fluent values. The use of invariance predicates, as presented in Chapter 5, constitutes our solution to the frame problem in our latter three logics.

The third kind of action rule is the *inexecutability axiom*. We shall assume that the union of effect and frame axioms is complete, that is, that the knowledge engineer intends that the conditions of these axioms are the only conditions under which the actions can be executed. Note that $[\alpha]\top > 0$

implies that α is executable and that $\neg([\alpha]\top > 0)$ implies that α is inexecutable. Therefore, if there is an effect axiom or frame axiom for α with condition ϕ, then one can assume the presence of an executability axiom $\phi \Rightarrow [\alpha]\top > 0$.

However, we must still specify that an action is inexecutable when none of the effect axiom conditions or frame axiom conditions is met. Hence, the following *inexecutability* axiom is assumed present.[4]

$$\neg(\phi_1 \vee \cdots \vee \phi_j \vee \phi_{j+1} \vee \cdots \vee \phi_{j+k}) \Rightarrow [\alpha]\top = 0$$

where $\phi_1, \ldots, \phi_j$ are the conditions of the effect axioms for α and $\phi_{j+1} \vee \cdots \vee \phi_{j+k}$ are the conditions of the frame axioms for α.

7.4.3 Perception Rules

Perception rules (denoted as the set PR) must be specified. Let $E(\alpha) = \{\varphi_{11}, \varphi_{12}, \ldots, \varphi_{21}, \varphi_{22}, \ldots, \varphi_{jn}\}$ be the set of all effects of action α executed under all executable conditions. For every action α, perception rules typically take the form

$$\begin{aligned}
\phi_1 &\Rightarrow (\varsigma_{11} \mid \alpha) = p_{11} \wedge \cdots \wedge (\varsigma_{1m} \mid \alpha) = p_{1n} \\
\phi_2 &\Rightarrow (\varsigma_{21} \mid \alpha) = p_{21} \wedge \cdots \wedge (\varsigma_{2m} \mid \alpha) = p_{2n} \\
&\vdots \\
\phi_k &\Rightarrow (\varsigma_{k1} \mid \alpha) = p_{k1} \wedge \cdots \wedge (\varsigma_{km} \mid \alpha) = p_{km},
\end{aligned}$$

where (i) the sum of perception probabilities $p_{i1}, \ldots, p_{im}$ of any rule i must lie in the range $[0, 1]$ (preferably 1), (ii) for any pair of conditions ϕ_i and $\phi_{i'}$, $\phi_i \wedge \phi_{i'} \equiv \bot$ and (iii) $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_k \equiv \bigvee_{\varphi \in E(\alpha)} \varphi$.

If the sum of perception probabilities $p_{i1}, \ldots, p_{im}$ of any rule i is 1, then any observations not mentioned in rule i are automatically *unperceivable* in a ϕ_i-world. However, in the case that the sum is not 1, this deduction about unperceivability cannot be made. Then the knowledge engineer should keep in mind that a perception rule of the form

$$\phi_i \rightarrow \cdots \wedge (\varsigma \mid \alpha) = 0 \wedge \cdots$$

implies that ς is unperceivable in a ϕ_i-world given that the world is reachable via α, and if this information is available, it should be included.

[4] Inexecutability axioms are also called *condition closure* axioms.

7.4.4 Utility Rules

Utility rules (denoted as the set UR) must be specified. Utility rules typically take the form

$$\begin{aligned}\phi_1 &\Rightarrow Reward(r_1)\\ \phi_2 &\Rightarrow Reward(r_2)\\ &\vdots\\ \phi_j &\Rightarrow Reward(r_j),\end{aligned}$$

meaning that in all worlds where ϕ_i is satisfied, the agent gets r_i units of reward. And for every action α,

$$\begin{aligned}\phi_1 &\Rightarrow Cost(\alpha, r_1)\\ \phi_2 &\Rightarrow Cost(\alpha, r_2)\\ &\vdots\\ \phi_j &\Rightarrow Cost(\alpha, r_j),\end{aligned}$$

meaning that the cost for performing α in a world where ϕ_i is satisfied is r_i units. The conditions are disjoint as for action and perception rules.

7.4.5 Initial Belief-states

The agent's initial belief-state IB must be specified, that is, a specification of the worlds the agent should believe it is in when it becomes active, and probabilities associated with those worlds. For instance,

$$\mathbf{B}(f \wedge h) = 0.35 \ \wedge \ \mathbf{B}(f \wedge \neg h) = 0.35 \ \wedge \ \mathbf{B}(\neg f \wedge h) = 0.2 \ \wedge \ \mathbf{B}(\neg f \wedge \neg h) = 0.1$$

is a specification of a particular initial belief-state. And

$$\mathbf{B}f < 0.7 \ \wedge \ \mathbf{B}(\neg f \wedge h) \leq 0.3 \ \wedge \ \mathbf{B}(\neg f \wedge \neg h) = 0.1$$

is a specification of a class of initial belief-states, all those satisfying the constraints. In general, an initial belief-state specification should have the form

$$\mathbf{B}\varphi_1 \bowtie p_1 \ \wedge \ \mathbf{B}\varphi_2 \bowtie p_2 \ \wedge \ \ldots \ \wedge \ \mathbf{B}\varphi_n \bowtie p_n,$$

where (i) $\bowtie \in \{<, \leq, =, \geq, >\}$ and (ii) the φ_i are mutually exclusive propositional sentences (i.e., for all $1 \leq i, j \leq n$ s.t. $i \neq j$, $\varphi_i \wedge \varphi_j \equiv \bot$). The above may or may not be a full specification. For a *full/complete* specification of a *particular* initial belief-state, a sentence of the following form can be provided.

$$\mathbf{B}\varphi_1 = p_1 \ \wedge \ \mathbf{B}\varphi_2 = p_2 \ \wedge \ \ldots \ \wedge \ \mathbf{B}\varphi_n = p_n,$$

where (i) the φ_i are mutually exclusive propositional sentences and (ii) $p_1 + p_2 + \ldots + p_n = 1$.

The union of SL, AR, PR and UR is referred to as an agent's *background knowledge* and is denoted BK.

In practical terms, the question to be answered in SDL is whether $BK \models IB \rightarrow \Theta^{-}$ holds, where $BK \subset \mathcal{L}_{SDL}$, IB is as described above, and $\Theta^{-} \in \mathcal{L}^{\not\Rightarrow}_{SDL}$ is some sentence of interest. (Recall that $\mathcal{L}^{\not\Rightarrow}_{SDL}$ is the subset of formulae of $\mathcal{L}_{SDL}$ excluding formulae containing subformulae of the form $\varphi \Rightarrow \Phi$.)

7.5 Using Entailment in SDL

To get a better feeling for the expressivity of SDL and its decision procedure, we work out a few examples. They are all based on the oil-drinking scenario. The set of fluents is $\mathcal{F} = \{\texttt{full}, \texttt{holding}\}$ abbreviated to f and h. Let $w_1 \models f \wedge h$, $w_2 \models f \wedge \neg h$, $w_3 \models \neg f \wedge h$ and $w_4 \models \neg f \wedge \neg h$. The set of actions is $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{weigh}\}$ abbreviated to g, d and w. The set of observations is $\Omega = \{\texttt{obsNil}, \texttt{obsLight}, \texttt{obsMedium}, \texttt{obsHeavy}\}$ abbreviated to oN, oL, oM and oH.

A complete specification of the POMDP model is provided as the basis for the examples in this section. The background knowledge base BK contains the following laws.

Transitions

- $\neg h \Rightarrow [g](f \wedge h) = 0.8 \wedge [g](\neg f \wedge h) = 0.1 \wedge [g](\neg f \wedge \neg h) = 0.1$
- $h \Rightarrow [g]\top = 0$
- $h \Rightarrow [d](\neg f \wedge h) = 0.95 \wedge [d](\neg f \wedge \neg h) = 0.05$
- $\neg h \Rightarrow [d]\top = 0$
- $f \wedge h \Rightarrow [w](f \wedge h) = 1$
- $f \wedge \neg h \Rightarrow [w](f \wedge \neg h) = 1$
- $\neg f \wedge h \Rightarrow [w](\neg f \wedge h) = 1$
- $\neg f \wedge \neg h \Rightarrow [w](\neg f \wedge \neg h) = 1$

Perceptions

- $\top \Rightarrow (oN \mid g) = 1 \wedge (oN \mid d) = 1$
- $f \wedge h \Rightarrow (oL \mid w) = 0.1 \wedge (oM \mid w) = 0.2 \wedge (oH \mid w) = 0.7$
- $\neg f \wedge h \Rightarrow (oL \mid w) = 0.5 \wedge (oM \mid w) = 0.3 \wedge (oH \mid w) = 0.2$
- $\neg h \Rightarrow (\forall v^{\varsigma}) \neg (v^{\varsigma} = oN) \rightarrow (v^{\varsigma} \mid w) = \frac{1}{3}$

Utility

- $f \Rightarrow Reward(0)$
- $\neg f \wedge h \Rightarrow Reward(10)$
- $\neg f \wedge \neg h \Rightarrow Reward(-5)$
- $\top \Rightarrow (\forall v^{\alpha})(v^{\alpha} = g \vee v^{\alpha} = d) \rightarrow Cost(v^{\alpha}, 1)$
- $f \Rightarrow Cost(w, 2)$
- $\neg f \Rightarrow Cost(w, 0.8)$

First Example

For the first example, we determine whether BK entails

$$\mathbf{B}(f \wedge h) = 0.35 \wedge \mathbf{B}(f \wedge \neg h) = 0.35 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1 \\ \rightarrow [\![g + oN]\!][\![w + oM]\!]\mathbf{B}h > 0.85.$$

Notice that the initial belief-state is fully specified.

For the tableau phase, the trunk is thus $(0, \bigwedge_{\kappa \in BK} \kappa \wedge \mathbf{B}(f \wedge h) = 0.35 \wedge \mathbf{B}(f \wedge \neg h) = 0.35 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1 \wedge \neg [\![g + oN]\!][\![w + oM]\!]\mathbf{B}h > 0.85)$.

Rule $\wedge$ yields

$$(0, \kappa), \ldots, (0, \kappa'), (0, \mathbf{B}(f \wedge h) = 0.35), (0, \mathbf{B}(f \wedge \neg h) = 0.35), (0, \mathbf{B}(\neg f \wedge h) = 0.2), \\ (0, \mathbf{B}(\neg f \wedge \neg h) = 0.1), (0, \neg [\![g + oN]\!][\![w + oM]\!]\mathbf{B}h > 0.85) \in \Gamma_1^0,$$

where $\kappa, \ldots, \kappa' \in BK$. Rule $\neg\Xi$ yields

$$(0, \neg Cont(g, oN) \vee [\![g + oN]\!] \neg [\![w + oM]\!]\mathbf{B}h > 0.85) \in \Gamma_2^0.$$

Then rule $\vee$ yields

$$(0, \neg Cont(g, oN)) \in \Gamma_0^1 \text{ and } (0, [\![g + oN]\!] \neg [\![w + oM]\!]\mathbf{B}h > 0.85) \in \Gamma_3^0.$$

We shall deal with the subtree rooted at Γ_3^0 later.

Note that

$$(0, \neg h \Rightarrow [g](f \wedge h) = 0.8 \wedge [g](\neg f \wedge h) = 0.1 \wedge [g](\neg f \wedge \neg h) = 0.1) \in \Gamma_1^0 \subset \Gamma_0^1.$$

Hence, by rule $\Rightarrow \wedge$,

$$(0, \neg h \Rightarrow [g](f \wedge h) = 0.8), (0, \neg h \Rightarrow [g](\neg f \wedge h) = 0.1), (0, \neg h \Rightarrow [g](\neg f \wedge \neg h) = 0.1) \in \Gamma_1^1.$$

Also note that

$$(0, \top \Rightarrow (oN \mid g) = 1 \wedge (oN \mid d) = 1) \in \Gamma_1^1,$$

and again by rule $\Rightarrow \wedge$,

$$(0, \top \Rightarrow (oN \mid g) = 1), (0, \top \Rightarrow (oN \mid d) = 1) \in \Gamma_2^1.$$

Assume that the tree saturates and that Γ' is any open leaf node of the (sub)tree rooted at Γ_2^1. Then in the SI phase, due to $(0, \neg h \Rightarrow [g](f \wedge h) = 0.8), (0, \top \Rightarrow (oN \mid g) = 1), (0, \neg h \Rightarrow (oL \mid w) = \frac{1}{3}), (0, \neg h \Rightarrow (oM \mid w) = \frac{1}{3}), (0, \neg h \Rightarrow (oH \mid w) = \frac{1}{3}) \in \Gamma'$, the following equations are in $SI(\Gamma')$.

$$\begin{aligned} pr_{2,1}^{g} &= 0.8 \\ pr_{4,1}^{g} &= 0.8 \\ pr_{2,1}^{g} + pr_{2,2}^{g} + pr_{2,3}^{g} + pr_{2,4}^{g} &= 1 \\ pr_{4,1}^{g} + pr_{4,2}^{g} + pr_{4,3}^{g} + pr_{4,4}^{g} &= 1 \end{aligned}$$

$$\begin{aligned} pr_{1}^{oN|g} &= 1 \\ pr_{2}^{oN|g} &= 1 \\ pr_{3}^{oN|g} &= 1 \\ pr_{4}^{oN|g} &= 1 \end{aligned}$$

$$\begin{aligned} pr_{1}^{oN|g} + pr_{1}^{oL|g} + pr_{1}^{oM|g} + pr_{1}^{oH|g} &= 1 \\ pr_{2}^{oN|g} + pr_{2}^{oL|g} + pr_{2}^{oM|g} + pr_{2}^{oH|g} &= 1 \\ pr_{3}^{oN|g} + pr_{3}^{oL|g} + pr_{3}^{oM|g} + pr_{3}^{oH|g} &= 1 \\ pr_{4}^{oN|g} + pr_{4}^{oL|g} + pr_{4}^{oM|g} + pr_{4}^{oH|g} &= 1 \end{aligned}$$

$$\begin{aligned} pr_{1}^{oL|w} &= 0.1 \\ pr_{3}^{oL|w} &= 0.5 \\ pr_{1}^{oM|w} &= 0.2 \\ pr_{3}^{oM|w} &= 0.3 \\ pr_{1}^{oH|w} &= 0.7 \\ pr_{3}^{oH|w} &= 0.2 \end{aligned}$$

$$\begin{aligned} pr_2^{oL|w} &= \tfrac{1}{3} \\ pr_4^{oL|w} &= \tfrac{1}{3} \\ pr_2^{oM|w} &= \tfrac{1}{3} \\ pr_4^{oM|w} &= \tfrac{1}{3} \\ pr_2^{oH|w} &= \tfrac{1}{3} \\ pr_4^{oH|w} &= \tfrac{1}{3} \end{aligned}$$

$$\begin{aligned} pr_1^{oN|w} + pr_1^{oL|w} + pr_1^{oM|w} + pr_1^{oH|w} &= 1 \\ pr_2^{oN|w} + pr_2^{oL|w} + pr_2^{oM|w} + pr_2^{oH|w} &= 1 \\ pr_3^{oN|w} + pr_3^{oL|w} + pr_3^{oM|w} + pr_3^{oH|w} &= 1 \\ pr_4^{oN|w} + pr_4^{oL|w} + pr_4^{oM|w} + pr_4^{oH|w} &= 1 \end{aligned}$$

Also

$$\omega_1^0 + \omega_2^0 + \omega_3^0 + \omega_4^0 = 1 \in SI(\Gamma').$$

And due to $(0, \neg Cont(g, oN)), (0, \mathbf{B}(f \wedge h) = 0.35), (0, \mathbf{B}(f \wedge \neg h) = 0.35), (0, \mathbf{B}(\neg f \wedge h) = 0.2), (0, \mathbf{B}(\neg f \wedge \neg h) = 0.1) \in \Gamma'$,

$$\begin{aligned} \sum_{j=1}^{n} pr_j^{oN|g} \sum_{i=1}^{n} pr_{i,j}^{g} \omega_i^0 &= 0 \qquad (7.6) \\ \omega_1^0 &= 0.35 \\ \omega_2^0 &= 0.35 \\ \omega_3^0 &= 0.2 \\ \omega_4^0 &= 0.1, \end{aligned}$$

respectively, are in $SI(\Gamma')$.

Now $SI(\Gamma')$ is infeasible: No term in (7.6) may be greater than zero, for example, $pr_1^{oN|g} \times pr_{2,1}^{g} \times \omega_2^0$ must equal zero, but $pr_1^{oN|g} = 1$ and $pr_{2,1}^{g} = 0.8$ and $\omega_2^0 = 0.35$. Therefore, the subtree rooted at Γ_0^1 is closed.

Coming back to Γ_3^0, due to rule Ξ,

$$(0 \xrightarrow{g,oN} 1, \neg[\![w + oM]\!]\mathbf{B}h > 0.85) \in \Gamma_4^0.$$

Then by the application of rule $\neg\Xi$ and then rule $\vee$,

$$(0 \xrightarrow{g,oN} 1, \neg[\![w + oM]\!]) \in \Gamma_0^2 \text{ and } (0 \xrightarrow{g,oN} 1, [\![w + oM]\!]\neg\mathbf{B}h > 0.85) \in \Gamma_6^0.$$

The case with the subtree rooted at Γ_0^2 is similar to the case above where the subtree is rooted at Γ_0^1. The subtree rooted at Γ_0^2 also closes. Rule Ξ applied to $(0 \xrightarrow{g,oN} 1, [\![w + oM]\!]\neg\mathbf{B}h > 0.85) \in \Gamma_6^0$

yields

$$(0 \xrightarrow{g,oN} 1 \xrightarrow{w,oM} 2, \neg\mathbf{B}h > 0.85) \in \Gamma_7^0$$

and then rule $\neg\mathbf{B}$ yields

$$(0 \xrightarrow{g,oN} 1 \xrightarrow{w,oM} 2, \mathbf{B}h \leq 0.85) \in \Gamma_8^0.$$

Still assuming that the tree saturates, and assume that Γ'' is any open leaf node of the (sub)tree rooted at Γ_8^0. Then in the SI phase, due to $(0, \mathbf{B}(f \wedge h) = 0.35), (0, \mathbf{B}(f \wedge \neg h) = 0.35), (0, \mathbf{B}(\neg f \wedge h) = 0.2), (0, \mathbf{B}(\neg f \wedge \neg h) = 0.1), (0 \xrightarrow{g,oN} 1 \xrightarrow{w,oM} 2, \mathbf{B}h \leq 0.85) \in \Gamma''$, the following equations are in $SI(\Gamma'')$.

$$\begin{aligned}
\omega_1^0 &= 0.35 \\
\omega_2^0 &= 0.35 \\
\omega_3^0 &= 0.2 \\
\omega_4^0 &= 0.1
\end{aligned}$$

$$\begin{aligned}
\omega_1^0 + \omega_2^0 + \omega_3^0 + \omega_4^0 &= 1 \\
\omega_1^2 + \omega_3^2 &\leq 0.85 \\
\omega_1^1 + \omega_2^1 + \omega_3^1 + \omega_4^1 &= 1 \\
\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 &= 1
\end{aligned}$$

$$\begin{aligned}
\omega_1^1 &= BT(0, 1, g, oN) \\
\omega_2^1 &= BT(0, 2, g, oN) \\
\omega_3^1 &= BT(0, 3, g, oN) \\
\omega_4^1 &= BT(0, 4, g, oN) \\
\omega_1^2 &= BT(1, 1, w, oM) \\
\omega_2^2 &= BT(1, 2, w, oM) \\
\omega_3^2 &= BT(1, 3, w, oM) \\
\omega_4^2 &= BT(1, 4, w, oM)
\end{aligned}$$

It turns out that ω_1^2 is constrained to equal 0.72973 and ω_3^2 is constrained to equal 0.12162, and $0.72973 + 0.12162 = 0.85135 > 0.85$. Therefore, the subtree rooted at Γ_0^1 is closed.

Therefore, the whole tree is closed and the initial entailment query holds.

Second Example

For this example, the same background knowledge base is used, but the initial belief-state is not fully specified. We shall determine whether BK entails

$$\begin{aligned}&\mathbf{B}f = 0.7 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1\\ &\rightarrow [\![g + oN]\!][\![w + oM]\!]\mathbf{B}h > 0.85.\end{aligned}$$

The tree for $(0, \bigwedge_{\kappa \in BK} \kappa \wedge \mathbf{B}f = 0.7 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1 \wedge \neg[\![g + oN]\!][\![w + oM]\!]\mathbf{B}h > 0.85)$ is very similar to the one generated in the first example. The difference is in the subtree dealing with the belief literals (the subtree rooted at Γ^0_3 in the example above). Assume that the tree saturates and assume that Γ'' is any open leaf node of the subtree under consideration. Now, instead of $\omega^0_1 = 0.35$ and $\omega^0_2 = 0.35$ in $SI(\Gamma'')$, we have $\omega^0_1 + \omega^0_2 = 0.7 \in SI(\Gamma'')$. Observe that this is not a full specification of the belief-state, because ω^0_1 and ω^0_2 may take any values in $[0, 1]$ as long as $\omega^0_1 + \omega^0_2 = 0.7$.

Let the initial belief-state $b^0 = \{(w_1, x), (w_2, 0.7 - x), (w_3, 0.2), (w_4, 0.1)\}$, where $x \in [0, 0.7]$. According to our calculations, the belief-state after the updates $[\![g + N]\!]$ and $[\![w + M]\!]$ is $b^2 = \{(w_1, \frac{0.128-0.16x}{0.179-0.223x}), (w_2, 0), (w_3, \frac{0.024-0.03x}{0.179-0.223x}), (w_4, \frac{0.0267-0.0333}{0.179-0.223x})\}$. The system of inequalities will enforce b^2, that is,

$$\begin{aligned}\omega^2_1 &= \frac{0.128 - 0.16x}{0.179 - 0.223x}\\ \omega^2_2 &= 0\\ \omega^2_3 &= \frac{0.024 - 0.03x}{0.179 - 0.223x}\\ \omega^2_4 &= \frac{0.0267 - 0.0333}{0.179 - 0.223x},\end{aligned}$$

where $x \in [0, 0.7]$ will be enforced by the system. Furthermore, $(0 \xrightarrow{g,oN} 1 \xrightarrow{w,oM} 2, \mathbf{B}h \leq 0.85) \in \Gamma''$ causes $\omega^2_1 + \omega^2_3 \leq 0.85$ to be in $SI(\Gamma'')$. Hence, $\frac{0.128-0.16x}{0.179-0.223x} + \frac{0.024-0.03x}{0.179-0.223x}$ must be less than or equal to 0.85. In other words, it is required that

$$\frac{0.152 - 0.19x}{0.179 - 0.223x} \leq 0.85. \tag{7.7}$$

But one can determine that there is no value for $x \in [0, 0.7]$ which will make (7.7) true. The system $SI(\Gamma'')$ is thus infeasible, the tree closes and the initial entailment query holds.

We draw the reader's attention to the fact that sensible entailments can be queried, even with a *partially* specified initial belief-state.

Third Example

This time we provide a complete specification of the initial belief-state again, as in the first example, but we under-specify the perception probabilities. Suppose that instead of the law $f \wedge h \Rightarrow (oL \mid w) = 0.1 \wedge (oM \mid w) = 0.2 \wedge (oH \mid w) = 0.7 \in BK$, we have only

$f \wedge h \Rightarrow (oH \mid w) = 0.7 \in BK'$. (That is, we modify BK to become BK'.)

From a system of inequalities generated from the entailment query and modified background knowledge, one can determine/calculate that the degree of belief in h (`holding`) after the updates $[\![g + oN]\!]$ and $[\![w + oM]\!]$ is $\frac{0.\overline{81}x+0.0\overline{27}}{0.\overline{81}x+0.\overline{06}}$, where $x \in [0, 0.3]$, is the probability of perceiving the the oil-can has medium weight at the world where the can is full and the robot is holding it. One can determine that, $\frac{0.\overline{81}x+0.0\overline{27}}{0.\overline{81}x+0.\overline{06}} \leq 0.85$ when $x \leq 0.1976$. The system $SI(\Gamma'')$ is thus feasible, the tree is open and BK' does not entail

$$\mathbf{B}(f \wedge h) = 0.35 \wedge \mathbf{B}(f \wedge \neg h) = 0.35 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1$$
$$\rightarrow [\![g + oN]\!][\![w + oM]\!]\mathbf{B}h > 0.85. \tag{7.8}$$

Now, because $x \leq 0.1976$, the system

$$\frac{0.\overline{81}x + 0.0\overline{27}}{0.\overline{81}x + 0.\overline{06}} \leq 0.85$$
$$x \geq 0.2$$
$$x \leq 0.3$$

is infeasible. So if BK'' is BK' with the law $f \wedge h \Rightarrow (oM \mid w) \geq 0.2$ added to it, the tree closes and BK'' entails (7.8). Observe that (7.8) goes from 'not entailed' to 'entailed' by adding a little more information; while the POMDP model remains incompletely specified.

Fourth Example

Here we shall determine whether BK entails

$$\mathbf{B}(f \wedge h) = 0.35 \wedge \mathbf{B}(f \wedge \neg h) = 0.35 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1$$
$$\rightarrow [\![g + oN]\!]\mathbf{U}[\![d]\!][\![d]\!] \leq 7.$$

For the tableau phase, the trunk is thus $(0, \bigwedge_{\kappa \in BK} \kappa \wedge \mathbf{B}(f \wedge h) = 0.35 \wedge \mathbf{B}(f \wedge \neg h) = 0.35 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1 \wedge \neg[\![g + oN]\!]\mathbf{U}[\![d]\!][\![d]\!] \leq 7)$.

Rule $\wedge$ yields

$$(0, \kappa), \ldots, (0, \kappa'), (0, \mathbf{B}(f \wedge h) = 0.35), (0, \mathbf{B}(f \wedge \neg h) = 0.35),$$
$$(0, \mathbf{B}(\neg f \wedge h) = 0.2), (0, \mathbf{B}(\neg f \wedge \neg h) = 0.1), (0, \neg[\![g + oN]\!]\mathbf{U}[\![d]\!][\![d]\!] \leq 7) \in \Gamma_1^0,$$

where $\kappa, \ldots, \kappa' \in BK$. Rule $\neg\Xi$ yields

$$(0, \neg[\![g + oN]\!] \vee [\![g + oN]\!]\neg\mathbf{U}[\![d]\!][\![d]\!] \leq 7) \in \Gamma_2^0.$$

Then rule $\vee$ yields

$$(0, \neg[\![g + oN]\!]) \in \Gamma_0^1) \text{ and } (0, [\![g + oN]\!]\neg\mathbf{U}[\![d]\!][\![d]\!] \leq 7) \in \Gamma_3^0.$$

The subtree rooted at Γ_0^1 closes, just like in the first example.

Applying rule Ξ to $(0, [\![g + oN]\!]\neg\mathbf{U}[\![d]\!][\![d]\!] \leq 7)$ yields

$$(0 \xrightarrow{g,oN} 1, \neg\mathbf{U}[\![d]\!][\![d]\!] \leq 7) \in \Gamma_3^0$$

and rule $\neg\mathbf{U}$ yields

$$(0 \xrightarrow{g,oN} 1, \mathbf{U}[\![d]\!][\![d]\!] > 7) \in \Gamma_4^0.$$

The following equations (amongst others) are in $SI(\Gamma'')$.

$$\begin{aligned}
pr_{1,3}^{d} &= 0.95 \\
pr_{3,3}^{d} &= 0.95 \\
pr_{1,4}^{d} &= 0.05 \\
pr_{3,4}^{d} &= 0.05 \\
pr_{2,1}^{d} &= 0 \\
pr_{2,2}^{d} &= 0 \\
pr_{2,3}^{d} &= 0 \\
pr_{2,4}^{d} &= 0 \\
pr_{4,1}^{d} &= 0 \\
pr_{4,2}^{d} &= 0 \\
pr_{4,3}^{d} &= 0 \\
pr_{4,4}^{d} &= 0
\end{aligned}$$

$$\begin{aligned}
pr_{1}^{oN|d} &= 1 \\
pr_{2}^{oN|d} &= 1 \\
pr_{3}^{oN|d} &= 1 \\
pr_{4}^{oN|d} &= 1
\end{aligned}$$

$$\begin{aligned}
\omega_1^0 &= 0.35 \\
\omega_2^0 &= 0.35 \\
\omega_3^0 &= 0.2 \\
\omega_4^0 &= 0.1 \\
\omega_1^0 + \omega_2^0 + \omega_3^0 + \omega_4^0 &= 1 \\
\omega_1^1 + \omega_2^1 + \omega_3^1 + \omega_4^1 &= 1
\end{aligned}$$

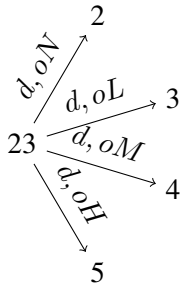


Fig. 7.3: The utility tree generated from $\{(1 \xrightarrow{d,oN} 2, [\![d]\!]), (1 \xrightarrow{d,oL} 3, [\![d]\!]), (1 \xrightarrow{d,oM} 4, [\![d]\!]), (1 \xrightarrow{d,oH} 5, [\![d]\!])\}$.

$$
\begin{aligned}
R_1 &= 0\\
R_2 &= 0\\
R_3 &= 10\\
R_4 &= -5
\end{aligned}
$$

$$
\begin{aligned}
C_1^d &= 1\\
C_2^d &= 1\\
C_3^d &= 1\\
C_4^d &= 1
\end{aligned}
$$

$$
\begin{aligned}
\omega_1^1 &= BT(0,1,g,N)\\
\omega_2^1 &= BT(0,2,g,N)\\
\omega_3^1 &= BT(0,3,g,N)\\
\omega_4^1 &= BT(0,4,g,N)
\end{aligned}
$$

According to the definition of untility trees (cf. § 7.2.2), $\Delta = \{(1 \xrightarrow{d,oN} 2, [\![d]\!]), (1 \xrightarrow{d,oL} 3, [\![d]\!]), (1 \xrightarrow{d,oM} 4, [\![d]\!]), (1 \xrightarrow{d,oH} 5, [\![d]\!])\}$. The only utility tree generated from Δ is shown in Figure 7.3. The following inequality generated for $(0 \xrightarrow{g,oN} 1, \mathbf{U}[\![d]\!][\![d]\!] > 7) \in \Gamma''$ using the utility tree is in $SI(\Gamma'')$.

$$
RC(d,1) \begin{array}{r} +\,\Pi(1,d,oN)RC(d,2)\\ +\,\Pi(1,d,oL)RC(d,3)\\ +\,\Pi(1,d,oM)RC(d,4)\\ +\,\Pi(1,d,oH)RC(d,5) \end{array} > 7. \tag{7.9}
$$

Also in $SI(\Gamma'')$ are

$$\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2 = 1$$

$$\omega_1^3 + \omega_2^3 + \omega_3^3 + \omega_4^3 = 1$$

$\omega_1^4 + \omega_2^4 + \omega_3^4 + \omega_4^4 = 1$

$\omega_1^5 + \omega_2^5 + \omega_3^5 + \omega_4^5 = 1$

$\Pi(1, d, oN) = 0 \;||\; \Pi(1, d, oN) \neq 0, \omega_1^2 = BT(1, 1, d, oN)$

$\Pi(1, d, oN) = 0 \;||\; \Pi(1, d, oN) \neq 0, \omega_2^2 = BT(1, 2, d, oN)$

$\Pi(1, d, oN) = 0 \;||\; \Pi(1, d, oN) \neq 0, \omega_3^2 = BT(1, 3, d, oN)$

$\Pi(1, d, oN) = 0 \;||\; \Pi(1, d, oN) \neq 0, \omega_4^2 = BT(1, 4, d, oN)$

$\Pi(1, d, oL) = 0 \;||\; \Pi(1, d, oL) \neq 0, \omega_1^3 = BT(1, 1, d, oL)$

$\Pi(1, d, oL) = 0 \;||\; \Pi(1, d, oL) \neq 0, \omega_2^3 = BT(1, 2, d, oL)$

$\Pi(1, d, oL) = 0 \;||\; \Pi(1, d, oL) \neq 0, \omega_3^3 = BT(1, 3, d, oL)$

$\Pi(1, d, oL) = 0 \;||\; \Pi(1, d, oL) \neq 0, \omega_4^3 = BT(1, 4, d, oL)$

$\Pi(1, d, oM) = 0 \;||\; \Pi(1, d, oM) \neq 0, \omega_1^4 = BT(1, 1, d, oM)$

$\Pi(1, d, oM) = 0 \;||\; \Pi(1, d, oM) \neq 0, \omega_2^4 = BT(1, 2, d, oM)$

$\Pi(1, d, oM) = 0 \;||\; \Pi(1, d, oM) \neq 0, \omega_3^4 = BT(1, 3, d, oM)$

$\Pi(1, d, oM) = 0 \;||\; \Pi(1, d, oM) \neq 0, \omega_4^4 = BT(1, 4, d, oM)$

$\Pi(1, d, oH) = 0 \;||\; \Pi(1, d, oH) \neq 0, \omega_1^5 = BT(1, 1, d, oH)$

$\Pi(1, d, oH) = 0 \;||\; \Pi(1, d, oH) \neq 0, \omega_2^5 = BT(1, 2, d, oH)$

$\Pi(1, d, oH) = 0 \;||\; \Pi(1, d, oH) \neq 0, \omega_3^5 = BT(1, 3, d, oH)$

$\Pi(1, d, oH) = 0 \;||\; \Pi(1, d, oH) \neq 0, \omega_4^5 = BT(1, 4, d, oH)$

Due to the law $\top \Rightarrow (oN \mid g) = 1 \;\wedge\; (oN \mid d) = 1 \in BK$, the law literal $\top \Rightarrow (oN \mid d) = 1$ is in Γ''. Recall that

$$pr_j^{oN|d} + pr_j^{oL|d} + pr_j^{oM|d} + pr_j^{oH|d} = \lceil pr_j^{oN|d} + pr_j^{oL|d} + pr_j^{oM|d} + pr_j^{oH|d} \rceil$$

is in $SI(\Gamma)$ for each j such that $w_j \models \top$ (in this case, $j = 1, 2, 3, 4$). And because $pr_j^{oN|d} = 1 \in SI(\Gamma)$, it follows that $pr_j^{oN|d} + pr_j^{L|d} + pr_j^{oM|d} + pr_j^{oH|d} = 1$. One can thus deduce that

$$pr_j^{oL|d} = pr_j^{oM|d} = pr_j^{oH|d} = 0. \tag{7.10}$$

Recall that $\Pi(e, \alpha, \varsigma, n) \stackrel{def}{=} \sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,j}^{\alpha} \omega_i^e$. Hence, by Equation 7.10, for $SI(\Gamma)$ to be feasible, each of $\Pi(1, d, oL, 4)RC(d, 3)$, $\Pi(1, d, oM, 4)RC(d, 4)$ and $\Pi(1, d, oH, 4)RC(d, 5)$ must be equal to zero. Therefore, by Inequality 7.9, $RC(d, 1) +$

$\Pi(1, d, N, 4)RC(d, 2)$ must be more than 7. That is,

$$\begin{aligned}&\omega_1^1(R_1 - C_1^d) + \omega_2^1(R_2 - C_2^d) + \omega_3^1(R_3 - C_3^d) + \omega_4^1(R_4 - C_4^d) + \\ &\sum_{j=1}^{n} pr_j^{oN|d} \sum_{i=1}^{n} pr_{i,j}^d \omega_i^1 \big(\omega_1^2(R_1 - C_1^d) + \\ &\quad \omega_2^2(R_2 - C_2^d) + \omega_3^2(R_3 - C_3^d) + \omega_4^2(R_4 - C_4^d)\big) > 7.\end{aligned} \tag{7.11}$$

We have calculated that for $SI(\Gamma'')$ to be feasible, it is required that $\omega_1^1 = 0.\overline{81}, \omega_2^1 = 0, \omega_3^1 = 0.\overline{09}, \omega_4^1 = 0.\overline{09}, \omega_1^2 = 0, \omega_2^2 = 0, \omega_3^2 = 0.95, \omega_4^2 = 0.05$ and $\sum_{j=1}^{n} pr_j^{N|d} \sum_{i=1}^{n} pr_{i,j}^d \omega_i^1 = 0.\overline{90}$. Therefore, (7.11) becomes $6.95455 > 7$, which is false. So $SI(\Gamma'')$ is infeasible, the tree closes and the entailment query holds.

Fifth Example

We query whether BK entails

$$\begin{aligned}&\mathbf{B}f = 0.7 \wedge \mathbf{B}(\neg f \wedge h) = 0.2 \wedge \mathbf{B}(\neg f \wedge \neg h) = 0.1 \\ &\rightarrow [\![g + oN]\!]\mathbf{U}[\![d]\!][\![d]\!] \leq 7.\end{aligned}$$

As in the second example, the initial belief-state is under-specified. The development of the decision proceeds almost exactly as in the previous example. The only difference is that where $\omega_1^0 = 0.35, \omega_2^0 = 0.35 \in SI(\Gamma'')$, now $\omega_1^0 + \omega_2^0 = 0.7 \in SI(\Gamma'')$. As in the previous example, $SI(\Gamma'')$ is feasible if and only if (7.11) is true. To evaluate (7.11), one needs the values of $\omega_1^1, \omega_2^1, \omega_3^1, \omega_4^1, \omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2$ and $\sum_{j=1}^{n} pr_j^{oN|d} \sum_{i=1}^{n} pr_{i,j}^d \omega_i^1$.

Setting the value of ω_1^0 to x and ω_2^0 to $0.7 - x$ as before, we have calculated that these values are constrained to be $\omega_1^1 = \frac{0.64-0.8x}{0.8-x}, \omega_2^1 = 0, \omega_3^1 = \frac{0.08-0.1x}{0.8-x}, \omega_4^1 = \frac{0.08-0.1x}{0.8-x}, \omega_1^2 = 0, \omega_2^2 = 0, \omega_3^2 = \frac{0.684-0.855x}{0.72-0.9x}, \omega_4^2 = \frac{0.036-0.045x}{0.72-0.9x}$, and $\sum_{j=1}^{n} pr_j^{oN|d} \sum_{i=1}^{n} pr_{i,j}^d \omega_i^1 = \frac{0.72-0.9x}{0.8-x}$. So (7.11) becomes

$$\begin{aligned}&\frac{0.64 - 0.8x}{0.8 - x}(R_1 - C_1^d) + \frac{0.08 - 0.1x}{0.8 - x}(R_3 - C_3^d) + \frac{0.08 - 0.1x}{0.8 - x}(R_4 - C_4^d) \\ &+\frac{0.72 - 0.9x}{0.8 - x}\Big(\frac{0.684 - 0.855x}{0.72 - 0.9x}(R_3 - C_3^d) + \frac{0.036 - 0.045x}{0.72 - 0.9x}(R_4 - C_4^d)\Big) > 7,\end{aligned}$$

which is equivalent to

$$\begin{aligned}&\frac{0.64 - 0.8x}{0.8 - x}(R_1 - C_1^d) + \frac{0.08 - 0.1x}{0.8 - x}(R_3 - C_3^d) + \frac{0.08 - 0.1x}{0.8 - x}(R_4 - C_4^d) \\ &+\frac{0.684 - 0.855x}{0.8 - x}(R_3 - C_3^d) + \frac{0.036 - 0.045x}{0.8 - x}(R_4 - C_4^d) > 7,\end{aligned}$$

which is equivalent to

$$\begin{aligned}&\frac{0.64 - 0.8x}{0.8 - x}(-1) + \frac{0.08 - 0.1x}{0.8 - x}(9) + \frac{0.08 - 0.1x}{0.8 - x}(-6) \\ &+\frac{0.684 - 0.855x}{0.8 - x}(9) + \frac{0.036 - 0.045x}{0.8 - x}(-6) > 7,\end{aligned}$$

which is equivalent to $x > 0.8$. But we know that $x \leq 0.7$.

So $SI(\Gamma'')$ is infeasible, the tree closes and the entailment query holds. This example shows that non-trivial entailments about the utility of sequences of actions can be confirmed, even without full knowledge about the initial belief-state.

7.6 Concluding Remarks

One advantage of having a logic for specifying POMDPs is that it can be done quite compactly. However, SDL is not unique in this regard. Although several frameworks have been proposed to deal with stochastic actions, noisy sensing, degree of belief and/or expected future rewards, not one of them can specify and reason about all these notions in a unified decidable logic. The Stochastic Decision Logic (SDL) can deal with all these notions to some degree. In particular, SDL is for specifying and reasoning about partially observable Markov decision processes (POMDPs). We discussed the model-theoretic correspondence between POMDPs and SDL and stated a theorem in this regard.

There is no notion of logical entailment in POMDP theory. Logical entailment can be applied to SDL sentences and hence to the POMDP models they represent. *A major contribution of this work is that it allows the user to determine whether or not a set of sentences is entailed by an arbitrarily precise specification of a POMDP model.* As far as we know, this is a novel property of SDL. Moreover, the procedure for deciding entailment is proved sound, complete and terminating. As a corollary, the entailment question for SDL is decidable.

SDL does not have a $\Box$ 'necessity operator'. Instead of $\Box$, we use $\Rightarrow$ in SDL. However, none of the necessity operators in our logics may be nested. If immutable propositions (cf. $\mathcal{LAP}$ in § 8.2) become indispensable, we may then add the $\Box/\Diamond$ operator appropriately. Or it might be possible to simulate the $\Box/\Diamond$ operator with quantification—at least to a sufficient degree. Quantifiers are defined in SDL.

Plans are not considered in LAO, SLAP or SLAOP, and only simple sequences of actions are considered in SDL. Automatic plan generation is highly desirable in cognitive robotics and for autonomous systems modeled as POMDPs. In future work, we would like to take SDL as the basis for developing a language or framework with which plans can be generated, in the fashion of DTGolog [Boutilier et al., 2000].

The reader may have noticed that the systems of inequalities are often nonlinear. The SI phases of SLAP and SLAOP involved only linear systems of inequalities, SDL's SI phase involves systems of nonlinear inequalities. Amato et al. [2007] solve POMDPs by setting up quadratically constrained linear programs (QCLPs) and then solving these. Their work is similar to ours in that they also appeal to a class of nonlinear systems (QCLPs). Their work is different to SDL's SI phase in that they seek to optimize planning in a non-logical setting.

The representation language proposed by Wang and Schmolze [2005] is well enough defined that it could be turned into a logical system. But what they seem to lack with respect to our work is the ability to make entailment queries, given an arbitrary (lack of) background information about the domain. Moreover, in their system, lack of probabilistic information is taken to mean uniform

distribution. In our system, a query might not be entailed by a uniform probability distribution for missing information, but might be entailed by non-uniform probability distribution. That is, SDL does not assume a uniform distribution when probabilistic information is lacking. Nevertheless, assuming uniform distributions when more is unknown may prove computationally faster and adequate for many domains.

We showed how an agent with belief-states and utilities, living in a stochastic domain and can be specified in the language of SDL and how POMDP models can be translated into a set of SDL sentences. Five examples were provided, illustrating the mechanics of the entailment decision procedure for SDL. Four of the five examples contain some form of incomplete specification. We showed that the decision procedure is robust enough to deal with these incomplete specifications and that useful inferences might be made, even when information is lacking. Reasoning about consequences with incomplete models in POMDP theory is typically not possible.

SDL is a decidable logic in which partially observable Markov decision processes (POMDPs) can be specified with compact representations, and queries can be posed about (i) the degree of belief in a propositional sentence after an arbitrary finite number of actions and observations and (ii) the utility of a finite sequence of actions after a number of actions and observations. The task of the logic is to check whether a query (stated in the language of the logic) follows from a knowledge base (KB), which is typically a POMDP model specification (also stated in the language of the logic). The main contribution of this work is that the POMDP model specification is allowed to be partial or incomplete with no restriction on the lack of information specified for the model. The model may even contain information about non-initial beliefs. Essentially, entailment of arbitrary queries (expressible in the language) can be answered. A sound, complete and terminating decision procedure was provided in the previous chapter.

8. RELATED WORK

We now discuss some related frameworks and logics in more depth, presented more or less chronologically.

8.1 ALX

Besides the fact that ALX [Huang et al., 1996] has elements of multi-modal propositional logic and that it is for reasoning about action, what makes it of interest to us is that it includes a notion of preference and a notion of update. ALX does not deal with sensing, though.

Another aspect of ALX that is of interest here, is that Huang et al. [1996] developed their logic to deal with incomplete information that an agent may have. Their "point of departure" is the conceptualization of Herbert A. Simon's *bounded rationality*: (i) an agent does not know all possible alternative actions in a given situation, that is, the agent has a limited, finite set of actions it can choose from, (ii) it does not know the exact outcome of each alternative, and (iii) the agent does not have a complete preference order on all situations. Besides these three limitations in knowledge, ALX also allows an agent to specify its current situation with partial information. That is, ALX follows the typical approach to specify an agent's current state when the agent has incomplete information ϕ about its current state: all states entailed by formula ϕ are states the agent believes it could be in. Huang et al. [1996] use Kripke's *possible worlds semantics* to formalize these notions.

In ALX, the set of states (possible worlds) the agent thinks it is currently in, is called a *situation*. Given two formulae ϕ and ψ, each representing situations (sets of states), ALX allows one to express the fact that situation ϕ is preferred to situation ψ. Huang et al. [1996] also provide a means for determining the situation where ϕ will be the case, given ψ is the case. That is, they provide for epistemic update.

ALXS, an inference system of logical axioms and inference rules for ALX, is given and proved sound, complete and decidable in the article [Huang et al., 1996].

Here, we want to show an example of a modal action logic that can express preference-driven practical reasoning. ALX does not involve stochastic notions or quantitative rewards, as does SDL.

8.2 The $\mathcal{LAP}$ Family

For the foundations of our logics we found inspiration from the Logic of Action and Plans $\mathcal{LAP}$ [Castilho et al., 1999]. Their tableau method was especially useful as a starting point for our decision procedures. However, $\mathcal{LAP}$ deals with uncertainty of action effects nondeterministically (i.e., only with disjunction); this is a coarse-grained approach to dealing with uncertainty.

$\mathcal{LAP}$ is a logic of actions and plans; it is a multi-modal logic, close to but simpler than propositional dynamic logic (PDL) [Harel et al., 2000]. Castilho et al. [1999] claim that $\mathcal{LAP}$ is sufficient to express most of the problems investigated in the field; it does not deal with sensing, however. They say the situation calculus is "too rich" for the purpose of reasoning about action and brings with it the undecidability of first-order logic. Moreover, they argue that *situations* are first-order objects in the situation calculus, but *situations* are not part of natural language. However, this criticism has been dealt with by the situation calculus based logic $\mathcal{ES}$ (see § 8.8).

$\Box$ is used to express laws. $\Box\Phi$ as a law says that Φ holds in any context or world. Such laws may be about phenomena that never vary, or about actions, for example, to specify when an action can be executed and what effects an action has. An action effect law that expresses 'If the gun is loaded, then after shooting the turkey, it is not alive' is $\Box(Loaded \rightarrow [shoot]\neg Alive)$.

Castilho et al. [1999] provide a sound and complete Hilbert style axiomatic system for $\mathcal{LAP}$. With the axiomatics, the following interesting and valid formula can be derived.

$$\Box\Phi \rightarrow [a_1][a_2]\ldots[a_n]\Phi,$$

for every $n \geq 1$. This clarifies the meaning of $\Box$ with respect to actions. $\Box$ can also be employed to capture, so-called, *immutable* propositions like $\Box(\neg Alive \rightarrow \Box\neg Alive)$. The diamond operator $\Diamond$ (dual of $\Box$) is also available in $\mathcal{LAP}$. $\Diamond\Phi$ is read, 'There exists a sequence of actions after which Φ will be true.' In this sense, the formula $\Diamond\Phi$ marks Φ as a goal, and the sequence of actions is a plan.

Although laws can be captured with global axioms, eliminating the need for $\Box$ to express laws, as in our logic LAO, it is not obvious how immutable propositions and goals for planning can be specified without $\Box$ and $\Diamond$. The SDL does have a 'necessity operator', however, none of the necessity operators in our logics may be nested.[1] If immutable propositions become indispensable, we may then add the $\Box/\Diamond$ operator appropriately. Or it might be possible to simulate the $\Box/\Diamond$ operator with quantification—at least to a sufficient degree. Quantifiers are defined in SDL. Plans are not considered in LAO, SLAP or SLAOP, and only simple sequences of actions are considered in SDL.

8.3 Modeling Action, Knowledge and Control

Geffner and Wainer [1998] present a "model" for specifying models of action and knowledge that is simpler than approaches before theirs, yet rigorous and meaningful enough, according to

[1] Instead of $\Box$, we use $\Rightarrow$ in SDL.

them. They mention that their work is an attempt to bridge the gap towards better planning and autonomous control.

"The language is a simplified first-order typed language that involves constant, function and predicate symbols but does not involve variables and quantification," [Geffner and Wainer, 1998]. One of the types is *action symbols*. Actions can be nondeterministic. It seems that their *random symbols* do the job of variables in classical first-order logic; the random symbols have denotations that can change. Other types are *fixed symbols* and *fluent symbols*.

Their approach includes a triplet $\langle D, A, C\rangle$ which is an *action theory*. D is a domain theory composed of a set of action rules (alias, effect axioms) and preconditions (the rules and conditions have the 'standard' forms, once one looks past the particular syntax). A is a set of *action occurrences* $p(t)[i]$, where $p(t)$ is an atom with p an action symbol and t a list of terms, and $i \in \{0, 1, \ldots\}$ is a time index. C is a typical set of *state formulae* specifying initial conditions.

Relating to our Stochastic Decision Logic (SDL), they define *knowledge* in terms of a set Bel of states: an expression x (symbol, term or formula) is 'known' in Bel, if for every state $s \in Bel$, the value of x in s is the same, whether true or false. Related to knowledge is observation; if the value of expression x is unobservable in some state s at time-index i, then x will be unknown in Bel' if $s \in Bel'$ at time-index i.

Geffner and Wainer [1998] provide a predicate $\mathbf{obs}(x)$ "with a special interpretation". They add to their action theory a set K of *observation rules* of the form

$$B \rightarrow \mathbf{obs}(x), \tag{8.1}$$

where B is an action formula. The formula in 8.1 can be read, 'x is observable if the conditions B are satisfied'. Expressions x in the set $O(s, a)$ are *observable* in a state s, given an action a:

$$O(s, a) \stackrel{def}{=} \{x \mid B \rightarrow \mathbf{obs}(x) \in K \text{ and } B^{s+a} = \mathbf{true}\},$$

where B^{s+a} is the denotation of B under the combined mappings of s and a.

"The *observations* o in the states s after doing a are the mappings that assign to each expression $x \in O(s, a)$ the denotation $x^o = x^s$," [Geffner and Wainer, 1998]. Observations are *mappings* in their approach. An analysis of the computational, logical and theoretical ramifications of this semantics is beyond the scope of this thesis.

Relative to our SLAOP and SDL, the most important deficiency of the framework discussed in this section ([Geffner and Wainer, 1998]) is that one cannot express and reason about stochastic actions and observations.

8.4 BHL's Approach

Bacchus et al. [1999] (BHL) supply a sound theory and specification for reasoning with noisy sensors and graded belief. They provide a way to 'carry along' the graded knowledge of sensor data—making it possible to change it and reason with it at any time in the future. Their whole

approach is not formulated as a logic: they use the situation calculus to specify their approach. But BEL—the agent's degree of belief in a proposition (see below)—fall outside the logical language.[2]

Intuitively, their aim is to represent an agent's uncertainty by having a notion of which configuration of situations are currently possible; the *possible-worlds* framework. Then further, each possible world is given a likelihood weight. With these notions in place, they show how an agent can have a belief (a probability) about any sentence in any defined situation. They show that the way beliefs are updated in their approach is equivalent to the standard Bayesian belief update formulae.

Bacchus et al. [1999] go beyond the MDP model by providing $\text{BEL}(\phi, s)$, the agent's (probabilistic) degree of belief in the formula ϕ in situation s. It is calculated as the ratio of the sum of those $p(s', s)$ where $p(s', s) \wedge \phi[s']$ is true to: the sum of *all* the $p(s', s)$, where $p(s', s)$ is the *relative weight* of the robot's belief that it is in s' and s. These relative weights get their initial values from the robot designer; in the initial situation S_0, all situations K-related to S_0 must be given a weight, where $K(s', s)$ is the accessibility relation used in BHL's interpretation of the possible-worlds framework—when the agent believes it is in s, it also believes it is in s'.

So BEL is defined in terms of $p(s', s)$. Therefore beliefs are updated whenever $p(s', s)$ is updated, which is whenever an action is performed and p's successor-state axiom is called. Hence, the agent's beliefs change whenever it moves or senses.

Furthermore, "A logical consequence [...] is that $\text{BEL}(\phi, s)$ is a probability distribution over the situations K-related to s," [Bacchus et al., 1999, p. 15].

Unfortunately, BEL is not defined as part of the syntax; the knowledge engineer can thus not write logical sentences involving BEL. And they do not address utilities of actions.

8.5 Imprecise Observations of Mobile Robots Specified by a Modal Logic

De Weerdt et al. [1999] present a modal logic to deal with imprecision in robot actions and sensors. The language of their logic contains formulae $pr(\phi) < \alpha$, where $\alpha \in [0, 1]$, and formulae $[a]\phi$, where a is an atomic action and ϕ is a propositional sentence. $pr(\phi) < \alpha$ has the meaning, 'the probability of ϕ is less than α' and $[a]\phi$ has the meaning 'ϕ is true after a was executed'. De Weerdt et al. [1999] also provide syntax for the compound actions $(a_1; a_2)$ and $(a_1 \cup a_2)$, with the usual PDL [Harel et al., 2000] interpretations. Finally, $\text{DONE}(a)$ and $\text{TODO}(a)$ are two special propositions with intended meanings 'a has just been done' and respectively, 'a will be done next'.

They present a model $\mathbb{M} = \langle S, \pi, Pr \rangle$ to represent the beliefs of the robot [De Weerdt et al., 1999], where S is a set of states, π is a truth assignment to the propositional atoms per state, and Pr assigns a probability to each state in S. They call the class of all models $\mathcal{M}$.

They state: "We view the execution of an action as a transformation of the whole model $\mathcal{M}$ to another model $\mathcal{M}'$ or even, in the case of non-deterministic action, to a set of models," [De Weerdt et al., 1999]. They define an 'update' function $E : \mathbf{A} \mapsto 2^{\mathcal{M}} \mapsto 2^{\mathcal{M}}$, where $\mathbf{A}$ is the set of compound actions. For instance, when the robot 'believes' the singleton model $\mathbb{M}$ and a is

[2] Another approach to reasoning with actions and degrees of belief in the situation calculus, is of Bacchus et al. [1994].

deterministic, then $E(a)(\mathbb{M}) = \mathbb{M}' = \langle S', \pi', Pr' \rangle$. Their semantics involves transitions between sets of models; transitions must be defined for each atomic action.

A clear description of nondeterminism seems to be missing from their paper. However, to define the behavior of a nondeterministic action a, De Weerdt et al. [1999] would presumably use *decomposition* by specifying a mapping from a_n to a set of possible outcomes / deterministic actions.

The authors discuss how to capture "imperfect observations" [De Weerdt et al., 1999] by specifying the behavior of an $\mathbf{obs}(d)$ action, where d would be some possible value returned by a sensor. There would then be a specification for the action $\mathbf{obs}(d)$ as a nondeterministic choice between all possible $\mathbf{obs}(d)$ in a set of observations D.

They show with some examples how the truth value of a sentence is determined, given the logic's semantics. However, they do not provide a proof system—axiomatic or otherwise—to prove statements in their language. Furthermore, no description is given of a systematic formalization of their intended domains. That is, they do not address domain specification. Also, their paper focuses on noisy sensing; no attention is given to nondeterminstic actions. All they say is,

> we can also specify the effect of a movement. A move action can, just as an observation, be seen as a non-deterministic choice between several move actions. These choices represent the uncertainty introduced by the move. [De Weerdt et al., 1999]

And they do not address utilities of actions.

8.6 Using Modal Logic in Mobile Robots

In his Master's dissertation, Van Diggelen [2002] presents a logic called $\mathcal{L}_{ProbDL}$ as the basis for a system to specify the behavior of mobile robots. "In $\mathcal{L}_{ProbDL}$ the performance of actions with non-deterministic effects [...] is assumed to be the only cause of uncertainty," [Van Diggelen, 2002, p. 33]. An agent is assumed to always know in what world it is and observation is always certain (complete).

Van Diggelen introduces $\mathcal{L}_{ProbDL}$ in essentially three steps: (1) as a simpler (non-probabilistic) version of propositional dynamic logic, (2) then extending it with probability theory, and (3) then adding epistemic notions.

In step (1), a structure $\mathcal{M} = \langle S, \pi, R \rangle$ is defined, where

- S is a (non-empty) set of possible states.
- $\pi : S \to (\mathcal{P} \to \{0, 1\})$ is a truth assignment function to the propositional atoms in $\mathcal{P}$ per state.
- $R : \mathcal{A} \to 2^{S \times S}$ is a relation containing the state transition relations per action in $\mathcal{A}$.

He defines the action-indexed *box* and *diamond* operators with the usual meanings. He also mentions the basic complex actions for PDL, but he modifies these when adding probability, so we discuss them next.

In step (2), the accessibility relation is changed to $R : \mathcal{A} \to 2^{S \times S \times (0,1]}$. The new third component of the range of R is the probability with which a world will be accessed. Note that if there is a

relationship between two worlds, the probability that the successor world will be accessed is never 0. Furthermore, the relation R is restricted to being a tree structure (with a single root). No reason for this restriction is given. Lastly, the usual constraint is made that the sum of effect probabilities must sum to 1.

The dynamic box modality is defined as follows.

$$\mathcal{M}, s \models [\alpha]^{\Delta}_{pr}\phi \text{ iff } \Big(\sum_{\{pr' | (s,t,pr') \in R(a), \mathcal{M}, t \models \phi\}} pr' \Big) \; \Delta \; pr,$$

where $\Delta = \{<, \leq, >, \geq, =\}$.

Action sequence $(a_1; a_2)$ and the *if-then-else* construct IF are defined, both dealing with probabilistic implications. As an example, $[a]^{\geq}_{0.2}\phi$ says that 'execution of a results in a state where ϕ is true in at least 20% of the cases'.

"Sensing actions are a special kind of action which take as a general form $S(\phi)$, where ϕ is a formula stated in propositional logic," [Van Diggelen, 2002, p. 33]. Sensing actions are allowed to have (side) effects, but they must be deterministic (which is intuitively reasonable).

Van Diggelen introduces a probabilistic diamond operator separately, with an unconventional (and in our view, unintuitive) semantic definition. He introduces "unreal" states that are reached (and are inescapable) when an agent performs an impossible action. Besides the strangeness of unreal states, a formula like for instance $\langle a \rangle \phi^{=}_{0.6}$ is philosophically problematic. Given that $[a]\phi^{=}_{0.6}$ means 'ϕ (necessarily) holds with probability 60%', $\langle a \rangle \phi^{=}_{0.6}$ should read 'ϕ possibly holds with probability 60%'. When a person says the latter sentence, it usually just means 'ϕ holds with probability 60%'. In normal natural language, 'ϕ possibly holds with probability 60%' does not have a clearly different meaning to 'ϕ must hold with probability 60%'. On page 42 of his dissertation, he writes, "[The agent] has a chance of 0.578 of remaining not shot after performing the $RusRoul$ action three times [...]" Van Diggelen [2002]. We would capture this formally as

$$[RusRoul; RusRoul; RusRoul]^{=}_{0.578} \neg shot,$$

but Van Diggelen captures it as

$$\langle RusRoul; RusRoul; RusRoul \rangle^{=}_{0.578} \neg shot.$$

We argue that the notion of 'probability of possibility' is unnatural for humans to reason with, and the knowledge engineer will thus simply never make use of the probabilistic diamond operator. This is especially due to the non-probabilistic (dynamic) diamond operator being sufficient for a logic for robotics, as is evident in the semantics and pragmatics of most stochastic action logics.

In step (3), an S5 system of axioms is added to $\mathcal{L}_{ProbDL}$ to model uncertainty in action outcomes. Correspondingly, an equivalence relation K is added to the $\mathcal{L}_{ProbDL}$, and the associated epistemic K operator is added to the syntax.

Furthermore, Van Diggelen defines a probabilistic version of the K operator. To define it specifi-

cally for his logic, he shows how to determine the probability of any state s: $StatePr(s)$. Then

$$\mathcal{M}, s \models K^{\Delta}_{pr}\phi \text{ iff } \Big(\sum_{\{t|(s,t)\in K,\mathcal{M},t\models\phi\}} StatePr(t)\Big)\Delta pr.$$

8.7 The ICL

Poole started work on a logic based framework for decision-making in uncertain environments when he extended previous work on Probabilistic Horn Abduction in the 90s. The result is the Independent Choice Logic (ICL) [Poole, 1998], with acyclic logic programs. The ICL$_{SC}$ uses the situation calculus as representation language [Poole, 1998]. A few years ago, he wrote a paper about the current state of the logic [Poole, 2008]. The ICL is not actually a logic; it is a structure with some components referring to sets of first-order logical formulae of a restricted form and a probability distribution over them. Although the restrictions are relatively tight, some types of formulae may still allow variables and quantification, and function symbols.

"The representation in this paper can be seen as a representation for POMDPs," [Poole, 1998]. Belief-states can be expressed and belief update can be performed (but maintenance of belief-states is not a necessary component of the system). For certain applications, SLAOP or SDL may be preferred due to their comparative simplicity. And because their semantics are very close to that of standard POMDP theory (esp. SDL), they may be easier to understand by people familiar with POMDPs. Finally, decidability of inferences made in the ICL are, in general, not guaranteed.

Poole mentions that it is argued in the AI community that a logic for knowledge representation should be at least as expressive as *first-order* logic [Poole, 1998, § 1.3]. But it is an *argument* exactly because there are people working in AI who argue for much simpler logics; providing only as much expressivity as is required for a particular application area. It seems like the ICL is intended for more general application than SDL or planned extensions to SDL is; our starting point is thus a simpler logic for a narrower application area.

The ICL is quite mature and some of its features are not part of SDL. He describes how to model utility and noisy sensors [Poole, 1998, § 2.6 and § 2.7] and says that the ICL$_{SC}$ can represent POMDPs using stochastic situation calculus rules to specify the state transition function and the reward function [Poole, 1998, § 3.2].[3] The use of a set of observables is similar to SDL's set of observations. Poole shows how the ICL can (i) represent Belief Networks, (ii) be seen as the procedural interpretation of logic programs and (iii) be used as a reasoning framework for abduction and logical argumentation.

The base logic of the ICL is first-order, versus propositional modal logic for our work, and the ICL formalism is unconventional (but not radical). Nevertheless, the application areas of our systems are the same and both use probability theory and Bayesian decision theory (although SDL still has a narrower application area).

[3] To deal with sensing, the basic theory structure presented below needs to be extended with the *observables* $\mathcal{O}$, similar to the choice space $\mathcal{C}$ and an *observable function* that associates choices from $\mathcal{C}$ with observations in $\mathcal{O}$.

8.8 $\mathcal{ESP}$

$\mathcal{ESP}$ [Gabaldon and Lakemeyer, 2007] is founded on $\mathcal{ES}$ [Levesque and Lakemeyer, 2004], which is a fragment of the situation calculus ("$\mathcal{ESP} = \mathcal{ES}$ + uncertainty"). The way they deal with uncertainty is inspired by Bacchus et al. [1999]'s approach. Every notion of interest is expressible within the defined language; no second-order logic is needed, which is the case for some parts of Bacchus et al. [1999]'s approach. An agent's epistemic state includes sets of probability distributions over both the initial situations and the outcome of stochastic actions after any number of actions.

It is a 'situation' based logic, but does not include situation terms.

> In addition, this result allows us to automatically transfer results obtained for $\mathcal{ES}$ to that fragment of the situation calculus, which is expressive enough to formulate basic action theories, and more. [Lakemeyer and Levesque, 2010, p. 3]

$\mathcal{ESP}$ is a modal dialect with *object* and *action* sorts, and with universal quantification and equality. It has fluent and rigid functions and predicates. Fluent predicates include the special predicates $Poss$ for defining preconditions on action executability and SF for defining whether a sensing action was successful. Reiter-style successor-state axioms can be expressed in $\mathcal{ESP}$ and regression is defined (via the successor-state axioms).

It has a modal operator $Know$ which allows the logic to distinguish between true sentences, and sentences 'known' (believed) to be true, possibly mistakenly so. $Know$ operates over a subset e of possible worlds. $e \subseteq W$ is called the *epistemic state*; e is fixed, that is, it does not change with execution of actions. $\mathcal{ESP}$ goes another step farther by including the notion of *only knowing* that has some interesting and desirable properties [Levesque and Lakemeyer, 2004, Lakemeyer and Levesque, 2010]. A probabilistic version of *only-knowing* is defined for $\mathcal{ESP}$.

$\Box\varphi$ is defined and is read, 'φ is true after any sequence of actions'. This operator is mostly used to state laws that are true at all times. For example, precondition, sensing and successor-state axioms have the form $\Box Poss(a) \equiv \ldots$, $\Box SF(a) \equiv \ldots$ and $\Box[a]F \equiv \ldots$, respectively, where a is an action and F a fluent predicate. The use of $\Box$ in this manner is identical to its use in $\mathcal{LAP}$.

$[a]\varphi$ is defined to mean 'after action a, φ is true'. But the semantic definition is not the same as the modal logic operator, because 'primitive' actions are deterministic. The modal logic diamond operator is thus not defined, besides, with $Poss$ available, it needs not be defined. Noise in actions and observations is formalized as follows in $\mathcal{ESP}$. Predicate $Choice(a, n)$ is added; it is true just in case n is one of the choices of stochastic action a. In the semantics, λ is a probability distribution over the outcomes of n—more precisely: "$\lambda(b, z, a, n) = p$ associates with each choice n of stochastic action a after any number of actions z starting in $b \in \mathcal{B}$ a likelihood p," [Gabaldon and Lakemeyer, 2007], where $\mathcal{B}$ is a finite partitioning of an infinite set of possible worlds W.

Since every action in $\mathcal{ESP}$ is associated with an outcome, $[a]\alpha$ of $\mathcal{ES}$ becomes $[(a, n)]\alpha$, which is a deterministic action. "We also include a new kind of modal operator $[\![a]\!]\alpha$, where a is a stochastic action. $[\![a]\!]\alpha$ is intended to mean that α holds after doing a regardless of which choice actually occurs," [Gabaldon and Lakemeyer, 2007]. It thus has the usual modal logic reading of traditional

modal logic's $[a]\alpha$.

A belief modal operator $HasP(\alpha, p)$ is also provided, meaning: statement α has probability p (similar to $\text{BEL}(\phi, s)$ of Bacchus et al. [1999]). But, unlike $\text{BEL}(\phi, s)$, $HasP(\cdot)$ has a syntax and semantics defined in the language of the logic. $HasP(\cdot)$ makes reference to $\mathcal{B}$. And in the semantics, μ is a probability distribution over $\mathcal{B}$.

With $\mathcal{ESP}$, one can model the dynamics of agents in stochastic domains, including stochastic sensing. In a sense, $\mathcal{ESP}$ is what Bacchus et al. [1999]'s approach would have been if it was all defined within a single logical language and if it did not mention situations explicitly.

The semantics of SDL is arguably simpler than that of $\mathcal{ESP}$. And to our knowledge, no decidable version or fragment of the situation calculus has been presented which can model and reason about POMDPs.

8.9 PDEL

There has been a series of articles on probabilistic dynamic epistemic logic (PDEL) [Kooi, 2003, Sack, 2009, Van Benthem et al., 2009], which add probabilistic notions to dynamic epistemic logic (DEL) [Van Ditmarsch et al., 2007].

DEL concerns reasoning about information and how to update new information received. DEL does not include actions, however, one may view the reception of information as an implicit 'act'. Nevertheless, these logics concern the 'movement' of information, not physical movement, the latter being the focus of the other logics reviewed in this chapter and this thesis.

Although DEL is not of particular interest to us, PDEL is, due to its treatment of *events*, which can be construed as observations and due to it having a notion of probability.

Van Benthem et al. [2009] mention that they move from traditional epistemic logic which updates an agent's knowledge with *propositional* information, to a logic where knowledge is updated with *event* information—an event in their language is not a propositional formula. Traditional epistemic logic has formulae $[\varphi]\psi$ meaning 'after knowledge update due to propositional information φ, ψ is true', whereas their more general epistemic logic includes events and information about events, instead of φ.

The terms *event* and *observation* often have the same meaning in probability theory. Observations in probability theory do not describe a state, but bring in information about natural *occurrences*. The authors [Van Benthem et al., 2009] allude that their events are closer to observations than logical propositions.

Lastly, notice that there is no mention of *actions* in their logic, although it is a *dynamic* kind of logic. Their logic has a dynamical aspect due to the occurrence of (exogenous) events that change an agent's informational content. The dynamical aspect of the logics introduced in this thesis is attributable more to ontic events (viz., physical actions) and, to some degree, to observations too (except in the case of SLAP, which has no observations).

8.10 $\mathcal{E}+$

Iocchi et al. [2009] present a logic called $\mathcal{E}+$ for reasoning about agents with sensing, qualitative nondeterminism and probabilistic uncertainty in action outcomes. The application area is plan generation for agents with nondeterministic and probabilistic uncertainy. This is our application area of interest too. One difference of $\mathcal{E}+$ to the logics presented in this thesis, is that $\mathcal{E}+$ is based on a fragment of the autoepistemic description logic $\mathcal{ALCK}_{\mathcal{NF}}$ for modeling dynamic systems, whereas our logics are multi-modal. And $\mathcal{E}+$ considers observations as fluents, whereas our logics consider observations as explicit object separate from fluents.

The authors show how to find finite horizon conditional plans from an initial state for a goal description (with "maximal goodness"). They prove the planning algorithm sound, complete and computable. But they do not address the question of whether an arbitrary sentence in their language is entailed by some knowledge base. The SDL decision procedure can determine the truth of arbitrary entailment queries, including sequences of actions, but complex plans cannot be expressed in SDL. We now look at $\mathcal{E}+$ in more detail, but we leave out discussions of their work on planning.

$\mathcal{E}+$ has a finite set of actions and a finite set of fluents. Actions are divided into *physical* and *sensing* actions; they are not overtly identifiable, but are identified through their syntactic use.

They describe the forms of special axioms in their language, a finite set of which, is called an *action description* AD.

Their logic also requires an *initial state description* δ_I which is a fluent conjunction.

A *state* of an AD is a fluent conjunction that satisfies all the domain constraint axioms of AD. A set of states S satisfies a formula ϕ if and only if every state in S satisfies ϕ.

> An *epistemic state* (or *e-state*) S of AD is a nonempty set of states s of AD such that (i) S satisfies every domain constraint axiom in AD, and (ii) there exists a fluent conjunction ϕ such that S is the set of all states s of AD that satisfy ϕ. [Iocchi et al., 2009, p. 8]

An initial state description represents an epistemic state, which is a set of possible states of the world.

An *extended action description* EAD includes the following axioms: *Nondeterministic conditional effect axioms* of the form

$$\textbf{caused } \psi_1, \ldots, \psi_n \textbf{ after } \alpha \textbf{ when } \phi,$$

where $\psi_1, \ldots, \psi_n$ and ϕ are fluent conjunctions, and $n \geq 2$, and

$$\textbf{caused } \psi_1 : p_1, \ldots, \psi_n : p_n \textbf{ after } \alpha \textbf{ when } \phi$$

is the form of *probabilistic conditional effect axioms*, where $p_1, \ldots, p_n > 0$ and $p_1 + \ldots + p_n = 1$. "Informally, if the current state of the world satisfies ϕ, then the successor state after executing α satisfies ψ_i with the probability p_i, for all $i \in \{1, \ldots, n\}$."

In $\mathcal{E}+$, an agent maintains and reasons over e-states. Executability of actions is defined in an e-state, and transitions between e-states through sensing actions, and deterministic, nondeterministic and probabilistic physical actions are also defined, all with respect to an EAD.

We shall not give the details of the semantics here. The authors, however, say that the probabilistic transitions in $\mathcal{E}+$ are similar to those in MDPs and POMDPs. "However, they are between e-states and thus involve *sets of states* rather than *single states*," [Iocchi et al., 2009, p. 14]. Although POMDPs deal with noise in sensing, $\mathcal{E}+$ does not. Furthermore, MDPs and POMDPs represent and reason with rewards of actions, whereas $\mathcal{E}+$ does not consider rewards. Noisy sensing is not dealt with, that is, sensing actions are deterministic. They mention that although they would like to be able to represent action rewards and costs as in POMDPs, $\mathcal{E}+$ does not yet provide the facilities.

8.11 Concluding Remarks

Every logic discussed above allows nesting of action modalities or the representation of sequences of actions. Out of the four logics developed in this thesis, only two allow nesting: The Logic of Actions and Observations allows nesting of "activity" operators (action-observation pairs) and the Stochastic Decision Logic allow sequences of "belief update" operators for reasoning about sequences of actions. Nesting or sequencing is not necessary for the purposes for which the other two logics, SLAP and SLAOP, were designed.

None of the logics reviewed in this chapter is developed particularly for representing and reasoning about stochastic domains as POMDPs. It has been seen that the Stochastic Decision Logic (Chap. 7) is very close to POMDP theory in its semantics.

9. CONCLUSIONS

The main contribution of this work is the definition of the Stochastic Decision Logic (SDL) and a decision procedure for determining whether or not a sentence is a semantic consequence of some background knowledge about a stochastic domain. The SDL was designed in four steps. That is, three logics were defined, each presenting one or more significant features, with the SDL combining all these features into a unified logic with new significant features. The Logic of Actions and Observations (LAO) presented observations at the same semantic level as actions, the Specification Logic of Actions with Probability (SLAP) presented a new way to specify and reason about stochastic actions, and the Specification Logic of Actions and Observations with Probability (SLAOP) added to SLAP the ability to specify and reason about stochastic observations (the same kind as in LAO) and action utility.

Although the SDL is not, strictly speaking, an extension of SLAOP, it takes SLAOP as a foundation and then adds the ability to express and reason about (uncertain) belief-states and sequences of actions and observations, and the utility of sequences of actions. What differentiates the SDL from other logics for reasoning about action and change with uncertainty is that its semantics is very closely based on partially observable Markov decision processes (POMDPs). A theorem stating the correspondence between POMDPs and the SDL, in the model-theoretic sense, was proved.

There is no notion of logical entailment in POMDP theory. Logical entailment can be applied to SDL sentences and hence to the POMDP models they represent. A major contribution of this work is that it allows the user to determine whether or not a set of sentences is entailed by an arbitrarily precise specification of a POMDP model. This is a novel property of the SDL. Moreover, the procedure for deciding entailment is proved sound, complete and terminating. As a corollary, the entailment question for the SDL is decidable.

Only simple sequences of actions are considered in the SDL; full-scale planning can presently not be performed. Automatic plan generation is highly desirable in cognitive robotics and for autonomous systems modeled as POMDPs. In future work, we would like to take the SDL as the basis for developing a language or framework with which plans can be generated, in the fashion of DTGolog [Boutilier et al., 2000] and PODTGolog [Rens, 2010]. In other words, we think that the SDL is a strong basis with which to design a system for logic-based planning with uncertainty. The uncertainty could lie in the action effects, environment observability or the initial state.

A 'necessity operator' ($\Box$ or $\Rightarrow$ in SDL) in our logics may not be nested. If immutable propositions, as in $\mathcal{LAP}$ [Castilho et al., 1999] become indispensable, we may then add the $\Box/\Diamond$ operator appropriately. Or it might be possible to simulate the $\Box/\Diamond$ operator with quantification—at least to a sufficient degree.

Whether to assume uniform probability distributions where information is lacking is debatable.

A knowledge engineer might prefer a system to automatically instantiate some standard (non-uniform) distributions or even other defaults when information is missing. A preliminary investigation was made in this topic in Chapter 5. More research is required here.

Finally, the computational complexity of SDL decision procedure is required. The systems of equations which must be solved in SLAOP can be effectively translated into elementary algebra in Tarski's sense [Tarski, 1957]. The elementary algebra is in EXPONENTIAL SPACE or PARALLEL EXPONENTIAL TIME [Ben-Or et al., 1986]. Given that checking unsatisfiability/validity of propositional logic is in the class co-NP in the number of atoms [Ben-Ari, 2001], SDL will, at least, be in PARALLEL EXPONENTIAL TIME in the worst case. Two places where complexity might explode in SDL is (i) with the obs rule, because it involves the $\Box$ operator, actions, observations *and* disjunction, and (ii) with the need to check feasibility of systems of inequalities for *every* open leaf node of a saturated tree. We would also like to investigate where the decision procedure could be optimized.

APPENDIX

A. PROOFS FOR THEOREMS AND LEMMATA

A.1 LAO

Lemma 4.3.1: Let T be a finished tree. For every node $\Gamma = \langle \Delta, \Sigma \rangle$ in T: If there exists a structure $\mathcal{S}$ such that for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and for every $x \in \mathit{Labels}(\Delta)$, there exists a $w' \in W$ such that for all $(x, \Phi) \in \Gamma$, $\mathcal{S}, w' \models \Phi$, then the (sub)tree rooted at Γ is open.

Proof:
(by induction on the height of the node Γ_k)

Base case:

Height $h = 0; \Gamma_k$ is a leaf. If there exists a structure $\mathcal{S}$ such that for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and for every $x \in \mathit{Labels}(\Delta)$, there exists a $w' \in W$ such that for all $(x, \Phi) \in \Gamma_k$, $\mathcal{S}, w \models \Phi$, then $(x', \bot) \notin \Gamma_k$ for all x'. Hence, the sub-tree consisting of Γ_k is open.

Induction step:

If $h > 0$, then some rule was applied to create the child(ren) $\Gamma_{k'} = \langle \Delta_{k'}, \Sigma_{k'} \rangle$ of $\Gamma_k = \langle \Delta_k, \Sigma_k \rangle$. We abbreviate "there exists a structure $\mathcal{S}^j = \langle W^j, R^j, O^j, N^j, Q^j \rangle$ such that for all $w \in W^j,\ \mathcal{S}^j, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and for every $x \in \mathit{Labels}(\Delta_k)$, there exists a $w' \in W^j$ such that for all $(x^j, \Phi^j) \in \Delta_j$ $\mathcal{S}^j, w' \models \Phi^j$" as $A(j)$ and we abbreviate "the (sub)tree rooted at Γ_j is open" as $B(j)$.

We must show the following for every rule. IF: If $A(k')$, then $B(k')$, THEN: If $A(k)$, then $B(k)$, where $\Gamma_{k'}$ is a child of Γ_k created due to the application of the rule. We assume the antecedent (induction hypothesis): If $A(k')$, then $B(k')$. To show the consequent, we must assume $A(k)$ and show that $B(k)$ follows.

Note that if the (sub)tree rooted at $\Gamma_{k'}$ is open, then the (sub)tree rooted at Γ_k is open. That is, if $B(k')$ then $B(k)$. So we want to show $B(k')$. But, by the induction hypothesis, $B(k')$ follows from $A(k')$. Therefore, it will suffice, in each case below, to assume $A(k)$, and prove $A(k')$.

- rule $\bot$:
  For the rule to have been applied, $(x, \Phi), (x, \neg\Phi) \in \Delta_k$, and after its application, $\Delta_{k'} = \Delta_k \cup \{(x, \bot)\}$. But there exists no structure $\mathcal{S}^k = \langle W^k, R^k \rangle$ such that there exists a $w' \in W^k$ such that $\mathcal{S}^k, w' \models \Phi$ and $\mathcal{S}^k, w' \models \neg\Phi$. Hence, assumption $A(k)$ is false; this rule could not have been applied.

- rule $\neg$:
  For the rule to have been applied, $(x, \neg\neg\Phi) \in \Delta_k$, and after its application, $\Delta_{k'} = \Delta_k \cup \{(x, \Phi)\}$. By assumption, $\mathcal{S}^k, w' \models \neg\neg\Phi$ for some $w' \in W^k$. Hence, $\mathcal{S}^k, w^k \models \Phi$. Thus, $A(k')$.

- rule $\wedge$:
  For the rule to have been applied, $(x, \Phi \wedge \Phi') \in \Delta_k$, and after its application, $\Delta_{k'} = \Delta_k \cup \{(x, \Phi), (x, \Phi')\}$. By assumption, $\mathcal{S}^k, w' \models \Phi \wedge \Phi'$ for some $w' \in W^k$. Hence, $\mathcal{S}^k, w' \models \Phi$ and $\mathcal{S}^k, w' \models \Phi'$. Thus, $A(k')$.

- rule $\vee$:
  For the rule to have been applied, $(x, \Phi \vee \Phi') \in \Delta_k$, and after its application, either $\Delta_{k'} = \Delta_k \cup \{(x, \Phi)\}$ or $\Delta_{k''} = \Delta_k \cup \{(x, \Phi')\}$. By assumption, $\mathcal{S}^k, w' \models \Phi \vee \Phi'$ for some $w' \in W^k$. Hence, $\mathcal{S}^k, w' \models \Phi$ or $\mathcal{S}^k, w' \models \Phi'$. Thus, $A(k')$ or $A(k'')$. Thus, $B(k')$ or $B(k'')$. Therefore, $B(k)$.

- rule $=$:
  For the rule to have been applied, $(x, c = c') \in \Delta_k$ or $(x, c \neq c') \in \Delta_k$. The rule is only applied when $(c = c')$, resp., $c \neq c'$ is unsatisfiable. Therefore, it is not the case that $A(k)$. But this contradicts our main assumption $A(k)$. Hence, rule $=$ could not have been applicable to Γ_k.

- rule $\Box$: For the rule to have been applied, $(x, [\varsigma \mid \alpha]\Phi) \in \Delta_k$ and $x \xrightarrow{\alpha} x', \varsigma \xrightarrow{\alpha} x' \in \Sigma_k$ and after its application, $\Delta_{k'} = \Delta_k \cup \{(x', \Phi)\}$. By assumption, $\mathcal{S}^k, w' \models [\varsigma \mid \alpha]\Phi$ for some $w' \in W^k$. That is,

$$\text{for all } w'' \in W^k, \text{ if } (w', w'') \in R_\alpha \text{ and } (\varsigma, w'') \in Q_\alpha, \text{ then } \mathcal{S}, w'' \models \Phi. \tag{A.1}$$

  But $x \xrightarrow{\alpha} x', \varsigma \xrightarrow{\alpha} x' \in \Sigma_k$ only if rule $\Diamond$ was applied, that is, only if $(x, \langle \varsigma \mid \alpha \rangle \Phi'') \in \Delta_k$. By assumption, $\mathcal{S}^k, w' \models \langle \varsigma \mid \alpha \rangle \Phi''$. That is, there exists a $w'' \in W^k$ such that $(w', w'') \in R_\alpha$, $(\varsigma, w'') \in Q_\alpha$ and $\mathcal{S}, w'' \models \Phi'$. Hence, by (A.1), $\mathcal{S}, w'' \models \Phi$. Thus, $A(k')$.

- rule $\Diamond$: For the rule to have been applied, $(x, \neg[\varsigma \mid \alpha]\Phi) \in \Delta_k$.

  If also $(x', \neg\Phi) \in \Delta_k$ for some label x', then after application of the rule, $\Delta_{k'} = \Delta_k$. By assumption, $\mathcal{S}^k, w' \models \neg[\varsigma \mid \alpha]\Phi$ for some $w' \in W^k$, that is, there exists a $w'' \in W^k$ such that $(w', w'') \in R_\alpha$, $(\varsigma, w'') \in Q_\alpha$ and $\mathcal{S}, w'' \models \neg\Phi$. Hence, there exists a $w'' \in W^k$ such that $\mathcal{S}, w'' \models \neg\Phi$. Thus, $A(k')$ holds trivially.

  Else (if $(x', \neg\Phi) \notin \Delta_k$ for some label x'), then after application of the rule, $\Delta_{k'} = \Delta_k \cup \{(x', \neg\Phi), (x', \bigwedge_{\kappa \in \mathcal{K}} \kappa)\}$, where x' is a fresh integer. By assumption, for all $w \in W^k$, $\mathcal{S}^k, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$. Also by assumption, $\mathcal{S}^k, w' \models \neg[\varsigma \mid \alpha]\Phi$ for some $w' \in W^k$, that is, there exists a $w'' \in W^k$ such that $(w', w'') \in R_\alpha$, $(\varsigma, w'') \in Q_\alpha$ and $\mathcal{S}, w'' \models \neg\Phi$. Hence, for all $w \in W^k$, $\mathcal{S}^k, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and there exists a $w'' \in W^k$ such that $\mathcal{S}, w'' \models \neg\Phi$. Thus, $A(k')$.

■

Theorem 4.3.1: (Soundness) If $\mathcal{K} \vdash_{LAO} \Phi$ then $\mathcal{K} \models_G \Phi$.

Proof:
$\mathcal{K} \models_G \Phi$
$\iff$ for all $\mathcal{S}$, if $\mathcal{S} \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$, then $\mathcal{S} \models \Phi$
$\iff$ for all $\mathcal{S}$, if for all $w \in W$, $\mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$, then for all $w' \in W$, $\mathcal{S}, w' \models \Phi$
$\iff$ not exists $\mathcal{S}$: not [if for all $w \in W$, $\mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$, then for all $w' \in W$, $\mathcal{S}, w' \models \Phi$]

$\iff$ not exists $\mathcal{S}$: [for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and not for all $w' \in W,\ \mathcal{S}, w' \models \Phi$]
$\iff$ not exists $\mathcal{S}$: [for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and exists $w' \in W,\ \mathcal{S}, w' \not\models \Phi$].

Let $\psi = \neg\Phi$. Then soundness can also be stated as, 'If the tree for $\bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \psi$ closes, then there does not exist an $\mathcal{S}$ such that for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and there exists a $w' \in W$ such that $\mathcal{S}, w' \models \psi$.' Contrapositively, 'If there exists an $\mathcal{S}$ such that for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and there exists a $w' \in W$ such that $\mathcal{S}, w' \models \psi$, then the tree for $\bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \psi$ is open.' Lemma 4.3.1 proves this.

Next, we show that steps 3 and 5 of the decision procedure are sound.

Step 3 says, 'For each open leaf node $\Gamma = \langle \Delta, \Sigma \rangle$: for every label $x \in \mathit{Labels}(\Delta)$, if w_x and $w_{x'}$ are fully specified and $w_x = w_{x'}$ where $x \neq x'$, then replace all x and x' in Δ by the same fresh integer x''.'

The labels of labeled formulae keep track of which sentences should hold in the same world. That is, if $(x, \Phi), (x, \Phi') \in \Delta$, then Φ and Φ' must both be satisfied at some world w associated with label x. Therefore, if it is known that two labels $x, x' \in \mathit{Labels}(\Delta)$, where $x \neq x'$, represent the same world w, then all the formulae associated with w due to being labeled by x or x', should be given the same label, say, x''. This is a sound procedure.

Step 5 says, for each open leaf node $\Gamma = \langle \Delta, \Sigma \rangle$: (i) If for every label $x \in \mathit{Labels}(\Delta)$, w_x is fully specified, then the tree is open, stop. Else continue. (ii) For every label $x \in \mathit{Labels}(\Delta)$, if neither $(x, f) \in \Delta$ nor $(x, \neg f) \in \Delta$ for some $f \in \mathcal{F}$, then create children Γ' and Γ'' of Γ, where $\Gamma' = \langle \Delta \cup \{(x, f)\}, \Sigma \rangle$ and $\Gamma'' = \langle \Delta \cup \{(x, \neg f)\}, \Sigma \rangle$.

(i) When this point in the procedure is reached, the tree is open. Hence, to state that it is open is sound. (ii) The effect of this point is equivalent to assuming $(x, f \vee \neg f) \in \Delta$ and then applying rule $\vee$ to it. Assuming that $(x, f \vee \neg f) \in \Delta$ is always sound. The reason for not including this point as a tableau rule is because it would cause the tableau method to become much more computationally expensive (even though optimization of the decision procedure is not our aim).

The following corollary is a consequence of the special case when $k = 0$ of Γ_k mentioned in the proof of Lemma 4.3.1. For every finished tree of a sentence Φ, if there exists a structure $\mathcal{S}$ and a $w \in W$ of $\mathcal{S}$ such that $\mathcal{S}, w \models \Phi$, then the tree is open. ■

Lemma 4.3.3: Let Γ be an open leaf node of a finished tree. For all $x \in \mathit{Labels}(\Delta)$, if $(x, \Phi) \in \Delta$, then $\mathcal{S}, w_x \models \Phi$.

Proof:
(by induction on the structure of Φ)

Let Θ be a subformula of Φ.

Base case:

- Θ is a literal. By construction, $w_x \models \Theta$. Hence $\mathcal{S}, w_x \models \Phi$.
- Θ is $c = c'$. Because $(x, \bot) \notin \Delta$, rule $=$ was not applied. Hence, c is identical to c', and $\mathcal{S}, w_x \models c = c'$.

- Θ is $\neg(c = c')$. Because $(x, \bot) \notin \Delta$, rule $=$ was not applied. Hence, c is not identical to c', and $\mathcal{S}, w_x \models \neg(c = c')$.

Induction step:

- Θ is $\neg\neg\Phi$. By rule $\neg$, $(x, \Phi) \in \Delta$. By induction hypothesis, $\mathcal{S}, w_x \models \Phi$. By the definition of $\neg$, $\mathcal{S}, w_x \models \neg\neg\Phi$.
- Θ is $\Phi \wedge \Phi'$. By rule $\wedge$, $(x, \Phi), (x, \Phi') \in \Delta$. By induction hypothesis, $\mathcal{S}, w_x \models \Phi$ and $\mathcal{S}, w_x \models \Phi'$. By the definition of $\wedge$, $\mathcal{S}, w_x \models \Phi \wedge \Phi'$.
- Θ is $\neg(\Phi \wedge \Phi')$. By rule $\vee$, $(x, \neg\Phi) \in \Delta$ or $(x, \neg\Phi') \in \Delta$. By induction hypothesis, $\mathcal{S}, w_x \models \Phi$ or $\mathcal{S}, w_x \models \Phi'$. By the definition of $\vee$, $\mathcal{S}, w_x \models \Phi \vee \Phi'$.
- Θ is $[\varsigma \mid \alpha]\Phi$. By rule $\Box$, whenever $x \xrightarrow{\alpha} x', \varsigma \xrightarrow{\alpha} x' \in \Sigma^j_k$, then $(x', \Phi) \in \Delta$. By construction, $(w_x, w_{x'}) \in R_\alpha$ iff $x \xrightarrow{\alpha} x' \in \Sigma$, and $(\varsigma, w_x) \in Q_\alpha$ iff $\varsigma \xrightarrow{\alpha} x \in \Sigma$. And by induction hypothesis, $\mathcal{S}, w_{x'} \models \Phi$. Hence, for all w', if $(w_x, w') \in R_\alpha$ and $(\varsigma, w') \in Q_\alpha$, then $\mathcal{S}, w' \models \Phi$. Thus, by definition of $[\varsigma \mid \alpha]\Phi$, $\mathcal{S}, w_x \models [\varsigma \mid \alpha]\Phi$.
- Θ is $\neg[\varsigma \mid \alpha]\Phi$.

  If $(x', \neg\Phi) \in \Delta$ for some label x', then, by rule $\Diamond$, $x \xrightarrow{\alpha} x', \varsigma \xrightarrow{\alpha} x' \in \Sigma$. By construction, $(w_x, w_{x'}) \in R_\alpha$ iff $x \xrightarrow{\alpha} x' \in \Sigma$, and $(\varsigma, w_x) \in Q_\alpha$ iff $\varsigma \xrightarrow{\alpha} x \in \Sigma$. And by induction hypothesis, $\mathcal{S}, w_{x'} \models \neg\Phi$. Hence, $w_{x'}$ and ς exist such that $(w_x, w_{x'}) \in R_\alpha$, $(\varsigma, w_{x'}) \in Q_\alpha$ and $\mathcal{S}, w_{x'} \models \neg\Phi$. Thus, by definition of $[\varsigma \mid \alpha]\Phi$, $\mathcal{S}, w_x \models \neg[\varsigma \mid \alpha]\Phi$.

  If $(x', \neg\Phi) \notin \Delta$ for some label x', then, by rule $\Diamond$, $(x', \neg\Phi), (x', \bigwedge_{\kappa\in\mathcal{K}} \kappa) \in \Delta$ with $x \xrightarrow{\alpha} x', \varsigma \xrightarrow{\alpha} x' \in \Sigma$ where x' is a fresh integer. By construction, $(w_x, w_{x'}) \in R_\alpha$ iff $x \xrightarrow{\alpha} x' \in \Sigma$, and $(\varsigma, w_x) \in Q_\alpha$ iff $\varsigma \xrightarrow{\alpha} x \in \Sigma$. And by induction hypothesis, $\mathcal{S}, w_{x'} \models \neg\Phi$. Hence, $w_{x'}$ and ς exist such that $(w_x, w_{x'}) \in R_\alpha$, $(\varsigma, w_{x'}) \in Q_\alpha$ and $\mathcal{S}, w_{x'} \models \neg\Phi$. Thus, by definition of $[\varsigma \mid \alpha]\Phi$, $\mathcal{S}, w_x \models \neg[\varsigma \mid \alpha]\Phi$.

■

Theorem 4.3.2: (Completeness) If $\mathcal{K} \models_G \Phi$ then $\mathcal{K} \vdash_{LAO} \Phi$.

Proof:
We know that $\mathcal{K} \models_G \Phi$ if and only if there does not exists an $\mathcal{S}$ such that [for all $w \in W$, $\mathcal{S}, w \models \bigwedge_{\kappa\in\mathcal{K}} \kappa$ and exists $w' \in W$, $\mathcal{S}, w' \not\models \Phi$].

Let $\psi = \neg\Phi$. Then completeness can also be stated as, 'If there does not exist an $\mathcal{S}$ such that for all $w \in W$, $\mathcal{S}, w \models \bigwedge_{\kappa\in\mathcal{K}} \kappa$ and there exists a $w' \in W$ such that $\mathcal{S}, w' \models \psi$, then the tree for $\bigwedge_{\kappa\in\mathcal{K}} \kappa \wedge \psi$ closes.' Contrapositively, 'If the tree for $\bigwedge_{\kappa\in\mathcal{K}} \kappa \wedge \psi$ is open, then there exists an $\mathcal{S}$ such that for all $w \in W$, $\mathcal{S}, w \models \bigwedge_{\kappa\in\mathcal{K}} \kappa$ and there exists a $w' \in W$ such that $\mathcal{S}, w' \models \psi$.'

It thus suffices to construct from an open leaf node of a finished tree for $\bigwedge_{\kappa\in\mathcal{K}} \kappa \wedge \psi$, a LAO structure $\mathcal{S} = \langle W, R, O, N, Q\rangle$ such that for all $w \in W$, $\mathcal{S}, w \models \bigwedge_{\kappa\in\mathcal{K}} \kappa$ and there exists a $w' \in W$ such that $\mathcal{S}, w' \models \psi$.

Assume $\Gamma = \langle\Delta, \Sigma\rangle$ is an open leaf node of a finished tree for $\bigwedge_{\kappa\in\mathcal{K}} \kappa \wedge \psi$. Then due to rule $\wedge$, $(0, \bigwedge_{\kappa\in\mathcal{K}} \kappa), (0, \psi) \in \Delta$, and due to rule $\Diamond$, $(x, \bigwedge_{\kappa\in\mathcal{K}} \kappa) \in \Delta$ for all $x \in \mathit{Labels}(\Delta)$ (note that new labels can only be introduced through rule $\Diamond$). By construction, $W = \{w_x \mid$

for each label $x \in Labels(\Delta)\}$. By Lemmata 4.3.2 and 4.3.3, there exists a structure $\mathcal{S}$ such that for all $x \in Labels(\Delta)$, for all $(x, \Phi) \in \Delta$, there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$. Hence, for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$. Therefore, the following corollary can be stated. The theorem is a direct consequence of the following corollary. If there is a finished open tree for $\bigwedge_{\kappa \in \mathcal{K}} \kappa \wedge \psi$, then there exists a LAO structure $\mathcal{S}$ such that for all $w \in W,\ \mathcal{S}, w \models \bigwedge_{\kappa \in \mathcal{K}} \kappa$ and there exists a $w' \in W$ such that $\mathcal{S}, w' \models \psi$. ■

A.2 SLAP

Lemma 5.3.1: Let T be a finished tree. For every node Γ in T: If there exists a structure $\mathcal{S}$ such that for all $(x, \Phi) \in \Gamma$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$, then the (sub)tree rooted at Γ is open.

Proof:
(by induction on the height of the node Γ_k)

Base case:

Height $h = 0; \Gamma_k$ is a leaf. If there exists a structure $\mathcal{S}$ such that for all $(x, \Phi) \in \Gamma_k$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$, then $(x', \bot) \notin \Gamma_k$ for all x'. Hence, the sub-tree consisting of Γ_k is open.

Induction step:

If $h > 0$, then some rule was applied to create the child(ren) $\Gamma_{k'}$ of Γ_k. We abbreviate "there exists a structure $\mathcal{S}^j = \langle W^j, R^j \rangle$ such that for all $(x^j, \Phi^j) \in \Gamma_j$ there exists a $w^j \in W^j$ such that $\mathcal{S}^j, w^j \models \Phi^j$" as $A(j)$ and we abbreviate "the (sub)tree rooted at Γ_j is open" as $B(j)$.

We must show the following for every rule/phase. IF: If $A(k')$, then $B(k')$, THEN: If $A(k)$, then $B(k)$. We assume the antecedent (induction hypothesis): If $A(k')$, then $B(k')$. To show the consequent, we must assume $A(k)$ and show that $B(k)$ follows.

Note that if the (sub)tree rooted at $\Gamma_{k'}$ is open, then the (sub)tree rooted at Γ_k is open. That is, if $B(k')$ then $B(k)$. So we want to show $B(k')$. But, by the induction hypothesis, $B(k')$ follows from $A(k')$. Therefore, it will suffice, in each case below, to assume $A(k)$, and prove $A(k')$.

- rule $\bot$:
  For the rule to have been applied, $\{(x, \Psi), (x, \neg\Psi)\} \subseteq \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \bot)\}$. But there exists no structure $\mathcal{S}^k = \langle W^k, R^k \rangle$ such that there exists a $w^k \in W^k$ such that $\mathcal{S}^k, w^k \models \Psi$ and $\mathcal{S}^k, w^k \models \neg\Psi$. Hence, assumption $A(k)$ is false and this rule could not have been applied.

- rule $\neg$:
  For the rule to have been applied, $(x, \neg\neg\Psi) \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \Psi)\}$. By assumption, $\mathcal{S}^k, w^k \models \neg\neg\Psi$. Hence, $\mathcal{S}^k, w^k \models \Psi$. Thus, $A(k')$.

- rule $\wedge$:
  For the rule to have been applied, $(x, \Psi \wedge \Psi') \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup$

$\{(x, \Psi), (x, \Psi')\}$. By assumption, $\mathcal{S}^k, w^k \models \Psi \wedge \Psi'$. Hence, $\mathcal{S}^k, w^k \models \Psi$ and $\mathcal{S}^k, w^k \models \Psi'$. Thus, $A(k')$.

- rule $\vee$:
  For the rule to have been applied, $(x, \Psi \vee \Psi') \in \Gamma_k$, and after its application, either $\Gamma_{k'} = \Gamma_k \cup \{(x, \Psi)\}$ or $\Gamma_{k''} = \Gamma_k \cup \{(x, \Psi')\}$. By assumption, $\mathcal{S}^k, w^k \models \Psi \vee \Psi'$. Hence, $\mathcal{S}^k, w^k \models \Psi$ or $\mathcal{S}^k, w^k \models \Psi'$. Thus, $A(k')$ or $A(k'')$. Thus, $B(k')$ or $B(k'')$. Therefore, $B(k)$.

- rule $\Diamond\varphi$:
  For the rule to have been applied, $(0, \neg[\alpha]_0\varphi) \in \Gamma_k$ or $(0, [\alpha]_q\varphi) \in \Gamma_k$ for $q > 0$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \varphi)\}$ where x is a fresh integer.

  By assumption, there exists a $w \in W$ such that $\mathcal{S}, w \models \neg[\alpha]_0\varphi$ or $\mathcal{S}, w \models [\alpha]_q\varphi$. Then by definition of $\langle\alpha\rangle$, there exists a $w'' \in W$ such that $(w, w'', pr) \in R_\alpha$ for $pr > 0$ and $\mathcal{S}, w'' \models \varphi$. Hence, for all $(x, \Phi') \in \Gamma_{k'}$ there exists a $w' \in W$ such that $\mathcal{S}, w' \models \Phi'$. Thus, $A(k')$.

- rule $\Box$:
  For the rule to have been applied, $\{(0, \Box\Phi), (x, \Phi'')\} \subseteq \Gamma_k$ for some $x \geq 0$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \Phi)\}$.

  By assumption, there exist $w, w' \in W$ such that $\mathcal{S}, w \models \Box\Phi$ and $\mathcal{S}, w' \models \Phi''$. Then by definition of $\Box$, for all $w'' \in W, \mathcal{S}, w'' \models \Phi$. That is, there exists a $w'' \in W$ such that $\mathcal{S}, w'' \models \Phi$. Hence, for all $(x, \Phi') \in \Gamma_{k'}$ there exists a $w''' \in W$ such that $\mathcal{S}, w''' \models \Phi'$. Thus, $A(k')$.

- In the SLI phase, the 'check' is: If $Z(F(\Gamma_k, \alpha, x)) = \emptyset$ for some action $\alpha \in \mathcal{A}$ and some label $x \in X(\Gamma)$, then create new leaf node $\Gamma_{k'} = \Gamma_k \cup \{(x, \bot)\}$.

  Recall that $F(\Gamma, \alpha, x) \stackrel{def}{=} \{[\alpha]_q\varphi \mid (x, [\alpha]_q\varphi) \in \Gamma\} \cup \{\neg[\alpha]_q\varphi \mid (x, \neg[\alpha]_q\varphi) \in \Gamma\}$.

  By assumption, for all actions $\alpha \in \mathcal{A}$, all labels $x \in X(\Gamma)$ and all $\delta \in F(\Gamma_k, \alpha, x)$, there exists a $w \in W$ such that $\mathcal{S}, w \models \delta$. Let α be an arbitrary action and x an arbitrary label in $X(\Gamma)$, and let

  $$R(\alpha, x)^{\#} = \{[\alpha]_{q_1}\varphi_1, [\alpha]_{q_2}\varphi_2, \ldots, [\alpha]_{q_g}\varphi_g, \neg[\alpha]_{q_{g+1}}\varphi_{g+1}, \neg[\alpha]_{q_{g+2}}\varphi_{g+2}, \ldots, \neg[\alpha]_{q_{g+h}}\varphi_{g+h}\}$$

  be an ordered set of the dynamic literals in $F(\Gamma_k, \alpha, x)$. Thus, there exists an R_α such that for w fixed,

  $$\begin{aligned}
  \sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \varphi_1} pr = q_1 \quad &\text{and} \\
  \sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \varphi_2} pr = q_2 \quad &\text{and} \\
  \vdots \\
  \sum_{(w,w',pr)\in R_\alpha, \mathcal{S}, w' \models \varphi_{g+h}} pr \neq q_{g+h}
  \end{aligned}$$

  such that $\sum_{(w,w',pr)\in R_\alpha} pr = 1$ or $\sum_{(w,w',pr)\in R_\alpha} pr = 0$, for $pr \in \mathbb{Q}_{[0,1]}$. Let $s_j = pr_j$

for $(w, w_j, pr_j) \in R_\alpha$, where $w_j \in W(\Gamma)^\#$. Then $(s_1, s_2, \ldots, s_n)$ is a solution to the SLI generated from $R(\alpha, x)^\#$ (or $F(\Gamma, \alpha, x)$). Hence, $Z(F(\Gamma_k, \alpha, x)) \neq \emptyset$. Therefore, no child is created for Γ_k and trivially, for all $(x, \Phi') \in \Gamma_{k'}$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi'$. Thus, $A(k')$.

■

Theorem 5.3.1: (Soundness) If $\vdash \Psi$ then $\models \Psi$. (Contrapositively, if $\not\models \Psi$ then $\not\vdash \Psi$.)

Proof:
Let $\psi = \neg\Psi$. Then $\not\vdash \Psi$ if and only if the tree for ψ is open. And

$$\begin{aligned} \not\models \Psi &\iff \text{not } (\forall \mathcal{S})\ \mathcal{S} \models \Psi \\ &\iff \text{not } (\forall \mathcal{S}, w)\ \mathcal{S}, w \models \Psi \\ &\iff (\exists \mathcal{S}, w)\ \mathcal{S}, w \models \psi. \end{aligned}$$

For the soundness proof, it thus suffices to show that if there exists a structure $\mathcal{S}$ and w in it such that $\mathcal{S}, w \models \psi$, then the tree rooted at $\Gamma_0^0 = \{(0, \psi)\}$ is open.

The following corollary is a consequent of the special case when $k = 0$ of Γ_k mentioned in the proof of Lemma 5.3.1. For every finished tree of a sentence Ψ, if there exists a structure $\mathcal{S}$ and a $w \in W$ of $\mathcal{S}$ such that $\mathcal{S}, w \models \Psi$, then the tree is open.

It is known that the first-order theory of rational numbers (linear arithmetic; without multiplication) is decidable; the Fourier-Motzkin method [Motzkin, 1936] and Dines' paper [Dines, 1919], for example, are proofs of this, and Ferrante and Rackoff [1975]'s method is a more efficient (almost polynomial) variant. Any system of equations and disequalities as they appear in this work, can easily be stated as an applicable first-order theory (see [Kroening and Strichman, 2008], e.g., and the appendix). In other words, there is a reliable means of determining whether there exists at least one solution to an SLI. Therefore, given the corollary to Lemma 5.3.1 stated above, every execution of a rule or procedure in the decision procedure is sound. ■

Lemma 5.3.3: Let Γ be the leaf node of a finished tree, where $(0, \Box\Phi) \in \Gamma$, for some $\Box\Phi \in \mathcal{L}_{SLAP}$. For every label $x \in X(\Gamma)$, there exists a term $(\Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m)$ of Φ such that $(x, \Phi_1), (x, \Phi_2), \ldots, (x, \Phi_m) \in \Gamma$.

Proof:
Let $\Phi := t_1 \vee t_2 \vee \cdots \vee t_z$. Let the labels mentioned in Γ (i.e., $X(\Gamma)$) be $\{0, 1, 2, \ldots, x'\}$. Rule $\Box$ is applied to $(0, \Box\Phi)$ for every label $(0, 1, 2, \ldots, x')$. Hence, due to multiple applications of rule $\Box$, the following labeled formulae are in Γ: $(0, t_1 \vee t_2 \vee \cdots \vee t_z)$, $(1, t_1 \vee t_2 \vee \cdots \vee t_z)$, $(2, t_1 \vee t_2 \vee \cdots \vee t_z)$, $\ldots$ $(x', t_1 \vee t_2 \vee \cdots \vee t_z)$. And due to multiple applications of rule $\vee$, one of the following sets is a subset of Γ.

- $\{(0, t_1), (1, t_1), (2, t_1), \ldots, (x', t_1)\}$,
- $\{(0, t_1), (1, t_1), (2, t_1), \ldots, (x', t_2)\}$,

$\vdots$

- $\{(0, t_1), (1, t_1), (2, t_1), \ldots, (x', t_z)\}$,

- $\{(0, t_2), (1, t_1), (2, t_1), \ldots, (x', t_1)\}$,
- $\{(0, t_2), (1, t_1), (2, t_1), \ldots, (x', t_2)\}$,

$\vdots$

- $\{(0, t_2), (1, t_1), (2, t_1), \ldots, (x', t_z)\}$,

$\vdots$

- $\{(0, t_z), (1, t_z), (2, t_z), \ldots, (x', t_z)\}$.

Now choose any one of these sets τ. For every label $x \in \{0, 1, 2, \ldots, x'\}$, $(x, t_k) \in \tau \subset \Gamma$ for some term $t_k := \Phi_{k1} \wedge \Phi_{k2} \wedge \cdots \wedge \Phi_{km_k}$ of Φ. Therefore, due to successive applications of rule $\wedge$, (x, Φ_{k1}), (x, Φ_{k2}), ..., $(x, \Phi_{km_k}) \in \Gamma$, for every label $x \in X(\Gamma)$. ■

Lemma 5.3.4: If Γ is the leaf node of an open branch of a finished tree, then there exists a structure $\mathcal{S}$ such that for all $(x, \Psi) \in \Gamma$, $\mathcal{S}, w \models \Psi$ for some $w \in W(\Gamma, x)$.

Proof:
(by induction on the structure of a formula)

Let $\mathcal{S}$ be constructed as described above.

The induction step will work as follows. Let $\gamma' \subseteq \Gamma$ be added to Γ due to some rule applied to $\gamma \subseteq \Gamma$. Thus, we need to prove that IF for all $(x', \Psi') \in \gamma'$, $\mathcal{S}, w' \models \Psi'$ for some $w' \in W(\Gamma, x')$, THEN for all $(x, \Psi) \in \gamma$, $\mathcal{S}, w \models \Psi$ for some $w \in W(\Gamma, x)$.

We assume the antecedent (induction hypothesis).

Base case:

- Ψ is a propositional literal. Then $\mathcal{S}, w \models \Psi$ for some $w \in W(\Gamma, x)$, by definition of $W(\Gamma, x)$.
- Ψ is $[\alpha]_q\varphi$. In the construction of $\mathcal{S}$, a solution in $Z(F(\Gamma, \alpha, x))$ is utilized for some $w \in W(\Gamma, x)$. Then as a direct consequence of the construction of $\mathcal{S}$, $\mathcal{S}, w \models [\alpha]_q\varphi$ for some $w \in W(\Gamma, x)$.
- Ψ is $\neg[\alpha]_q\varphi$. In the construction of $\mathcal{S}$, a solution in $Z(F(\Gamma, \alpha, x))$ is utilized for some $w \in W(\Gamma, x)$. Then as a direct consequence of the construction of $\mathcal{S}$, $\mathcal{S}, w \models \neg[\alpha]_q\varphi$ for some $w \in W(\Gamma, x)$.

Induction step:

- Ψ is $\neg\neg\psi$. By rule $\neg$, $(x, \psi) \in \Gamma$. By induction hypothesis, $\mathcal{S}, w \models \psi$. By the definition of $\neg$, $\mathcal{S}, w \models \neg\neg\psi$.
- Ψ is $\psi \wedge \psi'$. By rule $\wedge$, $(x, \psi), (x, \psi') \in \Gamma$. By induction hypothesis, $\mathcal{S}, w \models \psi$ and $\mathcal{S}, w \models \psi'$. By the definition of $\wedge$, $\mathcal{S}, w \models \psi \wedge \psi'$.
- Ψ is $\neg(\psi \wedge \psi')$. By rule $\vee$, $(x, \neg\psi) \in \Gamma$ or $(x, \neg\psi') \in \Gamma$. By induction hypothesis, $\mathcal{S}, w \models \neg\psi$ or $\mathcal{S}, w \models \neg\psi'$. By the definition of $\vee$, $\mathcal{S}, w \models \neg(\psi \wedge \psi')$.

- Ψ is $\Box\Phi$. Let $X(\Gamma) = \{0, 1, 2, \ldots, x'\}$. Due to successive application of rule $\Box$, $(0, \Phi)$, $(1, \Phi)$, …, $(x', \Phi) \in \Gamma$. Then, by induction hypothesis, $\mathcal{S}, w_0 \models \Phi$ for some $w_0 \in W(\Gamma, 0)$ and $\mathcal{S}, w_1 \models \Phi$ for some $w_1 \in W(\Gamma, 1)$ and $\cdots$ and $\mathcal{S}, w_{x'} \models \Phi$ for some $w_{x'} \in W(\Gamma, x')$.

  We need to show that $\mathcal{S}, w \models \Box\Phi$ for some $w \in W(\Gamma, 0)$, that is, that for all $w' \in W(\Gamma)$, $\mathcal{S}, w' \models \Phi$. This will be the case if: For every $w' \in W(\Gamma)$, there exists a term $t := \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m$ of Φ such that $\mathcal{S}, w' \models t$. Note that for $w_i, w_j \in W(\Gamma)$, if $w_i \neq w_j$, it is sufficient that there exist terms t_i and t_j of Φ such that $\mathcal{S}, w_i \models t_i$ and $\mathcal{S}, w_j \models t_j$, even if $t_i \neq t_j$.

  By Lemma 5.3.3, for every label $x \in X(\Gamma)$, there exists a term $t := \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m$ of Φ such that (x, Φ_1), (x, Φ_2), …, $(x, \Phi_m) \in \Gamma$.

  Then we define the set

  $$L(t) := \{\Phi_i \mid \Phi_i \text{ is a propositional literal conjunct of } t\}$$

  and the set

  $$\Delta(t) := \{\Phi_i \mid \Phi_i \text{ is a dynamic literal conjunct of } t\}.$$

  Note that $t \equiv \bigwedge_{\ell \in L(t)} \ell \wedge \bigwedge_{\delta \in \Delta(t)} \delta$.

  Let $\ell \in L(t)$. Then by induction hypothesis, $\mathcal{S}, w'' \models \ell$ for some $w'' \in W(\Gamma, x)$. Note that if $\mathcal{S}, w'' \models \ell$ for some $w'' \in W(\Gamma)$, then $w'' \in W(\Gamma, x)$. And if $w'' \models \ell$ for some $w'' \in W(\Gamma, x)$, then $w^* \models \ell$ for all $w^* \in W(\Gamma, x)$. Thus,

  $$\mathcal{S}, w^* \models \bigwedge_{\ell \in L(t)} \ell \text{ (for all } w^* \in W(\Gamma, x)).$$

  Hence, by definition of $W(\Gamma)$,

  $$\mathcal{S}, w' \models \bigwedge_{\ell \in L(t)} \ell \text{ (for all } w' \in W(\Gamma)). \tag{A.2}$$

  Let $\delta \in \Delta(t)$. Then by induction hypothesis, $\mathcal{S}, w'' \models \delta$ for some $w'' \in W(\Gamma, x)$. Recall that we defined $X(\Gamma)^{\#}$ to be some sequence of labels $(x_1, x_2, \ldots, x_n)$ such that $w_1 \in W(\Gamma, x_1)$, $w_2 \in W(\Gamma, x_2)$, …, $w_n \in W(\Gamma, x_n)$, where $(w_1, w_2, \ldots, w_n) = W(\Gamma)^{\#}$. By construction and definition of $X(\Gamma)^{\#}$, there is a label $x_i \in X(\Gamma)^{\#}$ such that a solution in $Z(F(\Gamma, \alpha, x_i))\}$ is used in the construction of $\mathcal{S}$, for every $w_i \in W(\Gamma)^{\#}$. Let w_i be w'' and x_i be x.

  Therefore, by (A.2) and the above argument for dynamic literals,

  $$\mathcal{S}, w' \models t \quad (\text{for all } w' \in W(\Gamma)).$$

■

Theorem 5.3.2: (Completeness) If $\models \Psi$ then $\vdash \Psi$. (Contrapositively, if $\nvdash \Psi$ then $\not\models \Psi$.)

Proof:
Let $\psi = \neg\Psi$. Then $\nvdash \Psi$ means that there is an open branch of a finished tree for ψ. And

$$\begin{aligned} \not\models \Psi &\iff (\exists \mathcal{S})\ \mathcal{S} \not\models \Psi \\ &\iff (\exists \mathcal{S}, w)\ \mathcal{S}, w \not\models \Psi \\ &\iff (\exists \mathcal{S}, w)\ \mathcal{S}, w \models \psi. \end{aligned}$$

For the completeness proof, it thus suffices to construct for some open branch of a finished tree for $\psi \in \mathcal{L}_{SLAP}$, an SLAP structure $\mathcal{S} = \langle W, R\rangle$ in which there is a world $w \in W$ such that ψ is true in $\mathcal{S}$ at w.

The following corollary holds. By Lemmata 5.3.2 and 5.3.4, given the leaf node Γ of an open branch of a finished tree, there exists a structure $\mathcal{S}$ such that for all $(x, \Phi) \in \Gamma$, for all $w \in W(\Gamma, x)$, $\mathcal{S}, w \models \Phi$. But $(0, \psi) \in \Gamma$. Thus, if there is a finished open tableau for ψ, then ψ is satisfiable.

Besides the references mentioned at the end of Section 5.3.1, Käufl [1988] says that the Inf-Sup-Method developed by Bledsoe [1975] and refined by Shostak [1977] "is a complete decision procedure for systems of linear inequalities over the rational numbers." The theorem follows directly from the corollary and the fact that there exist complete methods for determining the feasibility of SLIs as they appear in this work. ■

More on the decidability of the SLI phase (for SLAP): Let

$$a_1x_1 + \cdots + a_nx_n \bowtie b$$

be a *constraint*, where $\bowtie \in \{=, \leq, <\}$, the a_i and b are rational constants and the x_i are rational variables. The conjunction of such constraints is a quantifier-free fragment of the theory of rational linear arithmetic. We call the fragment $T_{\mathbb{Q}}$.

In terms of first-order logic, given some α and label x, we define the formula $A(\alpha, x) \in T_{\mathbb{Q}}$ as

$$\begin{array}{rl} c_{1,1}pr_1 + c_{1,2}pr_2 + \cdots + c_{1,n}pr_n & = q_1 \quad \wedge \\ c_{2,1}pr_1 + c_{2,2}pr_2 + \cdots + c_{2,n}pr_n & = q_2 \quad \wedge \\ \vdots & \\ c_{h,1}pr_1 + c_{h,2}pr_2 + \cdots + c_{h,n}pr_n & = q_h \quad \wedge \\ c_{h+1,1}pr_1 + c_{h+1,2}pr_2 + \cdots + c_{h+1,n}pr_n & \neq q_{h+1} \quad \wedge \\ c_{h+2,1}pr_1 + c_{h+2,2}pr_2 + \cdots + c_{h+2,n}pr_n & \neq q_{h+2} \quad \wedge \\ \vdots & \\ c_{h+k,1}pr_1 + c_{h+k,2}pr_2 + \cdots + c_{h+k,n}pr_n & \neq q_{h+k} \quad \wedge \\ pr_1 + pr_2 + \cdots + pr_n & = q*, \end{array} \tag{A.3}$$

where each of the first $h + k$ conjuncts ((in)equalities) represents an element in $F(\Gamma, \alpha, x)$ and in the last conjunct, $q* = 0$ or $q* = 1$. Due to $q*$ having two possible values, $A(\alpha, x)$ actually represents two systems or elements of $T_{\mathbb{Q}}$. Let all the *equations* (excluding disequations) be re-

presented by the system $\mathbf{Cpr} = \mathbf{q}$ of linear equations, where $\mathbf{C}$ is an $(h+1) \times n$ matrix, $\mathbf{pr}$ is an n-dimensional vector and $\mathbf{q}$ is an $(h+1)$-dimensional vector. Formula $A(\alpha, x)$ (system (A.3)) can then be written as

$$\mathbf{Cpr} = \mathbf{q} \ \wedge \ \bigwedge_{i=1}^{k} \sum_{j=1}^{n} c_{i,j} pr_{i,j} \neq q_i. \tag{A.4}$$

Remark A.2.1: A disequation $a_1 x_1 + \cdots + a_n x_n \neq b$ is equisatisfiable with

$$(a_1 x_1 + \cdots + a_n x_n < b) \quad \vee \quad (-a_1 x_1 - \cdots - a_n x_n < -b). \tag{A.5}$$

But formula (A.5) is not in $T_{\mathbb{Q}}$, although each of the two disjuncts is. By remark A.2.1, system (A.4) is satisfiable if and only if either

$$\mathbf{Cpr} = \mathbf{q} \ \wedge \ \bigwedge_{i=2}^{k} \sum_{j=1}^{n} c_{i,j} pr_{i,j} \neq q_i \ \wedge \ \sum_{j=1}^{n} c_{1,j} pr_{1,j} < q_1$$

is satisfiable or if

$$\mathbf{Cpr} = \mathbf{q} \ \wedge \ \bigwedge_{i=2}^{k} \sum_{j=1}^{n} c_{i,j} pr_{i,j} \neq q_i \ \wedge \ \sum_{j=1}^{n} c_{1,j} pr_{1,j} > q_1$$

is satisfiable. Following this reasoning, $A(\alpha, x)$ can be transformed into 2^k disequation-free systems $B_1, B_2, \ldots, B_{2^k} \in T_{\mathbb{Q}}^{-}$ such that $A(\alpha, x)$ is satisfiable if and only if at least one of $B_1, B_2, \ldots, B_{2^k}$ is satisfiable. That is, $A(\alpha, x)$ is satisfiable if and only if **general simplex** returns "satisfiable" for at least one of $B_1, B_2, \ldots, B_{2^k}$.

In fact, formulae in $T_{\mathbb{Q}}$ are not yet in the correct form to be taken as input to **general simplex**. Kroening and Strichman [2008] show how any formula in $T_{\mathbb{Q}}$ can be transformed into the so-called 'general form' required by **general simplex**. We refer the reader to their book for details.

General simplex allows one to set a lower bound l_i and an upper bound u_i for each variable x_i, such that $l_i \leq x_i \leq u_i$. For our problem, we set, $l_i = 0$ and $u_i = 1$, for $i = 1, \ldots, n$ (where $x_i = pr_i$).

Let $B(\Gamma, \alpha, x) := \{B_1, B_2, \ldots, B_{2^k}\}$ be induced from $A(\alpha, x)$ for some node Γ. Then $B(\Gamma, \alpha, x)$ is feasible if and only if the SLI generated from $F(\Gamma, \alpha, x)$ (cf. p. 103) is feasible.

A.3 SLAOP

Lemma 6.2.1: Determining whether an SI (as defined in this thesis) is feasible, is decidable.

Proof:
Tarski [1957] defines the first-order logic theory of elementary (real number) algebra as having an infinite number of variables (representing elements of $\mathbb{R}$), algebraic constants 1, 0, -1, two algebraic operation signs $+$ (addition) and $\cdot$ (multiplication), two algebraic relation symbols $=$ (equals) and $>$ (greater than), (logical) sentential connectives $\sim$ (negation), $\wedge$ (conjunction), $\vee$

(disjunction), the existential quantifier $\exists$, and a set of axioms defining the theory. "If ξ is any variable, then $(\exists\xi)$ is called a *quantifier expression*.[1] The expression $(\exists\xi)$ is to be read "there exists a ξ such that ."

For every SI (as defined in §6.2.2), the question of whether it has a solution can be represented in the language of first-order elementary algebra as follows.

$$
\begin{aligned}
&(\exists pr_1^\alpha)(\exists pr_2^\alpha)\cdots(\exists pr_n^\alpha)(\exists pr_1^\varsigma)(\exists pr_2^\varsigma)\cdots(\exists pr_m^\varsigma)\\
&c_{1,1}\cdot pr_1^\alpha + c_{1,2}\cdot pr_2^\alpha + \cdots + c_{1,n}\cdot pr_n^\alpha = q_1^\alpha \quad \wedge\\
&c_{2,1}\cdot pr_1^\alpha + c_{2,2}\cdot pr_2^\alpha + \cdots + c_{2,n}\cdot pr_n^\alpha = q_2^\alpha \quad \wedge\\
&\vdots\\
&c_{g,1}\cdot pr_1^\alpha + c_{g,2}\cdot pr_2^\alpha + \cdots + c_{g,n}\cdot pr_n^\alpha = q_g^\alpha \quad \wedge\\
&\sim (c_{g+1,1}\cdot pr_1^\alpha + c_{g+1,2}\cdot pr_2^\alpha + \cdots + c_{g+1,n}\cdot pr_n^\alpha = q_{g+1}^\alpha) \quad \wedge\\
&\sim (c_{g+2,1}\cdot pr_1^\alpha + c_{g+2,2}\cdot pr_2^\alpha + \cdots + c_{g+2,n}\cdot pr_n^\alpha = q_{g+2}^\alpha) \quad \wedge\\
&\vdots\\
&\sim (c_{g+h,1}\cdot pr_1^\alpha + c_{g+h,2}\cdot pr_2^\alpha + \cdots + c_{g+h,n}\cdot pr_n^\alpha = q_{g+h}^\alpha) \quad \wedge\\
&(pr_1^\alpha + pr_2^\alpha + \cdots + pr_n^\alpha = 1 \vee pr_1^\alpha + pr_2^\alpha + \cdots + pr_n^\alpha = 0) \quad \wedge\\
&pr_1^\sigma = q_1^\varsigma \quad \wedge\\
&pr_2^\sigma = q_2^\varsigma \quad \wedge\\
&\vdots\\
&pr_t^\sigma = q_t^\varsigma \quad \wedge\\
&\sim (pr_{t+1}^\sigma = q_{t+1}^\varsigma) \quad \wedge\\
&\sim (pr_{t+2}^\sigma = q_{t+2}^\varsigma) \quad \wedge\\
&\vdots\\
&\sim (pr_{t+v}^\sigma = q_{t+v}^\varsigma) \quad \wedge\\
&(pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_m^\varsigma = 1 \vee pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_m^\varsigma = 0)
\end{aligned}
$$

such that (algebraic constant) $c_{i,k} = 1$ or 0 as described in §6.2.2, the bracketed subformulae are present only if the corresponding (dis)equations are present in (6.2), and the pr_k^α, the pr_j^σ and the pr_j^ς are the variables.

Tarski [1957] provided a finite method which can always decide whether a sentence in the elementary algebra is in the theory. Hence, feasibility of SIs is decidable. ■

Lemma 6.3.1: Let Γ be the leaf node of a saturated tree. Suppose there exists a structure $\mathcal{S} = \langle W, R, O, N, Q, U\rangle$ such that $W = W(\Gamma)$, and for all $(x, \delta\omega) \in \Gamma$, where $\delta\omega$ is a dynamic or perception literal involving α, there exists a $w \in W(\Gamma)$ such that $\mathcal{S}, w \models \delta\omega$. Then there exists an $LA \in SoLA(\Gamma)$ such that for all $w \in W(\Gamma)$, $Z(F(\Gamma, \alpha, LA, w)) \neq \emptyset$ and $Z(G(\Gamma, \alpha, LA, w)) \neq \emptyset$.

[1] He actually uses the symbol E for existential quantification.

Proof:
We prove the contrapositive of the lemma. Assume that for all $LA \in SoLA(\Gamma)$, there exists a $w \in W(\Gamma)$ such that $Z(FG(\Gamma, \alpha, LA, w)) = \emptyset$.

Let LA be an arbitrary label assignment in $SoLA(\Gamma)$. Let $W(\Gamma)^{\#} = (w_1, w_2, \ldots, w_n)$ be an ordering of the worlds in $W(\Gamma)$, where $n = |W(\Gamma)|$. Let $\Omega^{\#} = (\varsigma_1, \varsigma_2, \ldots, \varsigma_m)$ be an ordering of the observations in Ω, where $m = |\Omega|$. Let $W = W(\Gamma)$. Let w be an arbitrary world in W and let α be an arbitrary action in $\mathcal{A}$.

Let $\{[\alpha]_{q_1^\alpha}\varphi_1^\alpha,\ [\alpha]_{q_2^\alpha}\varphi_2^\alpha,\ \ldots,\ [\alpha]_{q_g^\alpha}\varphi_g^\alpha,\ \neg[\alpha]_{q_{g+1}^\alpha}\varphi_{g+1}^\alpha,\ \neg[\alpha]_{q_{g+2}^\alpha}\varphi_{g+2}^\alpha,\ \ldots,\ \neg[\alpha]_{q_{g+h}^\alpha}\varphi_{g+h}^\alpha\}$ be an ordered set of the dynamic literals in $F(\Gamma, \alpha, LA, w)$.

Let $\{(\varsigma_1 \mid \alpha : q_1), \ldots, (\varsigma_t \mid \alpha : q_t), \neg(\varsigma_{t+1} \mid \alpha : q_{t+1}), \ldots, \neg(\varsigma_{t+v} \mid \alpha : q_{t+v})\}$ be an ordered set of the perception literals in $G(\Gamma, \alpha, LA, w)$.

If $Z(F(\Gamma, \alpha, LA, w_k) = \emptyset$, there is no solution $(s_1^\alpha, s_2^\alpha, \ldots, s_n^\alpha)$ for which

$$\begin{aligned}
\sum_{\substack{i=1 \\ R_\alpha(w_k,w_i)=s_i^\alpha \\ \mathcal{S},w_i \models \varphi_1^\alpha}}^{n} s_i^\alpha &= q_1^\alpha \quad \text{and} \\
\sum_{\substack{i=1 \\ R_\alpha(w_k,w_i)=s_i^\alpha \\ \mathcal{S},w_i \models \varphi_2^\alpha}}^{n} s_i^\alpha &= q_2^\alpha \quad \text{and} \\
&\vdots \\
\sum_{\substack{i=1 \\ R_\alpha(w_k,w_i)=s_i^\alpha \\ \mathcal{S},w_i \models \varphi_{g+h}^\alpha}}^{n} s_i^\alpha &\neq q_{g+h}^\alpha
\end{aligned}$$

such that $\sum_{w' \in W} R_\alpha(w_k, w') = 1$ or $\sum_{w' \in W} R_\alpha(w_k, w') = 0$.

If $Z(G(\Gamma, \alpha, LA, w_k) = \emptyset$, there is no solution $(s_1^\alpha, s_2^\alpha, \ldots, s_n^\alpha)$ for which

$$\begin{aligned}
Q_\alpha(w_k, N(\sigma_1)) &= s_1^\sigma \quad \text{and} \\
Q_\alpha(w_k, N(\sigma_2)) &= s_2^\sigma \quad \text{and} \\
&\vdots \\
Q_\alpha(w_k, N(\sigma_{t+v})) &= s_{t+v}^\sigma,
\end{aligned}$$

where $s_1^\sigma, s_2^\sigma, \ldots, s_{t+v}^\sigma \in \{s_1^\varsigma, s_2^\varsigma, \ldots, s_m^\varsigma\}$ and such that if $R_\alpha(w, w') > 0$, then $\sum_{o \in O} Q_\alpha(w', o) = 1$ (due to tableau rule obs), else if $R_\alpha(w, w') = 0$, then either $\sum_{o \in O} Q_\alpha(w', o) = 1$ or $\sum_{o \in O} Q_\alpha(w', o) = 0$.

Thus, there is no way to assign transition and perception probabilities such that R_α and Q_α conform to the definition of a SLAOP structure. That is, there is no SLAOP structure $\mathcal{S}$ such that for all $(x, \delta\omega) \in \Gamma$ there exists a $w \in W$ such that $\mathcal{S}, w \models \delta\omega$. Hence, the lemma holds. ■

Lemma 6.3.2: Let T be a finished tree. For every node Γ in T: If there exists a structure $\mathcal{S}$ such that for all $(x, \Phi) \in \Gamma$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$, then the (sub)tree rooted at Γ is open.

Proof:
(by induction on the height of the node Γ_k)

Base case:

Height $h = 0$; Γ_k is a leaf. If there exists a structure $\mathcal{S}$ such that for all $(x, \Phi) \in \Gamma_k$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$, then $(x', \bot) \notin \Gamma_k$ for all x'. Hence, the sub-tree consisting of Γ_k is open.

Induction step:

If $h > 0$, then some rule was applied to create the child(ren) $\Gamma_{k'}$ of Γ_k. We abbreviate "there exists a structure $\mathcal{S}^j = \langle W^j, R^j, O^j, N^j, Q^j, U^j \rangle$ such that for all $(x^j, \Phi^j) \in \Gamma_j$ there exists a $w^j \in W^j$ such that $\mathcal{S}^j, w^j \models \Phi^j$" as $A(j)$ and we abbreviate "the (sub)tree rooted at Γ_j is open" as $B(j)$.

We must show the following for every rule/phase. IF: If $A(k')$, then $B(k')$, THEN: If $A(k)$, then $B(k)$. We assume the antecedent (induction hypothesis): If $A(k')$, then $B(k')$. To show the consequent, we must assume $A(k)$ and show that $B(k)$ follows.

Note that if the (sub)tree rooted at $\Gamma_{k'}$ is open, then the (sub)tree rooted at Γ_k is open. That is, if $B(k')$ then $B(k)$. So we want to show $B(k')$. But, by the induction hypothesis, $B(k')$ follows from $A(k')$. Therefore, it will suffice, in each case below, to assume $A(k)$, and prove $A(k')$.

- rule $=$:
  For the rule to have been applied, $(x, (c = c')) \in \Gamma_k$ or $(x, \neg(c = c')) \in \Gamma_k$. The rule is only applied when $(c = c')$, resp., $\neg(c = c')$ is unsatisfiable. Therefore, Γ_k is unsatisfiable. But this contradicts our main assumption $A(k)$. Hence, rule $=$ could not have been applicable to Γ_k.

- rule $\bot$:
  For the rule to have been applied, $\{(x, \Psi), (x, \neg\Psi)\} \subseteq \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \bot)\}$. But there exists no structure $\mathcal{S}^k = \langle W^k, R^k \rangle$ such that there exists a $w^k \in W^k$ such that $\mathcal{S}^k, w^k \models \Psi$ and $\mathcal{S}^k, w^k \models \neg\Psi$. Hence, assumption $A(k)$ is false and this rule could not have been applied.

- rule $\neg$:
  For the rule to have been applied, $(x, \neg\neg\Psi) \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \Psi)\}$. By assumption, $\mathcal{S}^k, w^k \models \neg\neg\Psi$. Hence, $\mathcal{S}^k, w^k \models \Psi$. Thus, $A(k')$.

- rule $\wedge$:
  For the rule to have been applied, $(x, \Psi \wedge \Psi') \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \Psi), (x, \Psi')\}$. By assumption, $\mathcal{S}^k, w^k \models \Psi \wedge \Psi'$. Hence, $\mathcal{S}^k, w^k \models \Psi$ and $\mathcal{S}^k, w^k \models \Psi'$. Thus, $A(k')$.

- rule $\vee$:
  For the rule to have been applied, $(x, \Psi \vee \Psi') \in \Gamma_k$, and after its application, either $\Gamma_{k'} = \Gamma_k \cup \{(x, \Psi)\}$ or $\Gamma_{k''} = \Gamma_k \cup \{(x, \Psi')\}$. By assumption, $\mathcal{S}^k, w^k \models \Psi \vee \Psi'$. Hence,

$\mathcal{S}^k, w^k \models \Psi$ or $\mathcal{S}^k, w^k \models \Psi'$. Thus, $A(k')$ or $A(k'')$. Thus, $B(k')$ or $B(k'')$. Therefore, $B(k)$.

- rule $\Diamond\varphi$:
  For the rule to have been applied, $(0, \neg[\alpha]_0\varphi) \in \Gamma_k$ or $(0, [\alpha]_q\varphi) \in \Gamma_k$ for $q > 0$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \varphi)\}$ where x is a fresh integer.

  By assumption, there exists a $w \in W$ such that $\mathcal{S}, w \models \neg[\alpha]_0\varphi$ or $\mathcal{S}, w \models [\alpha]_q\varphi$. Then by definition of $\langle\alpha\rangle$, there exists a $w'' \in W$ such that $(w, w'', pr) \in R_\alpha$ for $pr > 0$ and $\mathcal{S}, w'' \models \varphi$. Hence, for all $(x, \Phi') \in \Gamma_{k'}$ there exists a $w' \in W$ such that $\mathcal{S}, w' \models \Phi'$. Thus, $A(k')$.

- rule obs:
  For the rule to have been applied, $(x, \neg[\alpha]_0\varphi) \in \Gamma_k$ or $(x, [\alpha]_q\varphi) \in \Gamma_k$ for $q > 0$ and some x, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(0, \Box(\delta_1 \to (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \Box(\delta_2 \to (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \cdots \vee \Box(\delta_n \to (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)))\}$, where $\delta_i \in cpt(\varphi)$.

  By assumption, there exists a $w \in W$ such that $\mathcal{S}, w \models \neg[\alpha]_0\varphi$ or $\mathcal{S}, w \models [\alpha]_q\varphi$. That is, there exists a $w'' \in W$ such that $R_\alpha(w, w'') > 0$ and $\mathcal{S}, w'' \models \varphi$.

  By definition of a SLAOP structure, for all $w^-, w^+ \in W$: if $R_\alpha(w^-, w^+) > 0$, then $\sum_{o\in O}, Q_\alpha(w^+, o) = 1$. Thus, there exists a $w'' \in W$ such that $\sum_{o\in O} Q_\alpha(w'', o) = 1$, where $\mathcal{S}, w'' \models \varphi$. This implies that there exists a $w'' \in W$ such that $w'' \models \varphi$ and there exists at least one observation $\varsigma \in \Omega$ such that it is not the case that $Q_\alpha(w'', N(\varsigma)) = 0$. Hence, for all $w' \in W$ if $\mathcal{S}, w' \models \delta_1$ or $\mathcal{S}, w' \models \delta_2$ or $\ldots$ or $\mathcal{S}, w' \models \delta_n$, then $(\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)$, where $\delta_i \in cpt(\varphi)$. Therefore, there exists a $w \in W$ such that $\mathcal{S}, w \models \Box(\delta_1 \to (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \Box(\delta_2 \to (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \cdots \vee \Box(\delta_n \to (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0))$. Thus, $A(k')$.

- rule $\Box$:
  For the rule to have been applied, $\{(0, \Box\Phi), (x, \Phi'')\} \subseteq \Gamma_k$ for some $x \geq 0$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \Phi)\}$.

  By assumption, there exist $w, w' \in W$ such that $\mathcal{S}, w \models \Box\Phi$ and $\mathcal{S}, w' \models \Phi''$. Then by definition of $\Box$, for all $w'' \in W, \mathcal{S}, w'' \models \Phi$. That is, there exists a $w'' \in W$ such that $\mathcal{S}, w'' \models \Phi$. Hence, for all $(x, \Phi') \in \Gamma_{k'}$ there exists a $w''' \in W$ such that $\mathcal{S}, w''' \models \Phi'$. Thus, $A(k')$.

- rule $\Diamond$:
  For the rule to have been applied, $(0, \neg\Box\Phi) \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(x, \neg\Phi)\}$ where x is a fresh integer.

  By assumption, there exists a $w \in W$ such that $\mathcal{S}, w \models \neg\Box\Phi$. Then by definition of $\Box$, there exists a $w' \in W$ such that $\mathcal{S}, w' \not\models \Phi$. That is, there exists a $w' \in W$ such that $\mathcal{S}, w' \models \neg\Phi$. Hence, for all $(x, \Phi') \in \Gamma_{k'}$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi'$. Thus, $A(k')$.

- Label Assignment Phase:
  If it can be shown that there exists an $LA \in SoLA(\Gamma_k)$ such that

  1. for no $w \in W(\Gamma_k)$, $E(\Gamma_k, LA, w)$ contains

- $Reward(r)$ and $Reward(r')$ such that $r \neq r'$, or
- $Reward(r)$ and $\neg Reward(r)$, or
- $Cost(\alpha, c)$ and $Cost(\alpha, c')$ (same action α) such that $c \neq c'$, or
- $Cost(\alpha, c)$ and $\neg Cost(\alpha, c)$ (same action α);

2. for all $\alpha \in \mathcal{A}$ and all $w \in W(\Gamma_k)$, $Z(F(\Gamma_k, \alpha, LA, w)) \neq \emptyset$ and $Z(G(\Gamma_k, \alpha, LA, w)) \neq \emptyset$,

then no child is created for Γ_k and trivially, for all $(x, \Phi') \in \Gamma_{k'}$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi'$.

Let LA be a member of $SoLA(\Gamma_k)$. By assumption, there exists a structure $\mathcal{S} = \langle W, R, O, N, Q, U \rangle$ such that for all $(x, \Phi) \in \Gamma_k$ there exists a $w \in W$ such that $\mathcal{S}, w \models \Phi$. The label assignment phase occurs only when Γ_k is the leaf node of a saturated tree. Thus, $w \in W(\Gamma_k, x)$. Each of the three cases is considered separately.

(1) For all $(x, \Phi) \in \Gamma_k$, either $x{:}w \in LA$ or $x{:}w \notin LA$. If $x{:}w \in LA$, then by the assumption, the first case must be true. If $x{:}w \notin LA$, then $\Phi \notin E(\Gamma_k, LA, w)$, and the first case is trivially true.

(2) By assumption, there exists a structure $\mathcal{S} = \langle W, R, O, N, Q, U \rangle$ such that for all $(x, \delta\omega) \in \Gamma_k$, where $\delta\omega$ is a dynamic or perception literal involving α, there exists a $w \in W$ such that $\mathcal{S}, w \models \delta\omega$.

Due to $w \in W(\Gamma_k, x)$, if $w \in W$, then $w \in W(\Gamma_k)$. That is, $W \subseteq W(\Gamma_k)$. Thus, a SLAOP structure $\mathcal{S}' = \langle W(\Gamma_k), R', O, N, Q', U \rangle$ can be constructed as follows. For all $\alpha \in \mathcal{A}$, for all $w, w' \in W(\Gamma_k)$, if $R_\alpha(w, w') = pr$, let $R'_\alpha(w, w') = pr$, else let $R'_\alpha(w, w') = 0$. Hence, for all $(x, \delta) \in \Gamma_k$ there exists a $w \in W(\Gamma_k)$ such that $\mathcal{S}', w \models \delta$, where δ is a dynamic literal involving α. And for all $\alpha \in \mathcal{A}$, for all $o \in O$, for all $w' \in W(\Gamma_k)$, if $Q_\alpha(w', o) = pr$, let $Q'_\alpha(w', o) = pr$, else let $Q'_\alpha(w', o) = 0$. Hence, for all $(x', \omega) \in \Gamma_k$ there exists a $w' \in W(\Gamma_k)$ such that $\mathcal{S}', w' \models \omega$, where ω is a perception literal involving α.

Then, by Lemma 6.3.1, there exists an $LA \in SoLA(\Gamma_k)$ such that for all $w \in W(\Gamma_k)$ and $\alpha \in \mathcal{A}$, $Z(F(\Gamma_k, \alpha, LA, w)) \neq \emptyset$ and $Z(G(\Gamma_k, \alpha, LA, w)) \neq \emptyset$.

Moreover, the systems of inequalities (SIs), as used in this chapter, can be described in the language of first-order logic elementary real number theory [Tarski, 1957, Seidenberg, 1954, Collins, 1975] (cf. Lem. 6.2.1). There exist sound methods for determining whether an SI is feasible [Tarski, 1957, Seidenberg, 1954, Collins, 1975]. In other words, there is a reliable means of determining whether there exists at least one solution to an SI. Action α is arbitrary; the second case is true.

■

Theorem 6.3.1: (Soundness) If $\vdash \Psi$ then $\models \Psi$. (Contrapositively, if $\not\models \Psi$ then $\not\vdash \Psi$.)

Proof:
Let $\psi = \neg\Psi$. Then $\not\vdash \Psi$ if and only if the tree for ψ is open. And

$$\begin{aligned} \not\models \Psi &\iff \text{not } (\forall \mathcal{S})\ \mathcal{S} \models \Psi \\ &\iff \text{not } (\forall \mathcal{S}, w)\ \mathcal{S}, w \models \Psi \\ &\iff (\exists \mathcal{S}, w)\ \mathcal{S}, w \models \psi. \end{aligned}$$

For the soundness proof, it thus suffices to show that if there exists a structure $\mathcal{S}$ and w in it such that $\mathcal{S}, w \models \psi$, then the tree rooted at $\Gamma_0^0 = \{(0, \psi)\}$ is open.

The following corollary is a consequent of the special case when $k = 0$ of Γ_k mentioned in the proof of Lemma 6.3.2. For every finished tree of a sentence Ψ, if there exists a structure $\mathcal{S}$ and a $w \in W$ of $\mathcal{S}$ such that $\mathcal{S}, w \models \Psi$, then the tree is open.

It is known that the first-order theory of rational numbers (linear arithmetic; without multiplication) is decidable; the Fourier-Motzkin method [Motzkin, 1936] and Dines' paper [Dines, 1919], for example, are proofs of this, and Ferrante and Rackoff [1975]'s method is a more efficient (almost polynomial) variant. Any system of equations and disequalities as they appear in this work, can easily be stated as an applicable first-order theory (see [Kroening and Strichman, 2008], e.g., and the appendix). In other words, there is a reliable means of determining whether there exists at least one solution to an SLI. Therefore, given the corollary to Lemma 6.3.2 stated above, every execution of a rule or procedure in the decision procedure is sound. ■

Lemma 6.3.3: $\mathcal{S}$ is a SLAOP structure.

Proof:
The components of the structure are well-formed:

- $W = W(\Gamma) = \bigcup_{x \geq 0}\{w \in C \mid w \models \ell \text{ for all } (x, \ell) \in \Gamma \text{ where } \ell \text{ is a propositional literal}\}$. That is, $W = \{w \in C \mid \text{for all } x, w \models \ell \text{ for all } (x, \ell) \in \Gamma \text{ where } \ell \text{ is a propositional literal}\}$. Thus, for W to be empty, it must be the case that for all $w \in C$, there exists some $(x, \ell) \in \Gamma$, for which $w \not\models \ell$. But this is a contradiction. Hence, W is not empty.

- Due to Γ being open (and by rule SI), we know that for all $\alpha \in \mathcal{A}$ and all $w \in W(\Gamma)$, there exists a solution in $Z(F(\Gamma, \alpha, LA, w))$.

  By construction, R maps each action $\alpha \in \mathcal{A}$ to R_α such that R_α is a relation in $(W \times W) \times [0, 1]$. Moreover, by the nature of the SI generated from $F(\Gamma, \alpha, LA, w)$, R_α is a (total) function $R_\alpha : (W \times W) \mapsto [0, 1]$.

  And by construction, the fact that $pr_1 + pr_2 + \cdots + pr_n = 1$ or $pr_1 + pr_2 + \cdots + pr_n = 0$ is an equation in any SI generated, either $\sum_{w' \in W} R_\alpha(w, w') = 1$ or $\sum_{w' \in W} R_\alpha(w, w') = 0$, for every $w \in W$.

- Assuming Ω is non-empty, O will be non-empty.

- By construction, $N : \Omega \mapsto O$ is a bijection.

- Due to Γ being open (and by rule SI), we know that for all $\alpha \in \mathcal{A}$ and all $w \in W(\Gamma)$, there

exists a solution in $Z(G(\Gamma, \alpha, LA, w))$.

By construction, Q maps each action $\alpha \in \mathcal{A}$ to Q_α such that Q_α is a relation in $(W \times O) \times [0, 1]$.

Moreover, by the nature of the SI generated from $G(\Gamma, \alpha, LA, w)$, Q_α is a (total) function $Q_\alpha : (W \times O) \mapsto [0, 1]$.

And by construction, the fact that in any SI generated, there is an equation $pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_m^\varsigma = \lceil pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_m^\varsigma \rceil$ and the fact that $\lceil pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_m^\varsigma \rceil$ equals 0 or 1, it must be the case that $\sum_{o \in O} Q_\alpha(w, o)$ equals 0 or 1. Furthermore, due to tableau rule obs, if there exists a w' such that $R_\alpha(w', w) > 0$, then $Q_\alpha(w, o) > 0$, which implies that $\lceil pr_1^\varsigma + pr_2^\varsigma + \cdots + pr_m^\varsigma \rceil = 1$, which implies that: For all $w, w' \in W$: if $R_\alpha(w', w) > 0$ for some w', then $\sum_{o \in O} Q_\alpha, (w, o) = 1$.

- By construction, $U = \langle Re, Co \rangle$, where $Re : W \mapsto \mathbb{R}$ and Co is a mapping from $\mathcal{A}$ to a function $Co_\alpha : C \mapsto \mathbb{R}$. Suppose $x{:}w, x'{:}w \in LA$ where $x \neq x'$. If $(x, Reward(r))$, $(x', Reward(r')) \in \Gamma$ such that $r \neq r'$, then by construction $Re(w) = r = r'$ which is impossible. But if $(x, Reward(r)), (x', Reward(r'))$ were in Γ, then LA would have caused $Reward(r)$ and $Reward(r')$ to be in $E(\Gamma, LA, w)$ and the branch would have closed. So either (i) $x{:}w, x'{:}w \in LA$ but $Reward(r)$ and $Reward(r')$ are not both in Γ, or (ii) $(x, Reward(r))$, $(x', Reward(r')) \in \Gamma$ but $x{:}w$ and $x'{:}w$ are not both in LA, or (iii) $x{:}w$ and $x'{:}w$ are not both in LA and $Reward(r)$ and $Reward(r')$ are not both in Γ.

■

Lemma 6.3.5: Let Γ be the leaf node of an open branch of a finished tree. We know that there exists a label assignment $LA \in SoLA(\Gamma)$ such that $Z(F(\Gamma, \alpha, LA, w))$ and $Z(G(\Gamma, \alpha, LA, w))$ are not empty, for all $w \in W(\Gamma)$ and all $\alpha \in \mathcal{A}$. If $\mathcal{S}$ is constructed as described above, then for all $(x, \Psi) \in \Gamma$, $\mathcal{S}, w \models \Psi$ for $x{:}w \in LA$.

Proof:
The proof will be by induction on the structure of a formula.

The induction step will work as follows. Let $\gamma' \subseteq \Gamma$ be added to Γ due to some rule applied to $\gamma \subseteq \Gamma$. Thus, we need to prove that IF for all $(x', \Psi') \in \gamma'$, $\mathcal{S}, w' \models \Psi'$ for $x'{:}w' \in LA$, THEN for all $(x, \Psi) \in \gamma$, $\mathcal{S}, w \models \Psi$ for $x{:}w \in LA$.

We assume the antecedent (induction hypothesis).

Base case:

- Ψ is a propositional literal. By definition, $\mathcal{S}, w' \models \Psi$ for all $w' \in W(\Gamma, x)$. But if $x{:}w \in LA$, then $w \in W(\Gamma, x)$. Thus $\mathcal{S}, w \models \Psi$.
- Ψ is $c = c'$. Because $(x, \bot) \notin \Gamma$ for some label x, rule $=$ was not applied. Hence, c is identical to c', and $\mathcal{S}, w \models c = c'$.
- Ψ is $\neg(c = c')$. Because $(x, \bot) \notin \Gamma$ for some label x, rule $=$ was not applied. Hence, c is not identical to c', and $\mathcal{S}, w \models \neg(c = c')$.

- Ψ is $Reward(r)$. By construction, $(w, r) \in Re$ for $(x, Reward(r)) \in \Gamma$ and $x{:}w \in LA$. Hence, $\mathcal{S}, w \models Reward(r)$.
- Ψ is $\neg Reward(r)$. By construction, $(w, r) \notin Re$. Hence, $\mathcal{S}, w \models \neg Reward(r)$.
- Ψ is $Cost(\alpha, c)$. By construction, $(w, c) \in Co_\alpha$ for $(x, Cost(\alpha, c)) \in \Gamma$ and $x{:}w \in LA$. Hence, $\mathcal{S}, w \models Cost(\alpha, c)$.
- Ψ is $\neg Cost(\alpha, c)$. By construction, $(w, c) \notin Co_\alpha$. Hence, $\mathcal{S}, w \models \neg Cost(\alpha, c)$.
- Ψ is $[\alpha]_q\varphi$. Then as a direct consequence of the construction of $\mathcal{S}$, $\mathcal{S}, w \models [\alpha]_q\varphi$.
- Ψ is $\neg[\alpha]_q\varphi$. Then as a direct consequence of the construction of $\mathcal{S}$, $\mathcal{S}, w \models \neg[\alpha]_q\varphi$.
- Ψ is $(\varsigma \mid \alpha : q)$. Then as a direct consequence of the construction of $\mathcal{S}$, $\mathcal{S}, w \models (\varsigma \mid \alpha : q)$.
- Ψ is $\neg(\varsigma \mid \alpha : q)$. Then as a direct consequence of the construction of $\mathcal{S}$, $\mathcal{S}, w \models \neg(\varsigma \mid \alpha : q)$.

Induction step:

- Ψ is $\neg\neg\psi$. By rule $\neg$, $(x, \psi) \in \Gamma$. By induction hypothesis, $\mathcal{S}, w \models \psi$ for $x{:}w \in LA$. By the definition of $\neg$, $\mathcal{S}, w \models \neg\neg\psi$.
- Ψ is $\psi \wedge \psi'$. By rule $\wedge$, $(x, \psi), (x, \psi') \in \Gamma$. By induction hypothesis, $\mathcal{S}, w \models \psi$ and $\mathcal{S}, w \models \psi'$ for $x{:}w \in LA$. By the definition of $\wedge$, $\mathcal{S}, w \models \psi \wedge \psi'$.
- Ψ is $\neg(\psi \wedge \psi')$. By rule $\vee$, $(x, \neg\psi) \in \Gamma$ or $(x, \neg\psi') \in \Gamma$ for $x{:}w \in LA$. By induction hypothesis, $\mathcal{S}, w \models \neg\psi$ or $\mathcal{S}, w \models \neg\psi'$. By the definition of $\vee$, $\mathcal{S}, w \models \neg(\psi \wedge \psi')$.
- Ψ is $\Box\Phi$. Let $X(\Gamma) = \{0, 1, 2, \ldots, x'\}$. Due to successive applications of rule $\Box$, $(0, \Phi)$, $(1, \Phi)$, …, $(x', \Phi) \in \Gamma$. Then, by induction hypothesis, $\mathcal{S}, w_0 \models \Phi$ for $0{:}w_0 \in LA$ and $\mathcal{S}, w_1 \models \Phi$ for $1{:}w_1 \in LA$ and $\cdots$ and $\mathcal{S}, w_{x'} \models \Phi$ for $x'{:}w_{x'} \in LA$.

  We need to show that $\mathcal{S}, w \models \Box\Phi$ for $0{:}w_0 \in LA$, that is, that for all $w' \in W(\Gamma)$, $\mathcal{S}, w' \models \Phi$. This will be the case if: For every $w' \in W(\Gamma)$, there exists a term $t := \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m$ of Φ such that $\mathcal{S}, w' \models t$. Note that for $w_i, w_j \in W(\Gamma)$, if $w_i \neq w_j$, it is sufficient that there exist terms t_i and t_j of Φ such that $\mathcal{S}, w_i \models t_i$ and $\mathcal{S}, w_j \models t_j$, even if $t_i \neq t_j$.

  By Lemma 6.3.4, for every label $x \in X(\Gamma)$, there exists a term $t := \Phi_1 \wedge \Phi_2 \wedge \cdots \wedge \Phi_m$ of Φ such that (x, Φ_1), (x, Φ_2), …, $(x, \Phi_m) \in \Gamma$.

  Then we define the set

$$L(t) := \{\Phi_i \mid \Phi_i \text{ is a propositional literal conjunct of } t\},$$

  the set

$$ERC(t) := \{\Phi_i \mid \Phi_i \text{ is an erc literal conjunct of } t\}$$

  the set

$$\Delta(t) := \{\Phi_i \mid \Phi_i \text{ is a dynamic literal conjunct of } t\}$$

  and the set

$$\Omega(t) := \{\Phi_i \mid \Phi_i \text{ is a or perception literal conjunct of } t\}.$$

Note that $t \equiv \bigwedge_{\ell \in L(t)} \ell \wedge \bigwedge_{\rho \in ERC(t)} \rho \wedge \bigwedge_{\delta\omega \in \Delta\Omega(t)} \delta\omega$.

Let $\ell \in L(t)$. Then by induction hypothesis, $\mathcal{S}, w'' \models \ell$ for $x{:}w'' \in LA$. Note that if $w'' \models \ell$ for some $w'' \in W(\Gamma, x)$, then $w^* \models \ell$ for all $w^* \in W(\Gamma, x)$. Thus,

$$\mathcal{S}, w^* \models \bigwedge_{\ell \in L(t)} \ell \text{ (for all } w^* \in W(\Gamma, x)),$$

and by definition of $W(\Gamma)$,

$$\mathcal{S}, w' \models \bigwedge_{\ell \in L(t)} \ell \text{ (for all } w' \in W(\Gamma)). \tag{A.6}$$

Let $\rho \in ERC(t)$. Then by induction hypothesis, $\mathcal{S}, w'' \models \rho$ for $x{:}w'' \in LA$. If ρ is $(b = b')$ or $\neg(b = b')$, then

$$\mathcal{S}, w''' \models \rho \text{ (for all } w''' \in W(\Gamma)). \tag{A.7}$$

By construction (see base case), for all $w' \in W(\Gamma)$,

$$\mathcal{S}, w' \models \bigwedge_{\rho \in E(\Gamma, LA, w)} \rho \tag{A.8}$$

and

$$\mathcal{S}, w' \models \bigwedge_{\delta \in F(\Gamma, LA, w)} \delta. \tag{A.9}$$

and

$$\mathcal{S}, w' \models \bigwedge_{\omega \in G(\Gamma, LA, w)} \omega. \tag{A.10}$$

By (A.7), (A.8) and Lemma 6.3.4, for all $w' \in W(\Gamma)$,

$$\mathcal{S}, w' \models \bigwedge_{\rho \in ERC(t)} \rho.$$

By (A.9) and Lemma 6.3.4, for all $w' \in W(\Gamma)$,

$$\mathcal{S}, w' \models \bigwedge_{\delta \in \Delta(t)} \delta.$$

By (A.10) and Lemma 6.3.4, for all $w' \in W(\Gamma)$,

$$\mathcal{S}, w' \models \bigwedge_{\omega \in \Omega(t)} \omega.$$

Hence, for all $w' \in W(\Gamma)$, there exists a $x \in X(\Gamma)$ such that

$$\mathcal{S}, w' \models \bigwedge_{\substack{i=1 \\ (x,\Phi_i)\in\Gamma}}^{m} \Phi_i,$$

(where $\bigwedge_{\substack{i=1 \\ (x,\Phi_i)\in\Gamma}}^{m} \Phi_i$ is a term of Φ) which implies that for all $w' \in W(\Gamma)$,

$$\mathcal{S}, w' \models \Phi,$$

which concludes the proof.

- Ψ is $\neg\Box\Phi$. By rule $\Diamond$, $(x', \neg\Phi) \in \Gamma$ for some $x' > x$. By induction hypothesis, $\mathcal{S}, w' \models \neg\Phi$ for $x'{:}w' \in LA$. That is, it is not the case that for all $w'' \in W(\Gamma), \mathcal{S}, w'' \models \Phi$. Hence, $\mathcal{S}, w \not\models \Box\Phi$, if and only if $\mathcal{S}, w \models \neg\Box\Phi$.

■

Theorem 6.3.2: (Completeness) If $\models \Psi$ then $\vdash \Psi$. (Contrapositively, if $\nvdash \Psi$ then $\not\models \Psi$.)

Proof:
Let $\psi = \neg\Psi$. Then $\nvdash \Psi$ means that there is an open branch of a finished tree for ψ. And

$$\begin{aligned} \not\models \Psi &\iff (\exists\mathcal{S})\ \mathcal{S} \not\models \Psi \\ &\iff (\exists\mathcal{S}, w)\ \mathcal{S}, w \not\models \Psi \\ &\iff (\exists\mathcal{S}, w)\ \mathcal{S}, w \models \psi. \end{aligned}$$

For the completeness proof, it thus suffices to construct for some open branch of a finished tree for $\psi \in \mathcal{L}_{SLAOP}$, a SLAOP structure $\mathcal{S} = \langle W, R, O, N, Q, U\rangle$ in which there is a world $w \in W$ in $\mathcal{S}$ such that ψ is satisfied in $\mathcal{S}$ at w.

By Lemmata 6.3.3 and 6.3.5, given the leaf node Γ of an open branch of a finished tree, there exists a structure $\mathcal{S}$ such that for all $(x, \Psi) \in \Gamma$, $\mathcal{S}, w \models \Psi$ for $x{:}w \in LA$. But $(0, \psi) \in \Gamma$. Thus, if there is a finished open tableau for ψ, then ψ is satisfiable. The theorem follows directly. ■

A.4 SDL

Lemma 7.3.1: Let T be a finished tree. For every node Γ in T: If there exists a structure $\mathcal{D}$ such that for all $(\Sigma, \Phi) \in \Gamma$ there exists a belief-state $b \in P$ and a world $w \in C$ such that $\mathcal{D}bw \models \Phi$, then the (sub)tree rooted at Γ is open.

Proof:
The proof will be by induction on the height of the node Γ_k.

Base case:

Height $h = 0$; Γ_k is a leaf. If there exists a structure $\mathcal{D}$ such that for all $(\Sigma, \Phi) \in \Gamma_k$ there exists a b and a w such that $\mathcal{D}bw \models \Phi$, then $(\Sigma', \bot) \notin \Gamma_k$ for all Σ'. Hence, the sub-tree consisting of Γ_k is open.

Induction step:

If $h > 0$, then some rule was applied to create the child(ren) $\Gamma_{k'}$ of Γ_k. We abbreviate "there exists a structure $\mathcal{D}^j = \langle R^j, Q^j, U^j \rangle$ such that for all $(\Sigma^j, \Phi^j) \in \Gamma_j$ there exists a b^j and a w^j such that $\mathcal{D}^j b^j w^j \models \Phi^j$" as $A(j)$ and we abbreviate "the (sub)tree rooted at Γ_j is open" as $B(j)$.

We must show the following for every rule/phase. IF: If $A(k')$, then $B(k')$, THEN: If $A(k)$, then $B(k)$, where $\Gamma_{k'}$ was created due to the application of some tableau rule or the SI phase to Γ_k. We assume the antecedent (induction hypothesis): If $A(k')$, then $B(k')$. To show the consequent, we must assume $A(k)$ and show that $B(k)$ follows.

Note that if the (sub)tree rooted at $\Gamma_{k'}$ is open, then the (sub)tree rooted at Γ_k is open. That is, if $B(k')$ then $B(k)$. So we want to show $B(k')$. But, by the induction hypothesis, $B(k')$ follows from $A(k')$. Therefore, it will suffice, in each case below, to assume $A(k)$, and prove $A(k')$.

- rule $\neg$:
  For the rule to have been applied, $(\Sigma, \Psi) \in \Gamma_k$ such that Ψ has a double negation somewhere in it, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma, \Psi')\}$, where Ψ' is Ψ with the double negation removed. By assumption, $\mathcal{D}^k b^k w^k \models \Psi$. Hence, $\mathcal{D}^k b^k w^k \models \Psi'$. Thus, $A(k')$.

- rule $\wedge$:
  For the rule to have been applied, $(\Sigma, \Psi \wedge \Psi') \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma, \Psi), (\Sigma, \Psi')\}$. By assumption, $\mathcal{D}^k b^k w^k \models \Psi \wedge \Psi'$. Hence, $\mathcal{D}^k b^k w^k \models \Psi$ and $\mathcal{D}^k b^k w^k \models \Psi'$. Thus, $A(k')$.

- rule $\vee$:
  For the rule to have been applied, $(\Sigma, \Psi \vee \Psi') \in \Gamma_k$, and after its application, either $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma, \Psi)\}$ or $\Gamma_{k''} = \Gamma_k \cup \{(\Sigma, \Psi')\}$. By assumption, $\mathcal{D}^k b^k w^k \models \Psi \vee \Psi'$. Hence, $\mathcal{D}^k b^k w^k \models \Psi$ or $\mathcal{D}^k b^k w^k \models \Psi'$. Thus, $A(k')$ or $A(k'')$. Thus, $B(k')$ or $B(k'')$. Therefore, $B(k)$.

- rule $=$:
  For the rule to have been applied, $(\Sigma, c = c') \in \Gamma_k$ or $(\Sigma, \neg(c = c')) \in \Gamma_k$. The rule is only applied when $(c = c')$, resp., $\neg(c = c')$ is unsatisfiable. Therefore, Γ_k is unsatisfiable. But this contradicts our main assumption $A(k)$. Hence, rule $=$ could not have been applicable to Γ_k.

- rule $\Rightarrow \wedge$:
  For the rule to have been applied, $(\Sigma, \varphi \Rightarrow \Phi \wedge \Phi') \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma, (\varphi \Rightarrow \Phi) \wedge (\varphi \Rightarrow \Phi'))\}$. By assumption, $\mathcal{D}^k b^k w^k \models \varphi \Rightarrow \Phi \wedge \Phi'$. Hence, $\mathcal{D}^k b^k w^k \models \varphi \Rightarrow \Phi$ and $\mathcal{D}^k b^k w^k \models \varphi \Rightarrow \Phi'$. Thus, $A(k')$.

- rule $\delta \Rightarrow$:
  For the rule to have been applied, $(\Sigma, \varphi \Rightarrow \Phi) \in \Gamma_k$ where Φ is a disjunction of literals, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma, \delta_1 \Rightarrow \Phi), (\Sigma, \delta_2 \Rightarrow \Phi), \ldots, (\Sigma, \delta_n \Rightarrow \Phi)\}$, where $\delta_i \in cpt(\varphi)$. By assumption, $\mathcal{D}^k b^k w^k \models \varphi \Rightarrow \Phi$. Hence, $\mathcal{D}^k b^k w^k \models \delta_1 \Rightarrow \Phi$ and $\mathcal{D}^k b^k w^k \models \delta_2 \Rightarrow \Phi$ and $\ldots$ and $\mathcal{D}^k b^k w^k \models \delta_n \Rightarrow \Phi$. Thus, $A(k')$.

- rule $\Rightarrow \vee$:
  For the rule to have been applied, $(\Sigma, \varphi \Rightarrow \Phi \vee \Phi') \in \Gamma_k$ where φ is definitive, and after its

application, $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma, (\varphi \Rightarrow \Phi) \vee (\varphi \Rightarrow \Phi'))\}$. By assumption, $\mathcal{D}^k b^k w^k \models \varphi \Rightarrow \Phi \vee \Phi'$. That is, for all $w \in C$, $\mathcal{D}^k b^k w \not\models \varphi$ or $\mathcal{D}^k b^k w \models \Phi \vee \Phi'$. Let $w' \models \varphi$. Then, because φ is definitive, for all $w \in C$ such that $w \neq w'$, $\mathcal{D}^k b^k w \not\models \varphi$. And necessarily, $\mathcal{D}^k b^k w' \models \Phi \vee \Phi'$, that is, $\mathcal{D}^k b^k w' \models \Phi$ or $\mathcal{D}^k b^k w' \models \Phi'$. Hence, for all $w'' \in C$, $\mathcal{D}^k b^k w'' \not\models \varphi$ or $\mathcal{D}^k b^k w'' \models \Phi$, or for all $w'' \in C$, $\mathcal{D}^k b^k w'' \not\models \varphi$ or $\mathcal{D}^k b^k w'' \models \Phi'$. Therefore, $\mathcal{D}^k b^k w \models \varphi \Rightarrow \Phi$ or $\mathcal{D}^k b^k w \models \varphi \Rightarrow \Phi'$, which implies that $\mathcal{D}^k b^k w \models (\varphi \Rightarrow \Phi) \vee (\varphi \Rightarrow \Phi')$. Thus, $A(k')$.

- rule Ξ:
  For the rule to have been applied, $(\Sigma, [\![\alpha + \varsigma]\!]\Psi) \in \Gamma_k$. And if Γ_k contains (Σ', Ψ') such that $\Sigma' = \Sigma \xrightarrow{\alpha,\varsigma} e$, then $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma', \Psi)\}$, else $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma \xrightarrow{\alpha,\varsigma} e', \Psi)\}$, where e' is a fresh integer. By assumption, $\mathcal{D}^k b^k w^k \models [\![\alpha + \varsigma]\!]\Psi$. Hence, $P_{NB}(\alpha, \varsigma, b) \neq 0$ and $\mathcal{D}^k b' w^k \models \Psi,$ where $b' = BU(\alpha, \varsigma, b^k)$. Thus, $A(k')$.

- rule $\neg\Xi$:
  For the rule to have been applied, $(\Sigma, \neg[\![\alpha + \varsigma]\!]\Psi) \in \Gamma_k$, and after its application, $\Gamma_{k'} = \Gamma_k \cup \{(\Sigma, \neg Cont(\alpha, \varsigma) \vee [\![\alpha + \varsigma]\!]\neg\Psi)\}$. By assumption, $\mathcal{D}^k b^k w^k \models \neg[\![\alpha + \varsigma]\!]\Psi$. Then by definition, it is not the case that $P_{NB}(\alpha, \varsigma, b) \neq 0$ and $\mathcal{D}^k b' w^k \models \Psi,$ where $b' = BU(\alpha, \varsigma, b^k)$. That is, either $P_{NB}(\alpha, \varsigma, b^k) = 0$ or $\mathcal{D}^k b' w^k \models \neg\Psi,$ where $b' = BU(\alpha, \varsigma, b^k)$. That is, either $\mathcal{D}^k b^k w^k \models \neg Cont(\alpha, \varsigma)$ or $\mathcal{D}^k b^k w^k \models [\![\alpha + \varsigma]\!]\neg\Psi$. Hence, $\mathcal{D}^k b^k w^k \models \neg Cont(\alpha, \varsigma) \vee [\![\alpha + \varsigma]\!]\neg\Psi$. Thus, $A(k')$.

- rules $\neg\mathbf{B}$ and $\neg\mathbf{U}$:
  In these cases, the assumption of $A(k)$ leads directly to $A(k')$.

- the SI phase:
  $A(k)$ is assumed. And Γ_k is an open leaf node of a finished and saturated tree. Let $SI(\Gamma_k)$ be the system of inequalities generated from the formulae in Γ_k according to the instructions for the SI phase. We must show that $SI(\Gamma_k)$ is feasible; then Γ_k will remain a leaf node (without the labeled formula $(0, \bot)$), and $A(k')$ will be trivially true.

  It can be seen from § 7.2.2 that the generation of equations and inequalities from law literals in Γ_k are direct translations of the semantic definitions of the respective formulae. The directness of their translations is, in part, due to law literals being independent of activity sequences. Given that $A(k)$, the subsystem of inequalities generated from only the law literals must be feasible.

  It may however not be clear how inequalities generated from continuity, belief and utility literals in Γ_k affect the feasibility of the whole system $SI(\Gamma_k)$. We analyse this issue next. The inequalities generated from these three kinds of literals are also direct translations of their semantic definitions, except that the variables representing the probabilities of worlds at particular belief-states are carefully chosen: 'World-probability variable' ω_k^e represents the probability of world $w_k \in C^\#$ at activity-point e. An activity sequence

$$0 \xrightarrow{\alpha_0,\varsigma_0} e_1 \xrightarrow{\alpha_1,\varsigma_1} e_2 \cdots e_i \xrightarrow{\alpha_i,\varsigma_i} \cdots \xrightarrow{\alpha_{z-1},\varsigma_{z-1}} e_z$$

  represents the sequence of update operators in a sentence of the form

$$[\![\alpha_0, \varsigma_0]\!][\![\alpha_1, \varsigma_1]\!] \cdots [\![\alpha_i, \varsigma_i]\!] \cdots [\![\alpha_{z-1}, \varsigma_{z-1}]\!]\Psi,$$

where e_i is the activity-point representing the belief-state b^{e_i} reached after updating belief-state b^0 with α_0 and ς_0 to obtain b^{e_1}, then updating b^{e_1} with α_1 and ς_1 to obtain b^{e_2}, then updating b^{e_2} with ...to obtain b^{e_i}. The inequalities are generated in such a way that $\omega_k^{e_i}$ represents $b^{e_i}(w_k)$.

For instance, if $(\Sigma, \neg Cont(\alpha, \varsigma)), (\Sigma \xrightarrow{\alpha,\varsigma} e_z, \Psi) \in \Gamma_k$, then $SI(\Gamma_k)$ should be infeasible. This is because $(\Sigma \xrightarrow{\alpha,\varsigma} e_z, \Psi) \in \Gamma_k$ due to $(\Sigma, [\![\alpha + \varsigma]\!]\Psi) \in \Gamma_k$, which implies that the belief-state obtained after updating the previous belief-state with α and ς (i.e., the belief-state represented by e_z) is reachable, but $(\Sigma, \neg Cont(\alpha, \varsigma)) \in \Gamma_k$ implies that the belief-state represented by e_z is not reachable.

For $SI(\Gamma_k)$ to be infeasible due to this contradiction, requires that the equations generated from the two formulae both refer to the *same variables* $\omega_k^{e_z}$ for $k = 1, 2, \ldots, n$. Then

$$\sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,j}^{\alpha} \omega_i^{e_z} = 0 \tag{A.11}$$

is generated for $(\Sigma, \neg Cont(\alpha, \varsigma))$ and

$$\omega_1^{e_z} + \omega_2^{e_z} + \cdots + \omega_n^{e_z} = 1 \tag{A.12}$$

and

$$\omega_k^{e_z} = \frac{pr_k^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,k}^{\alpha} \omega_i^{e_{z-1}}}{\sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,j}^{\alpha} \omega_i^{e_{z-1}}}, \text{ for } k = 1, 2, \ldots, n \tag{A.13}$$

and

$$\sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,j}^{\alpha} \omega_i^{e_{z-1}} \neq 0$$

are generated for $(\Sigma \xrightarrow{\alpha,\varsigma} e_z, \Psi)$. By (A.12), there exists a variable $\omega_k^{e_z} > 0$. By (A.13), whenever $\omega_k^{e_z} > 0$, then $pr_k^{\varsigma|\alpha} > 0$ and for some $i \in \{1, 2, \ldots, n\}$, $pr_{i,k}^{\alpha} > 0$ and $\omega_i^{e_{z-1}} > 0$. There exists a term $pr_k^{\varsigma|\alpha} \times pr_{i,k}^{\alpha} \times \omega_k^{e_z}$ of (A.11) which is greater than zero, which is a contradiction, because then $\sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,j}^{\alpha} \omega_i^{e_z} \neq 0$.

During the generation of activity sequences in rule Ξ, fresh integers are used only when another labeled formula with the same sequence does not exist, else the existing activity-point/integer is used. Rule Ξ is repeated here for convenience: If Γ_k^j contains $(\Sigma, [\![\alpha + \varsigma]\!]\Psi)$ then: if Γ_k^j contains (Σ', Ψ') such that $\Sigma' = \Sigma \xrightarrow{\alpha,\varsigma} e$, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(\Sigma', \Psi)\}$, else create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(\Sigma \xrightarrow{\alpha,\varsigma} e', \Psi)\}$, where e' is a fresh integer. 'World-probability variables' are consistently named according to activity-points in the generation of $SI(\Gamma_k)$. In this way, as in the example above, a chain of constraints is set up such that if $SI(\Gamma_k)$ is infeasible, then the conjunction of formulae involved in the generation of $SI(\Gamma_k)$ is unsatisfiable. In other words, if the conjunction of formulae involved in the generation of $SI(\Gamma_k)$ is satisfiable, then $SI(\Gamma_k)$ is feasible.

In general, $SI(\Gamma_k)$ is infeasible if and only if there is no solution (assignment of values to variables) such that all equations and inequalities in the system are simultaneously true. Assume $SI(\Gamma_k)$ is infeasible. Then there exists at least one equation or inequality that

is false. All the kinds of equations and inequalities that can possibly appear in $SI(\Gamma_k)$ are considered in the list below. We now show for each case that if the corresponding equation/inequality is false, then the conjunction of formulae involved in the generation of $SI(\Gamma_k)$ is unsatisfiable. This would contradict the assumption that the conjunction of formulae involved in the generation of $SI(\Gamma_k)$ is satisfiable. That is, the primary assumption $A(k)$ would be false, which cannot be. Hence, the secondary assumption that $SI(\Gamma_k)$ is infeasible would have to be false. Therefore, $SI(\Gamma_k)$ would be feasible, which is what we want to show.

1. $c_1 pr^{\alpha}_{j,1} + c_2 pr^{\alpha}_{j,2} + \cdots + c_n pr^{\alpha}_{j,n} \bowtie q$ such that $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$—for a formula $(\Sigma, \phi \Rightarrow [\alpha]\varphi \bowtie q) \in \Gamma$, for every j such that $w_j \models \phi$. Simply, if $c_1 pr^{\alpha}_{j,1} + c_2 pr^{\alpha}_{j,2} + \cdots + c_n pr^{\alpha}_{j,n} \bowtie q$ is false, then by the definition of $[\alpha]\varphi \bowtie q$, $\phi \Rightarrow [\alpha]\varphi \bowtie q$ is unsatisfiable.

2. $c_1 pr^{\alpha}_{j,1} + c_2 pr^{\alpha}_{j,2} + \cdots + c_n pr^{\alpha}_{j,n} \not\bowtie q$ such that $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$—for a formula $(\Sigma, \phi \Rightarrow \neg[\alpha]\varphi \bowtie q) \in \Gamma$, for every j such that $w_j \models \phi$. Symmetrically to the case above, $\phi \Rightarrow \neg[\alpha]\varphi \bowtie q$ is unsatisfiable.

3. $pr^{\alpha}_{j,1} + pr^{\alpha}_{j,2} + \cdots + pr^{\alpha}_{j,n} = \lceil pr^{\alpha}_{j,1} + pr^{\alpha}_{j,2} + \cdots + pr^{\alpha}_{j,n} \rceil$—for a formulae $(\Sigma, \phi \Rightarrow [\alpha]\varphi \bowtie q) \in \Gamma$ or $(\Sigma, \phi \Rightarrow \neg[\alpha]\varphi \bowtie q) \in \Gamma$, for some j such that $w_j \models \phi$. Now, $pr^{\alpha}_{j,1} + pr^{\alpha}_{j,2} + \cdots + pr^{\alpha}_{j,n} = \lceil pr^{\alpha}_{j,1} + pr^{\alpha}_{j,2} + \cdots + pr^{\alpha}_{j,n} \rceil$ being false will ensure that neither $\sum_{w' \in W} R_\alpha(w_j, w') = 1$ nor $\sum_{w' \in W} R_\alpha(w_j, w') = 0$. However, as stated in Definition 7.1.2, either $\sum_{w' \in W} R_\alpha(w_j, w') = 1$ or $\sum_{w' \in W} R_\alpha(w_j, w') = 0$ for *every* $w_j \in C$. Thus, no SDL structure exists which can satisfy $\phi \Rightarrow [\alpha]\varphi \bowtie q$ or $\phi \Rightarrow \neg[\alpha]\varphi \bowtie q$.

4. $pr^{\varsigma|\alpha}_j \bowtie q$—for a formula $(\Sigma, \phi \Rightarrow (\varsigma|\alpha) \bowtie q) \in \Gamma$, for some j such that $w_j \models \phi$. If $pr^{\varsigma|\alpha}_j \bowtie q$ is false, then by definition of $(\varsigma|\alpha) \bowtie q$, $\phi \Rightarrow (\varsigma|\alpha) \bowtie q$ is unsatisfiable.

5. $pr^{\varsigma|\alpha}_j \not\bowtie q$—for a formula $(\Sigma, \phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q) \in \Gamma$, for some j such that $w_j \models \phi$. Symmetrically to the case above, $\phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q$ is unsatisfiable.

6. $pr^{\varsigma_1|\alpha}_j + pr^{\varsigma_2|\alpha}_j + \cdots + pr^{\varsigma_m|\alpha}_j = \lceil (pr^{\alpha}_{1,j} + pr^{\alpha}_{2,j} + \cdots + pr^{\alpha}_{n,j})/n \rceil$—for a formula $(\Sigma, \phi \Rightarrow (\varsigma|\alpha) \bowtie q) \in \Gamma$ or $(\Sigma, \phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q) \in \Gamma$, for some j such that $w_j \models \phi$. Now, $pr^{\varsigma_1|\alpha}_j + pr^{\varsigma_2|\alpha}_j + \cdots + pr^{\varsigma_m|\alpha}_j = \lceil (pr^{\alpha}_{1,j} + pr^{\alpha}_{2,j} + \cdots + pr^{\alpha}_{n,j})/n \rceil$ being false will ensure that whenever there exists a $w_i \in C$ such that $R_\alpha(w_i, w_j) > 0$, then $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) \neq 1$, and whenever there exists a $w_i \in C$ such that $R_\alpha(w_i, w_j) = 0$, then $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) \neq 0$. However, as stated in Definition 7.1.2, if there exists a $w_i \in C$ such that $R_\alpha(w_i, w_j) > 0$, then $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 1$, else $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 0$ for *every* $w_j \in C$. Thus, no SDL structure exists which can satisfy $\phi \Rightarrow (\varsigma|\alpha) \bowtie q$ or $\phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q$.

7. For every $(\Sigma, \Psi) \in \Gamma$, the following equations are in $SI(\Gamma)$.

$$\omega_k^{e_{h+1}} = BT(e_h, k, \alpha_h, \varsigma_h) \text{ (for } k = 1, 2, \ldots, n \text{ and } h = 0, 1, \ldots, z-1), \quad \text{(A.14)}$$

$$\Pi(e_h, \alpha_h, \varsigma_h) \neq 0 \text{ (for } h = 0, 1, \ldots, z-1) \text{ and} \quad \text{(A.15)}$$

$$\omega_1^{e_h} + \omega_2^{e_h} + \cdots + \omega_n^{e_h} = 1 \text{ (for } h = 0, 1, \ldots, z), \quad \text{(A.16)}$$

where Σ is $0 \xrightarrow{\alpha_0,\varsigma_0} e_1 \xrightarrow{\alpha_1,\varsigma_1} e_2 \cdots \xrightarrow{\alpha_{z-1},\varsigma_{z-1}} e_z$, $\Sigma \neq 0$ and e_0 is 0.

Note that if $(\Sigma, \Psi) \in \Gamma$, then $(0 \xrightarrow{\alpha_0,\varsigma_0} e_1 \xrightarrow{\alpha_1,\varsigma_1} e_2 \cdots \xrightarrow{\alpha_{i-1},\varsigma_{i-1}} e_i, [\![\alpha_i + \varsigma_i]\!]\Psi')$ must be in Γ, for some $0 < i < z$. And by the semantics, $\mathcal{D}bw \models [\![\alpha_i + \varsigma_i]\!]\Psi'$ if and only if

$$P_{NB}(\alpha_i, \varsigma_i, b) \neq 0 \text{ and } \mathcal{D}b'w \models \Psi', \text{ where } b' = BU(\alpha_i, \varsigma_i, b).$$

If any one of (A.14), (A.15) or (A.16) cannot be made true, then all of them cannot simultaneously be true. Now assume that $\omega_k^{e_{i+1}} \neq BT(e_i, k, \alpha_i, \varsigma_i)$ for some $k \in \{1, 2, \ldots, n\}$ and some $i \in \{0, 1, \ldots, z-1\}$, or $\Pi(e_i, \alpha_i, \varsigma_i) = 0$ for some $i \in \{0, 1, \ldots, z-1\}$, or $\omega_1^{e_{i+1}} + \omega_2^{e_{i+1}} + \cdots + \omega_n^{e_{i+1}} \neq 1$ for some $i \in \{0, 1, \ldots, z\}$. Then, respectively,

$$b^{i+1}(w_k) \neq b'(w_k), \text{ where } b' = BU(\alpha_i, \varsigma_i, b), \text{ i.e., } b^{i+1} \neq b',$$

or

$$P_{NB}(\alpha_i, \varsigma_i, b) = 0,$$

or

$$\sum_{j=1}^{n} b^{i+1}(w_j) \neq 1,$$

where $b^{i+1} = \{(w_1, \omega_1^{e_{i+1}}), (w_2, \omega_2^{e_{i+1}}), \ldots, (w_n, \omega_n^{e_{i+1}})\}$. In any case, $[\![\alpha_i + \varsigma_i]\!]\Psi'$ is unsatisfiable.

8. $\Pi(e, \alpha, \varsigma) \neq 0$—for a formula $(\Sigma e, Cont(\alpha, \varsigma)) \in \Gamma$. If $\Pi(e, \alpha, \varsigma) = 0$, then by definition of $Cont(\alpha, \varsigma)$, there exists no $\mathcal{D}$, $b = \{(w_1, \omega_1^e), (w_2, \omega_2^e), \ldots, (w_n, \omega_n^e)\}$ and $w \in C$ for which $\mathcal{D}bw \models Cont(\alpha, \varsigma)$.

9. $\Pi(e, \alpha, \varsigma) = 0$—for a formula $(\Sigma e, \neg Cont(\alpha, \varsigma)) \in \Gamma$. If $\Pi(e, \alpha, \varsigma) \neq 0$, then by definition of $Cont(\alpha, \varsigma)$, there exists no $\mathcal{D}$ such that $\Pi(e, \alpha, \varsigma) \neq 0$, $b = \{(w_1, \omega_1^e), (w_2, \omega_2^e), \ldots, (w_n, \omega_n^e)\}$ and $w \in C$ for which $\mathcal{D}bw \not\models Cont(\alpha, \varsigma)$.

10. $c_1\omega_1^e + c_2\omega_2^e + \cdots + c_n\omega_n^e \bowtie q$, where $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$—for a formula $(\Sigma e, \mathbf{B}\varphi \bowtie q) \in \Gamma$. $c_1\omega_1^e + c_2\omega_2^e + \cdots + c_n\omega_n^e \bowtie q$ is false iff $\sum_{w_k \in C,\, w_k \models \varphi} b(w_k) \not\bowtie q$, where $b = \{(w_1, \omega_1^e), (w_2, \omega_2^e), \ldots, (w_n, \omega_n^e)\}$ iff $\mathcal{D}bw \models \mathbf{B}\varphi \not\bowtie q$ iff $\mathbf{B}\varphi \bowtie q$ is unsatisfiable when $c_1\omega_1^e + c_2\omega_2^e + \cdots + c_n\omega_n^e \not\bowtie q$.

11. $R_j = r$—for a formula $(\Sigma, \phi \Rightarrow Reward(r)) \in \Gamma$, for some j such that $w_j \models \phi$. If $R_j = r$ must be false, then it is not the case that for all $w' \in C$, $\mathcal{D}bw' \not\models \phi$ or $\mathcal{D}bw' \models Reward(r)$. Therefore, in this case, $\phi \Rightarrow Reward(r)$ is unsatisfiable.

12. $R_j \neq r$—for a formula $(\Sigma, \phi \Rightarrow \neg Reward(r)) \in \Gamma$, for some j such that $w_j \models \phi$. If $R_j \neq r$ must be false, then it is not the case that for all $w' \in C$, $\mathcal{D}bw' \not\models \phi$ or $\mathcal{D}bw' \models \neg Reward(r)$. Therefore, in this case, $\phi \Rightarrow \neg Reward(r)$ is unsatisfiable.

13. $C_j^\alpha = r$—for a formula $(\Sigma, \phi \Rightarrow Cost(\alpha, r)) \in \Gamma$, for some j such that $w_j \models \phi$. If $C_j^\alpha = r$ must be false, then it is not the case that for all $w' \in C$, $\mathcal{D}bw' \not\models \phi$ or $\mathcal{D}bw' \models Cost(\alpha, r)$. Therefore, in this case, $\phi \Rightarrow Cost(\alpha, r)$ is unsatisfiable.

14. $C_j^\alpha \neq r$—for a formula $(\Sigma, \phi \Rightarrow \neg Cost(\alpha, r)) \in \Gamma$, for some j such that $w_j \models \phi$. If $C_j^\alpha \neq r$ must be false, then it is not the case that for all $w' \in C$, $\mathcal{D}bw' \not\models \phi$ or $\mathcal{D}bw' \models \neg Cost(\alpha, r)$. Therefore, in this case, $\phi \Rightarrow \neg Cost(\alpha, r)$ is unsatisfiable.

15. $\omega_1^e(R_1 - C_1^\alpha) + \omega_2^e(R_2 - C_2^\alpha) + \cdots + \omega_n^e(R_n - C_n^\alpha) \bowtie q$ (abbreviated as $RC(\alpha, e) \bowtie q$)—for a formula $(\Sigma e, \mathbf{U}[\![\alpha]\!] \bowtie q) \in \Gamma$. If $RC(\alpha, e) \bowtie q$ must be false, then by definition, $\mathbf{U}[\![\alpha]\!] \bowtie q$ is unsatisfiable at b, where $b = \{(w_1, \omega_1^e), (w_2, \omega_2^e), \ldots, (w_n, \omega_n^e)\}$.

16. $U([\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_y]\!], e_{z,\text{-}}) \bowtie q$—for a utility literal of the form $(\Sigma e_z, \mathbf{U}[\![\alpha_1]\!]\ [\![\alpha_2]\!] \cdots [\![\alpha_y]\!] \bowtie q)$ for $y \geq 2$. Let e be an arbitrary activity-point representing belief-state $b = \{(w_1, \omega_1^e), (w_2, \omega_2^e), \ldots, (w_n, \omega_n^e)\}$. Then by the generation of $SI(\Gamma)$ and by the definition of a structure $\mathcal{D}$,

$$RC(\alpha, b) = RC(\alpha, e), \tag{A.17}$$

$$P_{NB}(\alpha, \varsigma, b) = \Pi(e, \alpha, \varsigma) \text{ and} \tag{A.18}$$

$$b'(w_k) = s\omega_k^{e'} = BT(e, k, \alpha, \varsigma), \text{ where } b' = BU(\alpha, \varsigma, b). \tag{A.19}$$

Hence, due to equalities (A.17), (A.18) and (A.19),

$$\begin{aligned} &RC(\alpha_1, e_{z,\text{-}}) + \sum_{\varsigma_i \in \Omega^\#} \Pi(e_{z,\text{-}}, \alpha_1, \varsigma_i) U(\Lambda, e_{z+1,i}) = \\ &RC(\alpha_1, e_{z,\text{-}}) + \sum_{\substack{\varsigma_i \in \Omega^\# \\ U(\Lambda, e_{z+1,i}) = r}} \Pi(e_{z,\text{-}}, \alpha_1, \varsigma_i) \cdot r = \\ &RC(\alpha_1, b^z) + \sum_{\substack{\varsigma \in \Omega \\ b' = BU(\alpha_1, \varsigma, b) \\ \mathcal{D}b'w \models \mathbf{U}\Lambda = r}} P_{NB}(\alpha_1, \varsigma, b^z) \cdot r \bowtie q, \end{aligned} \tag{A.20}$$

where $b^z = \{(w_1, \omega_1^{e_z}), (w_2, \omega_2^{e_z}), \ldots, (w_n, \omega_n^{e_z})\}$ and Λ is $[\![\alpha_2]\!] \cdots [\![\alpha_y]\!]$. Therefore, if $U([\![\alpha_1]\!]\Lambda, e_{z,\text{-}}) \bowtie q$ must be false, $\mathcal{D}b^z w \models \mathbf{U}[\![\alpha_1]\!]\Lambda \bowtie q$ must be false (for all w; such that $\mathcal{D}$ satisfies equalities (A.17), (A.18) and (A.19)).

17. $\omega_1^{e_i} + \cdots + \omega_n^{e_i} = 1$—for some activity-point/node e_i in some utility tree. If $\omega_1^{e_i} + \cdots + \omega_n^{e_i} = 1$ must be false, then Inequality (A.20) is undefined, where $b^i = \{(w_1, \omega_1^{e_i}), (w_2, \omega_2^{e_i}), \ldots, (w_n, \omega_n^{e_i})\}$, $b^{i+1} = \{(w_1, \omega_1^{e_{i+1}}), (w_2, \omega_2^{e_{i+1}}), \ldots, (w_n, \omega_n^{e_{i+1}})\}$ and $b^{i+1} = BU(\alpha_i, \varsigma, b^i)$. Therefore,
$\mathbf{U}[\![\alpha_1]\!]\Lambda \bowtie q$ cannot be satisfied.

18. Assume

$$\Pi(e, \alpha, \varsigma) = 0 \;||\; \Pi(e, \alpha, \varsigma) \neq 0, \omega_1^{e'} = BT(e, 1, \alpha, \varsigma), \ldots, \omega_n^{e'} = BT(e, n, \alpha, \varsigma) \tag{A.21}$$

is in $SI(\Gamma)$ for some $e \xrightarrow{\alpha, \varsigma} e'$ in some utility tree. Recall that (A.21) is in $SI(\Gamma)$ due to $\mathbf{U}[\![\alpha_1]\!]\Lambda \bowtie q$ being in Γ.

Let $SI^-(\Gamma)$ be $SI(\Gamma)$ without (A.21). Recall that $SI(\Gamma)$ is feasible if and only if $SI^-(\Gamma) \cup \{\Pi(e, \alpha, \varsigma) = 0\}$ is feasible *or* $SI^-(\Gamma) \cup \{\Pi(e, \alpha, \varsigma) \neq 0, \omega_1^{e'} = BT(e, 1, \alpha, \varsigma)$,

$\ldots, \omega_n^{e'} = BT(e, n, \alpha, \varsigma)\}$ is feasible. Thus, if $SI(\Gamma)$ is infeasible due to (A.21), then (i) $SI^{-}(\Gamma) \cup \{\Pi(e, \alpha, \varsigma) = 0\}$ must be infeasible *and* (ii) $SI^{-}(\Gamma) \cup \{\Pi(e, \alpha, \varsigma) \neq 0$, $\omega_1^{e'} = BT(e, 1, \alpha, \varsigma)$, $\ldots$, $\omega_n^{e'} = BT(e, n, \alpha, \varsigma)\}$ must be infeasible. We assume $\Pi(e, \alpha, \varsigma) = 0$ must be false in case (i). Therefore, in case (ii), $\Pi(e, \alpha, \varsigma) \neq 0$ must be true and at least one of the equations $\omega_1^{e'} = BT(e, 1, \alpha, \varsigma), \ldots, \omega_n^{e'} = BT(e, n, \alpha, \varsigma)$ must be false.

Suppose it is $\omega_k^{e'} = BT(e, k, \alpha, \varsigma)$ which is false for some $k \in \{1, 2, \ldots, n\}$. Then $b'(w_k) \neq BT(e, k, \alpha, \varsigma)$, where $b' = \{(w_1, \omega_1^{e'}), \ldots, (w_k, \omega_k^{e'}), \ldots, (w_n, \omega_n^{e'})\}$. This implies that $b' \neq BU(\alpha, \varsigma, b)$, where $b = \{(w_1, \omega_1^{e}), (w_2, \omega_2^{e}), \ldots, (w_n, \omega_n^{e})\}$. But the semantic definition of $\mathbf{U}[\![\alpha_1]\!]\Lambda \bowtie r$ requires that $b' = BU(\alpha, \varsigma, b)$ is always true. Hence, $\mathbf{U}[\![\alpha_1]\!]\Lambda \bowtie r$ cannot be satisfied.

■

Theorem 7.3.1: (Soundness) If $\vdash \Psi$ then $\models \Psi$. (Contrapositively, if $\not\models \Psi$ then $\not\vdash \Psi$.)

Proof:
Let $\psi = \neg\Psi$. Then $\not\vdash \Psi$ if and only if the tree for ψ is open. And

$$\begin{aligned}
\not\models \Psi &\iff \text{not } (\forall \mathcal{D})\ \mathcal{D} \models \Psi \\
&\iff \text{not } (\forall \mathcal{D}, b)\ \mathcal{D}b \models \Psi \\
&\iff \text{not } (\forall \mathcal{D}, b, w)\ \mathcal{D}bw \models \Psi \\
&\iff (\exists \mathcal{D}, b, w)\ \mathcal{D}bw \models \psi.
\end{aligned}$$

For the soundness proof, it thus suffices to show that if there exists a structure $\mathcal{D}$, a belief-state b and a world w such that $\mathcal{D}bw \models \psi$, then the tree rooted at $\Gamma_0^0 = \{(0, \psi)\}$ is open. But this is a corollary of Lemma 7.3.1. ■

Lemma 7.3.2: $\mathcal{S}$ is an SDL structure.

Proof:
The components of the structure are well-formed:

- Due to Γ being open (and by the SI phase), we know that $SI(\Gamma)$ is feasible and hence, there exists a solution in $Z(\Gamma)$.

  By construction, R maps each action $\alpha \in \mathcal{A}$ to R_α such that R_α is a relation in $(C \times C) \times [0, 1]$. Moreover, by the nature of the SI generated from Γ, R_α is a (total) function $R_\alpha : (C \times C) \mapsto [0, 1]$.

  And by construction, the fact that

  $$pr_{j,1}^{\alpha} + pr_{j,2}^{\alpha} + \cdots + pr_{j,n}^{\alpha} = \lceil pr_{j,1}^{\alpha} + pr_{j,2}^{\alpha} + \cdots + pr_{j,n}^{\alpha} \rceil$$

  is an equation in any SI generated, means that either $\sum_{w' \in W} R_\alpha(w_j, w') = 1$ or $\sum_{w' \in W} R_\alpha(w_j, w') = 0$, for every $w_j \in C$.

- Due to Γ being open (and by the SI phase), we know that $SI(\Gamma)$ is feasible and hence, there exists a solution in $Z(\Gamma)$.

By construction, Q maps each action $\alpha \in \mathcal{A}$ to Q_α such that Q_α is a relation in $(C \times \Omega) \times [0,1]$.

Moreover, by the nature of the SI generated from Γ, Q_α is a (total) function $Q_\alpha : (C \times \Omega) \mapsto [0,1]$.

And by construction, the fact that in any SI generated, there is an equation

$$pr_j^{\varsigma_1|\alpha} + pr_j^{\varsigma_2|\alpha} + \cdots + pr_j^{\varsigma_m|\alpha} = \lceil (pr_{1,j}^{\alpha} + pr_{2,j}^{\alpha} + \cdots + pr_{n,j}^{\alpha})/n \rceil$$

for all $w_j \in C$, if there exists a $w_i \in C$ such that $R_\alpha(w_i, w_j) > 0$, then $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 1$, else $\sum_{\varsigma \in \Omega} Q_\alpha(w_j, \varsigma) = 0$.

- By construction, $U = \langle Re, Co \rangle$, where $Re : C \mapsto \mathbb{R}$ and Co is a mapping from $\mathcal{A}$ to a function $Co_\alpha : C \mapsto \mathbb{R}$, for each $\alpha \in \mathcal{A}$.

■

Lemma 7.3.3: Let Γ be an open leaf node of a finished tree. We know that $Z(\Gamma)$ is not empty. If $\mathcal{D}$ is constructed as described above, then for all $(\Sigma, \Psi) \in \Gamma$, there exists a b and a w such that $\mathcal{D}bw \models \Psi$.

Proof:
The proof will be by induction on the structure of a formula.

The induction step will work as follows. Let $\gamma' \subseteq \Gamma$ be added to Γ due to some rule applied to $\gamma \subseteq \Gamma$. Thus, we need to prove that IF for all $(\Sigma', \Psi') \in \gamma'$, there exists a b' and a w' such that $\mathcal{D}b'w' \models \Psi'$, THEN for all $(\Sigma, \Psi) \in \gamma$, there exists a b and a w such that $\mathcal{D}bw \models \Psi$.

We assume the antecedent (induction hypothesis).

Base case:

- Ψ is $c = c'$. Because $(\Sigma, \bot) \notin \Gamma$ for some Σ, rule $=$ was not applied. Hence, c is identical to c', and $\mathcal{D}bw \models c = c'$ (for all b and w).
- Ψ is $\neg(c = c')$. Because $(\Sigma, \bot) \notin \Gamma$ for some Σ, rule $=$ was not applied. Hence, c is not identical to c', and $\mathcal{S}, w \models \neg(c = c')$ (for all b and w).
- Ψ is $Cont(\alpha, \varsigma)$ (i.e., $(\Sigma e, Cont(\alpha, \varsigma)) \in \Gamma$). Due to the SI phase,

$$\Pi(e, \alpha, \varsigma, n) \neq 0,\ \omega_1^e + \omega_2^e + \cdots + \omega_n^e = 1 \in SI(\Gamma).$$

By construction, this implies that

$$\sum_{j=1}^{n} s_j^{\varsigma|\alpha} \sum_{i=1}^{n} s_{i,j}^{\alpha} s\omega_i^e \neq 0 \ \text{ and } \ s\omega_1^e + s\omega_2^e + \cdots + s\omega_n^e = 1.$$

iff

$$\sum_{j=1,\, w_j \in C^{\#}}^{n} Q_\alpha(\varsigma, w_j) \sum_{i=1,\, w_i \in C^{\#}}^{n} R_\alpha(w_i, w_j) b^e(w_i) \neq 0,$$

where

$$b^e = \{(w_1, s\omega_1^e), (w_2, s\omega_2^e), \ldots, (w_n, s\omega_n^e)\}$$

iff

$$P_{NB}(\alpha, \varsigma, b^e) \neq 0$$

iff

$$\mathcal{D}b^e w \models Cont(\alpha, \varsigma).$$

- Ψ is $\neg Cont(\alpha, \varsigma)$ (i.e., $(\Sigma e, \neg Cont(\alpha, \varsigma)) \in \Gamma$). Due to the SI phase,

$$\Pi(e, \alpha, \varsigma, n) = 0 \text{ and } s\omega_1^e + s\omega_2^e + \cdots + s\omega_n^e = 1 \in SI(\Gamma).$$

By construction, this implies that

$$\sum_{j=1}^{n} s_j^{\varsigma|\alpha} \sum_{i=1}^{n} s_{i,j}^{\alpha} s\omega_i^e = 0 \text{ and } s\omega_1^e + s\omega_2^e + \cdots + s\omega_n^e = 1.$$

iff

$$\sum_{j=1,\, w_j \in C^{\#}}^{n} Q_\alpha(\varsigma, w_j) \sum_{i=1,\, w_i \in C^{\#}}^{n} R_\alpha(w_i, w_j) b^e(w_i) = 0,$$

where

$$b^e = \{(w_1, s\omega_1^e), (w_2, s\omega_2^e), \ldots, (w_n, s\omega_n^e)\}$$

iff

$$P_{NB}(\alpha, \varsigma, b^e) = 0$$

iff

$$\mathcal{D}b^e w \models \neg Cont(\alpha, \varsigma).$$

- Ψ is $\mathbf{B}\varphi \bowtie q$ (i.e., $(\Sigma e, \mathbf{B}\varphi \bowtie q) \in \Gamma$). Due to the SI phase,

$$c_1\omega_1^e + c_2\omega_2^e + \cdots + c_n\omega_n^e \bowtie q,$$

and

$$\omega_1^e + \omega_2^e + \cdots + \omega_n^e = 1,$$

where $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$.

By construction, this implies that

$$\sum_{k=1,\, w_k \in C^{\#},\, w_k \models \varphi}^{n} b^e(w_k) \bowtie q,$$

where

$$b^e = \{(w_1, s\omega_1^e), (w_2, s\omega_2^e), \ldots, (w_n, s\omega_n^e)\}$$

iff

$$\mathcal{D}b^e w \models \mathbf{B}\varphi \bowtie q.$$

- Ψ is $\mathbf{U}\Lambda \bowtie q$ (i.e., $(\Sigma e_z, \mathbf{U}\Lambda \bowtie q) \in \Gamma$). When Λ is $[\![\alpha]\!]$, $RC(\alpha, e_z) \bowtie q \in SI(\Gamma)$. That is, by construction,

$$s\omega_1^{e_z}(sR_1 - sC_1^{\alpha}) + s\omega_2^{e_z}(sR_2 - sC_2^{\alpha}) + \cdots + s\omega_n^{e_z}(sR_n - sC_n^{\alpha}) \bowtie q,$$

where $s\omega_k^{e_z}, sR_k, sC_k^{\alpha} \in sln$, for $k = 1, 2, \ldots, n$. Let $b^z = \{(w_1, s\omega_1^{e_z}), (w_2, s\omega_2^{e_z}), \ldots, (w_n, s\omega_n^{e_z})\}$. Then,

$$\sum_{k=1, w_k \in C^{\#}}^{n} \big(Re(w_k) - Co_{\alpha}(w_k)\big) b^z(w_k) \bowtie q$$

iff

$$\mathcal{D}b^z w \models \mathbf{U}[\![\alpha]\!] \bowtie q.$$

Dealing with $(\Sigma e_z, \mathbf{U}[\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_y]\!] \bowtie q) \in \Gamma$, for $y \geq 2$ is more complicated: The inequality added to $SI(\Gamma)$ is

$$U([\![\alpha_1]\!][\![\alpha_2]\!] \cdots [\![\alpha_y]\!], e_{z,\text{-}}) \bowtie q, \text{ (cf. Eq. (7.3), § 7.2.2)}$$

where $e_{z,\text{-}} = e_z$. That is,

$$RC(\alpha_1, e_{z,\text{-}}) + \sum_{\varsigma_i \in \Omega^{\#}} \Pi(e_{z,\text{-}}, \alpha_1, \varsigma_i) U(\Lambda, e_{z+1,i}) \bowtie q,$$

where

$$U([\![\alpha_y]\!], e_{z+y-1,x}) = RC(\alpha_y, e_{z+y-1,x})$$

and Λ is $[\![\alpha_2]\!] \cdots [\![\alpha_y]\!]$.

Let e be an arbitrary activity-point representing belief-state $b = \{(w_1, s\omega_1^e), (w_2, s\omega_2^e), \ldots, (w_n, s\omega_n^e)\}$. Then by the generation of $SI(\Gamma)$ and by construction of $\mathcal{D}$,

$$RC(\alpha, b) = RC(\alpha, e), \tag{A.22}$$

$$P_{NB}(\alpha, \varsigma, b) = \Pi(e, \alpha, \varsigma) \text{ and} \tag{A.23}$$

$$b'(w_k) = s\omega_k^{e'} = BT(e, k, \alpha, \varsigma), \text{ where } b' = BU(\alpha, \varsigma, b). \tag{A.24}$$

Hence, due to equalities (A.22), (A.23) and (A.24),

$$\begin{aligned} & RC(\alpha_1, e_{z,\text{-}}) + \sum_{\varsigma_i \in \Omega^{\#}} \Pi(e_{z,\text{-}}, \alpha_1, \varsigma_i) U(\Lambda, e_{z+1,i}) \\ & = RC(\alpha_1, e_{z,\text{-}}) + \sum_{\substack{\varsigma_i \in \Omega^{\#} \\ U(\Lambda, e_{z+1,i}) = r}} \Pi(e_{z,\text{-}}, \alpha_1, \varsigma_i) \cdot r \\ & = RC(\alpha_1, b^z) + \sum_{\substack{\varsigma \in \Omega \\ b' = BU(\alpha_1, \varsigma, b) \\ \mathcal{D}b'w \models \mathbf{U}\Lambda = r}} P_{NB}(\alpha_1, \varsigma, b^z) \cdot r \\ & \bowtie q, \end{aligned}$$

where $b^z = \{(w_1, s\omega_1^{e_z}), (w_2, s\omega_2^{e_z}), \ldots, (w_n, s\omega_n^{e_z})\}$. Therefore,

$$\mathcal{D}b^z w \models \mathbf{U}[\![\alpha_1]\!]\Lambda \bowtie q \text{ (for all } w).$$

- Ψ is $\varphi \Rightarrow c = c'$. Because $(\Sigma, \bot) \notin \Gamma$ for some Σ, rule $=$ was not applied. Hence, c is identical to c', and $\mathcal{D}bw \models c = c'$ (for all b and w). Therefore, $\mathcal{D}bw \models \varphi \Rightarrow c = c'$ (for all b and w).
- Ψ is $\varphi \Rightarrow \neg(c = c')$. Because $(\Sigma, \bot) \notin \Gamma$ for some Σ, rule $=$ was not applied. Hence, c is not identical to c', and $\mathcal{S}, w \models \neg(c = c')$ (for all b and w). Therefore, $\mathcal{D}bw \models \varphi \Rightarrow \neg(c = c')$ (for all b and w).
- Ψ is $\varphi \Rightarrow Reward(r)$. By construction, $(w, r) \in Re$ where $w \models \varphi$. Hence, whenever $\mathcal{D}bw \models \varphi$, $\mathcal{D}bw \models Reward(r)$ (for all b). That is, $\mathcal{D}bw' \models \varphi \Rightarrow Reward(r)$ (for all b and w').
- Ψ is $\varphi \Rightarrow \neg Reward(r)$. By construction, $(w, r) \notin Re$ where $w \models \varphi$. Hence, whenever $\mathcal{D}bw \models \varphi$, $\mathcal{D}bw \models \neg Reward(r)$ (for every b). That is, $\mathcal{D}bw' \models \varphi \Rightarrow \neg Reward(r)$ (for all b and w').
- Ψ is $\varphi \Rightarrow Cost(\alpha, r)$. By construction, $(w, r) \in Co_\alpha$ where $w \models \varphi$. Hence, whenever $\mathcal{D}bw \models \varphi$, $\mathcal{D}bw \models Cost(\alpha, r)$ (for all b). That is, $\mathcal{D}bw' \models \varphi \Rightarrow Cost(\alpha, r)$ (for all b and w').
- Ψ is $\varphi \Rightarrow \neg Cost(\alpha, r)$. By construction, $(w, r) \notin Co_\alpha$ where $w \models \varphi$. Hence, whenever $\mathcal{D}bw \models \varphi$, $\mathcal{D}bw \models \neg Cost(\alpha, r)$ (for all b). That is, $\mathcal{D}bw' \models \varphi \Rightarrow \neg Cost(\alpha, r)$ (for all b and w').
- Ψ is $\phi \Rightarrow [\alpha]\varphi \bowtie q$. By construction,

  $$c_1 s_{j,1}^{\alpha} + c_2 s_{j,2}^{\alpha} + \cdots + c_n s_{j,n}^{\alpha} \bowtie q,$$

  where $w_j \models \phi$, and $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$, and the $s_{j,k}^{\alpha}$ are in $sln \in Z(\Gamma)$. Then as a direct consequence of the construction of $\mathcal{D}$, whenever $\mathcal{D}bw_j \models \phi$, $\mathcal{D}bw_j \models [\alpha]\varphi \bowtie q$ (for all b and $w_j \in C$). That is, $\mathcal{D}bw' \models \phi \Rightarrow [\alpha]\varphi \bowtie q$ (for all b and w').

- Ψ is $\phi \Rightarrow \neg[\alpha]\varphi \bowtie q$. By construction,

$$c_1 s^{\alpha}_{j,1} + c_2 s^{\alpha}_{j,2} + \cdots + c_n s^{\alpha}_{j,n} \not\bowtie q,$$

where $w_j \models \phi$, and $c_k = 1$ if $w_k \models \varphi$, else $c_k = 0$, and the $s^{\alpha}_{j,k}$ are in $sln \in Z(\Gamma)$. Then as a direct consequence of the construction of $\mathcal{D}$, whenever $\mathcal{D}bw_j \models \phi$, $\mathcal{D}bw_j \models \neg[\alpha]\varphi \bowtie q$ (for all b and $w_j \in C$). That is, $\mathcal{D}bw' \models \phi \Rightarrow \neg[\alpha]\varphi \bowtie q$ (for all b and w').

- Ψ is $\phi \Rightarrow (\varsigma|\alpha) \bowtie q$. By construction,

$$s^{\varsigma|\alpha}_j \bowtie q,$$

where $w_j \models \phi$ and $s^{\varsigma|\alpha}_j$ is in $sln \in Z(\Gamma)$. Then as a direct consequence of the construction of $\mathcal{D}$, whenever $\mathcal{D}bw_j \models \phi$, $\mathcal{D}bw_j \models (\varsigma|\alpha) \bowtie q$ (for all b and $w_j \in C$). That is, $\mathcal{D}bw' \models \phi \Rightarrow (\varsigma|\alpha) \bowtie q$ (for all b and w').

- Ψ is $\phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q$. By construction,

$$s^{\varsigma|\alpha}_j \not\bowtie q,$$

where $w_j \models \phi$ and $s^{\varsigma|\alpha}_j$ is in $sln \in Z(\Gamma)$. Then as a direct consequence of the construction of $\mathcal{D}$, whenever $\mathcal{D}bw_j \models \phi$, $\mathcal{D}bw_j \models \neg(\varsigma|\alpha) \bowtie q$ (for all b and $w_j \in C$). That is, $\mathcal{D}bw' \models \phi \Rightarrow \neg(\varsigma|\alpha) \bowtie q$ (for all b and w').

Induction step:

- Ψ contains a double negation $\neg\neg$. By rule $\neg$, $(\Sigma, \Psi') \in \Gamma$, where Ψ' is Ψ with the $\neg\neg$ removed. By induction hypothesis, $\mathcal{D}b'w' \models \Psi'$ for some b' and w'. By the definition of $\neg$, $\mathcal{D}b'w' \models \Psi$.

- Ψ is $\psi \wedge \psi'$. By rule $\wedge$, $(\Sigma, \psi), (\Sigma, \psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \psi$ and $\mathcal{D}b'w' \models \psi'$ for some b' and w'. By the definition of $\wedge$, $\mathcal{D}b'w' \models \psi \wedge \psi'$.

- Ψ is $\neg(\psi \wedge \psi')$. By rule $\vee$, $(\Sigma, \neg\psi) \in \Gamma$ or $(\Sigma, \neg\psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \neg\psi$ or $\mathcal{D}b'w' \models \neg\psi'$. By the definition of $\vee$, $\mathcal{D}b'w' \models \neg(\psi \wedge \psi')$.

- Ψ is $\psi \vee \psi'$. By rule $\vee$, $(\Sigma, \neg\psi) \in \Gamma$ or $(\Sigma, \neg\psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \neg\psi$ or $\mathcal{D}b'w' \models \neg\psi'$. By the definition of $\vee$, $\mathcal{D}b'w' \models \neg(\psi \wedge \psi')$.

- Ψ is $\neg\mathbf{B}\varphi \bowtie q$ (i.e., $(\Sigma, \neg\mathbf{B}\varphi \bowtie q) \in \Gamma$). By rule $\neg\mathbf{B}$, $(\Sigma, \mathbf{B}\varphi \not\bowtie q) \in \Gamma$.[2] By induction hypothesis, $\mathcal{D}b'w' \models \mathbf{B}\varphi \not\bowtie q$. Therefore, $\mathcal{D}b'w' \models \neg\mathbf{B}\varphi \bowtie q$.

- Ψ is $\neg\mathbf{U}\Lambda \bowtie q$ (i.e., $(\Sigma, \neg\mathbf{U}\Lambda \bowtie q) \in \Gamma$). (This case is similar to the case when Ψ is $\neg\mathbf{B}\varphi \bowtie q$.) By rule $\neg\mathbf{U}$, $(\Sigma, \mathbf{U}\Lambda \not\bowtie q) \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \mathbf{U}\Lambda \not\bowtie q$. Therefore, $\mathcal{D}b'w' \models \neg\mathbf{U}\Lambda \bowtie q$.

- Ψ is $\varphi \Rightarrow \Psi'$, where Ψ' is not a literal. Due to the preprocessing step, Ψ' is in CNF. $\varphi \Rightarrow \Psi'$ can be in one of four forms: (i) φ is not definitive and Ψ' has the form $\psi \wedge \psi'$, (ii) φ is not

[2] For conciseness, we abuse notation with $\not\bowtie$. In the case of $\bowtie$ being $=$, Γ contains either $(\Sigma, \mathbf{B}\varphi < q)$ or $(\Sigma, \mathbf{B}\varphi > q)$. By induction hypothesis, $\mathcal{D}b'w' \models \mathbf{B}\varphi < q$, respectively, $\mathcal{D}b'w' \models \mathbf{B}\varphi < q$, which implied $\mathcal{D}b'w' \not\models \mathbf{B}\varphi = q$.

definitive and Ψ' has the form $\psi \vee \psi'$, (iii) φ is definitive and Ψ' has the form $\psi \wedge \psi'$, (iv) φ is definitive and Ψ' has the form $\psi \vee \psi'$.

(i) By rule $\Rightarrow \wedge$, $(\Sigma, \varphi \Rightarrow \psi \wedge \varphi \Rightarrow \psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \varphi \Rightarrow \psi \wedge \varphi \Rightarrow \psi'$, which logically implies $\mathcal{D}b'w' \models \varphi \Rightarrow \psi \wedge \psi'$, which implies $\mathcal{D}b'w' \models \varphi \Rightarrow \Psi$.

(ii) By rule $\delta \Rightarrow$, $(\Sigma, (\delta_1 \Rightarrow \Phi') \wedge (\delta_2 \Rightarrow \Phi') \wedge \cdots \wedge (\delta_n \Rightarrow \Phi'))\}$, where $\delta_i \in cpt(\varphi)$. By induction hypothesis, $\mathcal{D}b'w' \models (\delta_1 \Rightarrow \Phi') \wedge (\delta_2 \Rightarrow \Phi') \wedge \cdots \wedge (\delta_n \Rightarrow \Phi')$, which logically implies $\mathcal{D}b'w' \models \varphi \Rightarrow \Psi'$.

(iii) By rule $\Rightarrow \wedge$, $(\Sigma, \varphi \Rightarrow \psi \wedge \varphi \Rightarrow \psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \varphi \Rightarrow \psi \wedge \varphi \Rightarrow \psi'$, which logically implies $\mathcal{D}b'w' \models \varphi \Rightarrow \psi \wedge \psi'$, which implies $\mathcal{D}b'w' \models \varphi \Rightarrow \Psi$.

(iv) By rule $\Rightarrow \vee$, $(\Sigma, \varphi \Rightarrow \psi \vee \varphi \Rightarrow \psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \varphi \Rightarrow \psi \vee \varphi \Rightarrow \psi'$, which logically implies $\mathcal{D}b'w' \models \varphi \Rightarrow \psi \vee \psi'$, which implies $\mathcal{D}b'w' \models \varphi \Rightarrow \Psi$.

- Ψ is $[\![\alpha + \varsigma]\!]\Psi'$ (i.e., $(\Sigma e, [\![\alpha + \varsigma]\!]\Psi') \in \Gamma$). By rule Ξ, $(\Sigma e \xrightarrow{\alpha,\varsigma} e', \Psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \Psi'$. By the SI phase, the following are in $SI(\Gamma)$.

$$\omega_k^{e'} = BT(e, k, \alpha, \varsigma, n) \text{ (for } k = 1, 2, \ldots, n),$$

$$\Pi(e, \alpha, \varsigma, n) \neq 0$$

$$\omega_1^e + \omega_2^e + \cdots + \omega_n^e = 1$$

and

$$\omega_1^{e'} + \omega_2^{e'} + \cdots + \omega_n^{e'} = 1.$$

Hence, by construction,

$$s\omega_k^{e'} = \frac{s_k^{\varsigma|\alpha} \sum_{i=1}^n s_{i,k}^{\alpha} s\omega_i^e}{\sum_{j=1}^n s_j^{\varsigma|\alpha} \sum_{i=1}^n s_{i,j}^{\alpha} s\omega_i^e} \text{ (for } k = 1, 2, \ldots, n), \tag{A.25}$$

$$\sum_{j=1}^{n} s_j^{\varsigma|\alpha} \sum_{i=1}^{n} s_{i,j}^{\alpha} s\omega_i^e \neq 0 \tag{A.26}$$

$$s\omega_1^e + s\omega_2^e + \cdots + s\omega_n^e = 1$$

and

$$s\omega_1^{e'} + s\omega_2^{e'} + \cdots + s\omega_n^{e'} = 1.$$

Let $b = \{(w_1, s\omega_1^e), (w_2, s\omega_2^e), \ldots, (w_n, s\omega_n^e)\}$ and let $b' = \{(w_1, s\omega_1^{e'}), (w_2, s\omega_2^{e'}), \ldots, (w_n, s\omega_n^{e'})\}$. Then by the set of Equations A.25, it must be that $b' = BU(\alpha, \varsigma, b)$. Further, Equation A.26 implies $P_{NB}(\alpha, \varsigma, b) \neq 0$. Therefore,

$$P_{NB}(\alpha, \varsigma, b) \neq 0 \text{ and } \mathcal{D}b'w' \models \Psi, \text{ where } b' = BU(\alpha, \varsigma, b)$$

iff

$$\mathcal{D}bw \models [\![\alpha + \varsigma]\!]\Psi \text{ (for all } w).$$

- Ψ is $\neg[\![\alpha+\varsigma]\!]\Psi'$. By rule $\neg\Xi$, $(\Sigma, \neg Cont(\alpha,\varsigma) \vee [\![\alpha+\varsigma]\!]\neg\Psi') \in \Gamma$. By induction hypothesis, $\mathcal{D}b'w' \models \neg Cont(\alpha,\varsigma) \vee [\![\alpha+\varsigma]\!]\neg\Psi'$, that is, $\mathcal{D}b'w' \models \neg Cont(\alpha,\varsigma)$ or $\mathcal{D}b'w' \models [\![\alpha+\varsigma]\!]\neg\Psi'$. Next, we show that both these cases lead to $\mathcal{D}b'w' \models \neg[\![\alpha + \varsigma]\!]\Psi'$.

  By definition,

  $$\mathcal{D}bw \models [\![\alpha + \varsigma]\!]\Psi \iff P_{NB}(\alpha,\varsigma,b) \neq 0 \text{ and } \mathcal{D}b''w \models \Psi, \text{ where } b'' = BU(\alpha,\varsigma,b).$$

  Hence, if $\mathcal{D}b'w' \models \neg Cont(\alpha,\varsigma)$, then $P_{NB}(\alpha,\varsigma,b') = 0$, implying $\mathcal{D}b'w' \not\models [\![\alpha + \varsigma]\!]\Psi'$. And $\mathcal{D}b'w' \models [\![\alpha + \varsigma]\!]\neg\Psi'$ implies $P_{NB}(\alpha,\varsigma,b') \neq 0$ and $\mathcal{D}b''w' \models \neg\Psi'$, where $b'' = BU(\alpha,\varsigma,b')$, which implies $\mathcal{D}b'w' \not\models [\![\alpha + \varsigma]\!]\Psi'$.

■

Theorem 7.3.2: (Completeness) If $\models \Psi$ then $\vdash \Psi$. (Contrapositively, if $\nvdash \Psi$ then $\not\models \Psi$.)

Proof:
Let $\psi = \neg\Psi$. Then $\nvdash \Psi$ means that there is an open leaf node of a finished tree for ψ. And

$$\begin{aligned}
\not\models \Psi &\iff (\exists\mathcal{D})\ \mathcal{D} \not\models \Psi \\
&\iff (\exists\mathcal{D}, b)\ \mathcal{D}b \not\models \Psi \\
&\iff (\exists\mathcal{D}, b, w)\ \mathcal{D}bw \not\models \Psi \\
&\iff (\exists\mathcal{D}, b, w)\ \mathcal{D}bw \models \psi.
\end{aligned}$$

For the completeness proof, it thus suffices to construct for some open leaf node of a finished tree for $\psi \in \mathcal{L}_{SDL}$, an SDL structure $\mathcal{D} = \langle R, Q, U\rangle$ such that there is a belief-state b and a world $w \in C$ such that ψ is satisfied in $\mathcal{D}$ at b at w.

By Lemmata 7.3.2 and 7.3.3, given an open leaf node Γ of a finished tree, there exists a structure $\mathcal{D}$, belief-state b and world w such that for all $(\Sigma, \Psi) \in \Gamma$, $\mathcal{D}bw \models \Psi$. But $(0, \psi) \in \Gamma$. Thus, if there is a finished open tableau for ψ, then ψ is satisfiable. The theorem follows directly. ■

On the decidability of feasibility of systems of inequalities (for SDL)

Lemma A.4.1: Determining whether an SI (as defined in this thesis) is feasible, is decidable.

Proof:
Tarski [1957] defines the first-order logic theory of elementary (real number) algebra as having an infinite number of variables (representing elements of $\mathbb{R}$), algebraic constants 1, 0, -1, two algebraic operation signs $+$ (addition) and $\cdot$ (multiplication), two algebraic relation symbols $=$ (equals) and $>$ (greater than), (logical) sentential connectives $\sim$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), the existential quantifier $\exists$, and a set of axioms defining the theory. "If ξ is any variable, then $(\exists\xi)$ is called a *quantifier expression*.[3] The expression $(\exists\xi)$ is to be read "there exists a ξ such that ."

[3] He actually uses the symbol E for existential quantification.

We show that every equation, disequation and inequality (or *(in)equality* for short) to be included in a system of inequlities as described in Section 7.2.2, can be represented in the language of first-order elementary algebra (FOEA).

First, assuming A and B are in the language of FOEA, note that any (in)equality of the form

$$A < B, A \leq B, A = B, A \geq B \text{ or } A > B$$

is true if and only if, respectively, the FOEA sentence

$$B > A, \sim (A > B), A = B, \sim (B > A) \text{ or } A > B$$

is true. And any (in)equality of the form

$$A \not\bowtie B$$

is true if and only if the FOEA sentence

$$\sim (A \bowtie' B)$$

is true, where $A \bowtie' B$ is the FOEA sentence corresponding to the (in)equality $A \bowtie B$. (In the rest of this section, we denote the FOEA sentence corresponding to the (in)equality $A \bowtie B$ as $A \bowtie' B$.)

Now it is easy to see that (in)equalities of the forms

$$\begin{aligned}
&c_1 pr_{j,1}^{\alpha} + c_2 pr_{j,2}^{\alpha} + \cdots + c_n pr_{j,n}^{\alpha} \bowtie q,\\
&c_1 pr_{j,1}^{\alpha} + c_2 pr_{j,2}^{\alpha} + \cdots + c_n pr_{j,n}^{\alpha} \not\bowtie q,\\
&pr_j^{\varsigma|\alpha} \bowtie q,\\
&pr_j^{\varsigma|\alpha} \not\bowtie q
\end{aligned}$$

have corresponding FOEA sentence representations

$$\begin{aligned}
&(\exists pr_{j,1}^{\alpha})(\exists pr_{j,2}^{\alpha}) \cdots (\exists pr_{j,n}^{\alpha}) c_1 \cdot pr_{j,1}^{\alpha} + c_2 \cdot pr_{j,2}^{\alpha} + \cdots + c_n \cdot pr_{j,n}^{\alpha} \bowtie' q,\\
&(\exists pr_{j,1}^{\alpha})(\exists pr_{j,2}^{\alpha}) \cdots (\exists pr_{j,n}^{\alpha}) \sim (c_1 \cdot pr_{j,1}^{\alpha} + c_2 \cdot pr_{j,2}^{\alpha} + \cdots + c_n \cdot pr_{j,n}^{\alpha} \bowtie' q),\\
&(\exists pr_j^{\varsigma|\alpha}) pr_j^{\varsigma|\alpha} \bowtie' q,\\
&(\exists pr_j^{\varsigma|\alpha}) \sim (pr_j^{\varsigma|\alpha} \bowtie' q),
\end{aligned}$$

where c_k is the constant 1 or 0, and the $pr_{j,k}^{\alpha}$ and $pr_j^{\varsigma|\alpha}$ are variables.

Equation

$$pr_{j,1}^{\alpha} + pr_{j,2}^{\alpha} + \cdots + pr_{j,n}^{\alpha} = \lceil pr_{j,1}^{\alpha} + pr_{j,2}^{\alpha} + \cdots + pr_{j,n}^{\alpha} \rceil$$

has the corresponding FOEA sentence representation

$$(\exists pr_{j,1}^{\alpha})(\exists pr_{j,2}^{\alpha}) \cdots (\exists pr_{j,n}^{\alpha})(pr_{j,1}^{\alpha} + pr_{j,2}^{\alpha} + \cdots + pr_{j,n}^{\alpha} = 1 \vee pr_{j,1}^{\alpha} + pr_{j,2}^{\alpha} + \cdots + pr_{j,n}^{\alpha} = 0)$$

and equation

$$pr_j^{\varsigma_1|\alpha} + pr_j^{\varsigma_2|\alpha} + \cdots + pr_j^{\varsigma_m|\alpha} = \lceil (pr_{1,j}^{\alpha} + pr_{2,j}^{\alpha} + \cdots + pr_{n,j}^{\alpha})/n \rceil$$

has the corresponding FOEA sentence representation

$$\begin{aligned}
&(\exists pr_{1,j}^{\alpha})(\exists pr_{2,j}^{\alpha}) \cdots (\exists pr_{n,j}^{\alpha})(\exists pr_j^{\varsigma_1|\alpha})(\exists pr_j^{\varsigma_2|\alpha}) \cdots (\exists pr_j^{\varsigma_m|\alpha}) \\
&(\sim (pr_{1,j}^{\alpha} + pr_{2,j}^{\alpha} + \cdots + pr_{n,j}^{\alpha} > 0) \vee pr_j^{\varsigma_1|\alpha} + pr_j^{\varsigma_2|\alpha} + \cdots + pr_j^{\varsigma_m|\alpha} = 1) \wedge \\
&(\sim (pr_{1,j}^{\alpha} + pr_{2,j}^{\alpha} + \cdots + pr_{n,j}^{\alpha} = 0) \vee pr_j^{\varsigma_1|\alpha} + pr_j^{\varsigma_2|\alpha} + \cdots + pr_j^{\varsigma_m|\alpha} = 0).
\end{aligned}$$

It can also be seen that any equation

$$\sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,j}^{\alpha} \omega_i^{e_h} = 0$$

has the corresponding FOEA sentence representation

$$\begin{aligned}
&(\exists pr_1^{\varsigma|\alpha})(\exists pr_2^{\varsigma|\alpha}) \cdots (\exists pr_n^{\varsigma|\alpha})(\exists pr_{1,1}^{\alpha})(\exists pr_{1,2}^{\alpha}) \cdots (\exists pr_{n,n}^{\alpha}) \\
&\sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \cdot (pr_{1,j}^{\alpha} \cdot \omega_1^{e_h} + pr_{2,j}^{\alpha} \cdot \omega_2^{e_h} + \cdots + pr_{n,j}^{\alpha} \cdot \omega_n^{e_h}) = 0,
\end{aligned}$$

where the summation symbol $\sum_{j=1}^{n}$ in the FOEA sentence is the obvious abbreviation.

And using the summation symbol to its full, it can also be seen that any equation

$$\omega_k^{e_{h+1}} = \frac{pr_k^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,k}^{\alpha} \omega_i^{e_h}}{\sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \sum_{i=1}^{n} pr_{i,j}^{\alpha} \omega_i^{e_h}}$$

has the corresponding FOEA sentence representation

$$\begin{aligned}
&(\exists pr_1^{\varsigma|\alpha})(\exists pr_2^{\varsigma|\alpha}) \cdots (\exists pr_n^{\varsigma|\alpha})(\exists pr_{1,1}^{\alpha})(\exists pr_{1,2}^{\alpha}) \cdots (\exists pr_{n,n}^{\alpha}) \\
&\omega_k^{e_{h+1}} \cdot \sum_{j=1}^{n} pr_j^{\varsigma|\alpha} \cdot \sum_{i=1}^{n} pr_{i,j}^{\alpha} \cdot \omega_i^{e_h} = pr_k^{\varsigma|\alpha} \cdot \sum_{i=1}^{n} pr_{i,k}^{\alpha} \cdot \omega_i^{e_h}
\end{aligned}$$

where $A \cdot \sum_{j=1}^{n} B_j$ means $A \cdot B_1 + \ldots + A \cdot B_n$.

For any (in)equality not given a FOEA sentence representation above, it should be easy for the reader to derive the FOEA sentence representation.

Tarski provided a finite method which can always decide whether a sentence in the elementary algebra is in the theory [Tarski, 1957]. Hence, feasibility of SIs is decidable. ■

BIBLIOGRAPHY

C. Amato, D. Bernstein, and S. Zilberstein. Solving POMDPs using quadratically constrained linear programs. In Ramon Lopez de Mantaras, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2418–2424, Menlo Park, CA, January 2007. AAAI Press.

K. Aström. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

F. Bacchus. *Representing and Reasoning with Uncertain Knowledge*. MIT Press, Cambridge, MA, 1990.

F. Bacchus, A. Grove, J. Halper, and D. Koller. Forming beliefs about a changing world. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94*, pages 222–229, 1994.

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

J. Barwise and J. Etchemendy. *The Language of First-Order Logic*. Center for the Study of Language and Information, Stanford, USA, 1992.

R. Bellman. A markov decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.

M. Ben-Ari. *Mathematical Logic for Computer Science*. Springer Verlag, London, 2nd edition, 2001.

M. Ben-Ari. *Mathematical Logic for Computer Science*. Springer Verlag, London Berlin Heidelberg, 3rd edition, 2012.

M. Ben-Or, D. Kozen, and J.Reif. The complexity of elementary algebra and geometry. *Journal of Computr and Systems Sciences*, 32(2):251–264, 1986.

P. Blackburn, M. De Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, UK, 2001.

P. Blackburn, J. Van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, Amsterdam, The Netherlands / Oxford, UK, 2007.

W. Bledsoe. A new method for proving certain Presburger formulas. In *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, volume 1 of *IJCAI'75*, pages 15–21,

San Francisco, CA, USA, 1975. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1624626.1624629.

C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1168–1175, Menlo Park, CA, 1996. AAAI Press.

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research (JAIR)*, 11:1–94, 1999.

C. Boutilier, R. Reiter, M. Soutchanski, and S. Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00) and of the Twelfth Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 355–362. AAAI Press, Menlo Park, CA, 2000. ISBN 0-262-51112-6.

R. Brachman and H. Levesque. *Knowledge representation and reasoning*. Morgan Kaufmann, San Fransisco, CA, 2nd edition, 2004.

G. Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense (Cambridge Tracts in Theoretical Computer Science)*. Cambridge University Press, reissue edition, 2012.

A. Cassandra, L. Kaelbling, and M. Littman. Acting optimally in partially observable stochastic domains. Technical Report CS-94-20, Brown Universiteit Leuven, Department of Computer Science, Providence, Rhode Island 02912, 1994.

M. Castilho, O. Gasquet, and A. Herzig. Formalizing action and change in modal logic I: The frame problem. *Journal of Logic and Computation*, 9(5):701–735, 1999.

A. Chagrov and M. Zakharyaschev. *Modal Logic (Oxford Logic Guides, Vol. 35)*. Oxford University Press, Oxford, England, 1997.

B. Chellas. *Modal Logic: an introduction*. Cambridge University Press, Cambridge, MA, 1980.

G. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *The Second GI Conference on Automata Theory and Formal Languages*, pages 515–532, 1975.

G. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963 & 1998.

A. Darwiche. Bayesian Networks. In B. Porter F. Van Harmelen, V. Lifshitz, editor, *The Handbook of Knowledge Representation*, pages 467–510. Elsevier Science, 2008.

G. De Giacomo and M. Lenzerini. PDL-based framework for reasoning about actions. In *Topics in Artificial Intelligence, (Proceedings of IA*AI'95)*, volume 992 of *Lecture Notes in Computer Science*, pages 103–114. Springer Verlag, 1996.

M. De Weerdt, F. De Boer, W. Van der Hoek, and J.-J. Meyer. Imprecise observations of mobile robots specified by a modal logic. In *Proceedings of the Fifth Annual Conference of the Advanced School for Computing and Imaging (ASCI-99)*, pages 184–190, 1999.

L. Dines. Systems of linear inequalities. *The Annals of Mathematics*, 20(3):191–199, 1919. ISSN 0003486X.

R. Fagin and J. Halpern. Reasoning about knowledge and probability. *Journal of the ACM*, 41(2): 340–367, 1994.

R. Fagin, J. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.

J. Ferrante and C. Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM Journal of Computation*, 4(1):69–76, 1975. doi: http://dx.doi.org/10.1137/0204006.

D. Fierens, H. Blockeel, M. Bruynooghe, and J. Ramon. Logical bayesian networks and their relation to other probabilistic logical models. In S. Kramer and B. Pfahringer, editors, *ILP 2005*, volume 3625 of *LNAI*, pages 121–135, Heidelberg/Berlin, 2005. Springer Verlag.

M. Fitting. Introduction. In M. D' Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 1–44. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

A. Gabaldon and G. Lakemeyer. $\mathcal{ESP}$: A logic of only-knowing, noisy sensing and acting. In *Proceedings of the Twenty-second National Conference on Artificial Intelligence (AAAI-07)*, pages 974–979. AAAI Press, 2007.

P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, Massachusetts/England, 1988.

S. Gass. *Linear programming : methods and applications*. Dover Publications, fifth edition, 2010.

H. Geffner and B. Bonet. Solving large POMDPs using real time dynamic programming. In *In Working Notes of AAAI Fall Symposium on POMDPs*, 1998.

H. Geffner and J. Wainer. Modeling action, knowledge and control. In *Proceedings of the European Conference on Artificial Intelligence (ECAI-98)*, pages 532–536, 1998.

R. Goré. Tableau methods for modal and temporal logics. In M. D' Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

A. Grove, J. Halpern, and D. Koller. Random worlds and maximum entropy. *Journal of Artificial Intelligence Research (JAIR)*, 2:33–88, 1994.

J. Halpern. *Reasoning about Uncertainty*. The MIT Press, Cambridge, MA, 2003.

D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

J. Hintikka. *Knowledge and belief*. Cornell University Press, Ithaca, NY, 2nd edition, 1962.

R. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

Z. Huang, M. Masuch, and L. Pólos. ALX, an action logic for agents with bounded rationality. *Artificial Intelligence*, 82:75–127, 1996.

G. Hughes and M. Cresswell. *A New Introduction to Modal Logic*. Routledge, New York, NY, 1996.

L. Iocchi, T. Lukasiewicz, D. Nardi, and R. Rosati. Reasoning about actions with sensing under qualitative and probabilistic uncertainty. *ACM Transactions on Computational Logic*, 10(1): 5:1–5:41, 2009.

M. Jaeger. A logic for default reasoning about probabilities. In *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence*, UAI'94, pages 352–359, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-332-8. URL http://dl.acm.org/citation.cfm?id=2074394.2074439.

E. Jaynes. Where do we stand on maximum entropy? In *The Maximum Entropy Formalism*, pages 15–118. MIT Press, 1978.

L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

T. Käufl. Reasoning about systems of linear inequalities. In E. Lusk and R. Overbeek, editors, *Proceedings of the Ninth International Conference on Automated Deduction*, volume 310 of *LNCS*, pages 563–572. Springer Verlag, 1988. ISBN 3-540-19343-X.

G. Kern-Isberner, editor. *Conditionals in Nonmonotonic Reasoning and Belief Revision: Considering Conditionals as Agents*. Number 2087 in Lecture Notes in Computer Science. Springer Verlag, Berlin/Heidelberg, 2001. ISBN 3-540-42367-2.

S. Koenig. Agent-centered search. *Artificial Intelligence Magazine*, 22:109–131, 2001. ISSN 0738-4602. URL http://dl.acm.org/citation.cfm?id=567363.567371.

B. Kooi. Probabilistic dynamic epistemic logic. *Journal of Logic, Language and Information*, 12 (4):381–408, 2003.

R. Kowalski. *Computational Logic and Human Thinking*. Cambridge University Press, The Edinburgh Building, Cambridge, UK, 2011.

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

S. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24(1):1–14, 1959.

D. Kroening and O. Strichman. *Decision Procedures: An Algorithmic Point of View*. Texts in Theoretical Computer Science. Springer Verlag, 2008. ISBN 978-3-540-74104-6.

G. Lakemeyer and H. Levesque. A semantic characterization of a useful fragment of the situation calculus with knowledge. In *Special issue in honor of John McCarthy, Artificial Intelligence*. Elsevier, 2010.

H. Levesque and G. Lakemeyer. Situations, si! Situation terms no! In *Proceedings of the Conference on Principles of Knowledge Representation and Reasoning (KR-04)*, pages 516–526. AAAI Press, 2004.

H. Levesque and G. Lakemeyer. Cognitive Robotics. In B. Porter F. Van Harmelen, V. Lifshitz, editor, *The Handbook of Knowledge Representation*, pages 869–886. Elsevier Science, 2008.

H. Levesque and F. Pirri, editors. *Logical Foundations for Cognitive Agents: Contributions in Honor of Ray Reiter*. Springer Verlag, 1999.

H. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1–3):59–84, 1997.

W. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28:47–66, 1991.

J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963.

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

J.-J. Meyer. Dynamic logic for reasoning about actions and agents. In Jack Minker, editor, *Logic-based artificial intelligence*, pages 281–311. Kluwer Academic Publishers, Norwell, MA, USA, 2000. ISBN 0-7923-7224-7. URL `http://dl.acm.org/citation.cfm?id=566344.566365`.

G. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

T. Motzkin. *Beiträge zur Theorie der linearen Ungleichungen*. PhD thesis, Universitat Zurich, 1936.

K. Murty. *Linear programming*. John Wiley and sons, revised edition, 1983.

N. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

J. Pineau. *Tractable Planning Under Uncertainty: Exploiting Structure*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, August 2003.

D. Poole. Decision theory, the situation calculus and conditional plans. *Linköping Electronic Articles in Computer and Information Science*, 8(3), 1998.

D. Poole. The independent choice logic and beyond. In L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors, *Probabilistic Inductive Logic Programming: Theory and Application*, volume 4911 of *Lecture Notes in Artificial Intelligence*, pages 222–243. Springer Verlag, 2008.

D. Poole and A. Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, New York, USA, 2010.

S. Popkorn. *First Steps in Modal Logic*. Cambridge University Press, 1994.

H. Prendinger and G. Schurz. Reasoning about action and change: A dynamic logic approach. *Journal of Logic, Language and Information*, 5(2):209–245, 1996.

M. Puterman. *Markov Decision Processes: Discrete Dynamic Programming*. Wiley, New York, NY, 1994.

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*, pages 359–380. Academic Press Professional, Inc., San Diego, CA, USA, 1991.

R. Reiter. *Knowledge in action: logical foundations for specifying and implementing dynamical systems*. MIT Press, Massachusetts/England, 2001.

G. Rens. A belief-desire-intention architecture with a logic-based planner for agents in stochastic domains. Master's thesis, School of Computing, University of South Africa, 2010.

G. Rens, I. Varzinczak, T. Meyer, and A. Ferrein. A logic for reasoning about actions and explicit observations. In Jiuyong Li, editor, *AI 2010: Advances in Artificial Intelligence: Proceedings of the Twenty-third Australasian Joint Conference*, volume 6464 of *LNAI*, pages 395–404, Berlin/Heidelberg, December 2010. Springer Verlag. ISBN 978-3-642-17431-5.

G. Rens, G. Lakemeyer, and T. Meyer. A logic for specifying agent actions and observations with probability. In K. Kersting and M. Toussaint, editors, *Proceedings of the Sixth Starting AI Researchers' Symposium (STAIRS 2012)*, volume 241 of *Frontiers in Artificial Intelligence and Applications*, pages 252–263. IOS Press, 2012. url: http://www.booksonline.iospress.nl/Content/View.aspx?piid=31509.

G. Rens, T. Meyer, and G. Lakemeyer. On the logical specification of probabilistic transition models. In *Proceedings of the Eleventh International Symposium on Logical Formalizations of Commonsense Reasoning (COMMONSENSE 2013)*, University of Technology, Sydney, May 2013. UTSe Press. URL `http://www.commonsense2013.cs.ucy.ac.cy/docs/commonsense2013\_submission\_9.pdf`.

G. Rens, T. Meyer, and G. Lakemeyer. SLAP: Specification logic of actions with probability. *Journal of Applied Logic*, 12(2):128–150, 2014a. ISSN 1570-8683.

G. Rens, T. Meyer, and G. Lakemeyer. A logic for specifying stochastic actions and observations. In C. Beierle and C. Meghini, editors, *Proceedings of the Eighth International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, Lecture Notes in Computer Science, pages 305–323. Springer-Verlag, 2014b.

S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 32:663–704, 2008. doi: http://dx.doi.org/10.1613/jair.2567.

S. Russell and P. Norvig. *Artificial intelligence: A Modern Approach*. Prentice Hall, New Jersey, 2nd edition, 2003.

J. Sack. Extending probabilistic dynamic epistemic logic. *Synthese*, 169:124–257, 2009.

S. Sanner and K. Kersting. Symbolic dynamic programming for first-order POMDPs. In *Proceedings of the Twenty-fourth National Conference on Artificial Intelligence (AAAI-10)*, pages 1140–1146. AAAI Press, 2010.

A. Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, 60:365–374, 1954.

A. Shirazi and E. Amir. Probabilistic modal logic. In *Proceedings of the Twenty-second National Conference on Artificial Intelligence (AAAI-07)*, pages 489–494. AAAI Press, 2007.

R. Shostak. On the SUP-INF method for proving Presburger formulas. *Journal of the ACM*, 24 (4):529–543, October 1977. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/322033.322034.

R. Smallwood and E. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

T. Smith and R. Simmons. Point-based POMDP algorithms: Improved analysis and implementation. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 542–549. AUAI Press, 2005.

A. Tarski. A decision method for elementary algebra and geometry. Technical report, The RAND Corporation, Santa Monica, Calif., 1957.

S. Thiébaux, J. Hertzberg, W. Schoaff, and M. Schneider. A stochastic model of actions and plans for anytime planning under uncertainty. *International Journal of Intelligent Systems*, 10(2): 155–183, 1995.

M. Thielscher. From situation calculus to fluent calculus: State update axioms as a solution to the inferential frame problem. *Artificial Intelligence*, 111(1–2):277–299, 1999.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Massachusetts/England, 2005.

J. Van Benthem, J. Gerbrandy, and B. Kooi. Dynamic update with probabilities. *Studia Logica*, 93(1):67–96, 2009.

J. Van Diggelen. Using modal logic in mobile robots. Master's thesis, Cognitive Artificial Intelligence, Utrecht University, 2002.

H. Van Ditmarsch, W. Van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*. Springer Verlag, Dordrecht, The Netherlands, 2007.

Y. Virin, G. Shani, S. Shimony, and R. Brafman. Scaling up: Solving pomdps through value based clustering. In *Proceedings of the Twenty-second AAAI Conference on Artificial Intelligence (AAAI-07)*, pages 1290–1295, Menlo Park, CA, July 2007. AAAI Press.

C. Wang and J. Schmolze. Planning with POMDPs using a compact, logic-based representation. In *Proceedings of the Seventeenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI-05)*, pages 523–530, Los Alamitos, CA, USA, 2005. IEEE Computer Society.