SLAP: Specification Logic of Actions with Probability

Gavin Rens[a], Thomas Meyer[a], Gerhard Lakemeyer[b]

[a] *Centre for Artificial Intelligence Research, University of KwaZulu-Natal, and CSIR Meraka, South Africa*
[b] *RWTH Aachen University, Germany*

Abstract

A logic for specifying probabilistic transition systems is presented. Our perspective is that of agents performing actions. A procedure for deciding whether sentences in this logic are valid is provided. One of the main contributions of the paper is the formulation of the decision procedure: a tableau system which appeals to solving systems of linear equations. The tableau rules eliminate propositional connectives, then, for all open branches of the tableau tree, systems of linear equations are generated and checked for feasibility. Proofs of soundness, completeness and termination of the decision procedure are provided.

Keywords: probabilistic actions, modal logic, tableau method, systems of linear inequalities

1. Introduction

In this article, 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.

Modal logic [1, 2, 3, 4] is considered to be well suited to reasoning about beliefs and changing situations and MDP theory [5, 6, 7] has proven to be a good general framework for formalizing dynamic stochastic systems.

Email addresses: grens@csir.co.za (Gavin Rens), tmeyer@csir.co.za (Thomas Meyer), gerhard@cs.rwth-aachen.de (Gerhard Lakemeyer)

Next we introduce a scenario to illustrate concepts throughout the article. Imagine a robot that is in need of an oil refill. 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 'weight' sensor. And once holding the can, the robot may wish to replace it on the floor.

The domain is (partially) formalized as follows (one cannot model the (epistemic) effects of observations with SLAP). The robot has the set of (intended) actions $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{weigh}, \texttt{replace}\}$ with expected meanings. The robot experiences its world (domain) through three Boolean features: $\mathcal{P} = \{\texttt{full}$, $\texttt{drank}$, $\texttt{holding}\}$ 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 BK 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 $(\texttt{full} \land \neg\texttt{drank} \land \texttt{holding}) \rightarrow [\texttt{drink}]_{0.15}(\texttt{full} \land \neg\texttt{drank} \land \texttt{holding})$ follow from BK?

- 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 $(\neg\texttt{full} \land \neg\texttt{holding}) \rightarrow ([\texttt{grab}]_{0.9}(\neg\texttt{full} \land \texttt{holding}) \land [\texttt{grab}]_{0.1}(\neg\texttt{full} \land \neg\texttt{holding}))$ follow from BK?

In another paper [8], we propose how to write sentences in the language of SLAP to capture a model of probabilistic transitions due to the execution of actions of some agent. And in that paper, we suggest which assumptions can and perhaps should be made about such specifications to make them more parsimonious. In the context of SLAP, we are interested in three things in the domain of interest: (i) The initial condition IC, that is, a specification of the world the agent finds itself in when it becomes active. (ii) Domain constraints or static laws SL, that is, facts and laws about the domain that do not change. (iii) Information about when actions are possible and impossible, the effects of actions and conditions for the effects—the dynamics of the environment or system. Refer to these as the *action description* (AD). How to write these axioms is the focus of that paper [8].

Let the union of all the axioms in SL and AD be denoted by the set BK—the agent's *background knowledge*. IC is not part of the agent's background knowledge.

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

In Section 2.2, we define *entailment*, which depends on the notion of *validity*. We must defer a discussion of the use of SLAP to Section 2.3, after the syntax and semantics have been presented. The focus of this article, though, is on a decision procedure for entailment of SLAP sentences from sets of SLAP sentences, and on the computational property of the decision procedure.

Section 2 defines SLAP. Section 3 provides a decision procedure for determining entailment of sentences in SLAP. In Section 4, we prove that the procedure is sound, complete and that it terminates, that is, we show that SLAP is decidable with respect to entailment. Sections 5 and 6 cover some related work, and respectively, summarizes what has been achieved in this article, and discusses future work.

2. Specification Logic of Actions with Probability

First we present the syntax of SLAP, then we state its semantics.

2.1. Syntax

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

1. a finite set of *propositional variables* (alias, *fluents*) $\mathcal{P} = \{p_1, \ldots, p_n\}$,
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 2.1. Let $\alpha \in \mathcal{A}$, $q \in \mathbb{Q}_{[0,1]}$ and $p \in \mathcal{P}$. The language of SLAP, denoted $\mathcal{L}_{SLAP}$, is the least set of Ψ defined by the grammar[1]:

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

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'. $[\alpha]$ 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 2.2. 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*.

[1] In [8], we erroneously omitted $\Psi \wedge \Psi$ from the definition of Ψ.

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

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

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 $\varphi \in \mathcal{L}_{SLAP}$.

2.2. Semantics

SLAP structures are derived from Markov decision processes (MDPs) [5, 6, 7]. 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.

Standard 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. This is the standard Kripke-style semantics (see, e.g., [9, 10, 1]).

SLAP structures are non-standard: 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 $w : \mathcal{P} \mapsto \{0, 1\}$ 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 2.3. 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*.

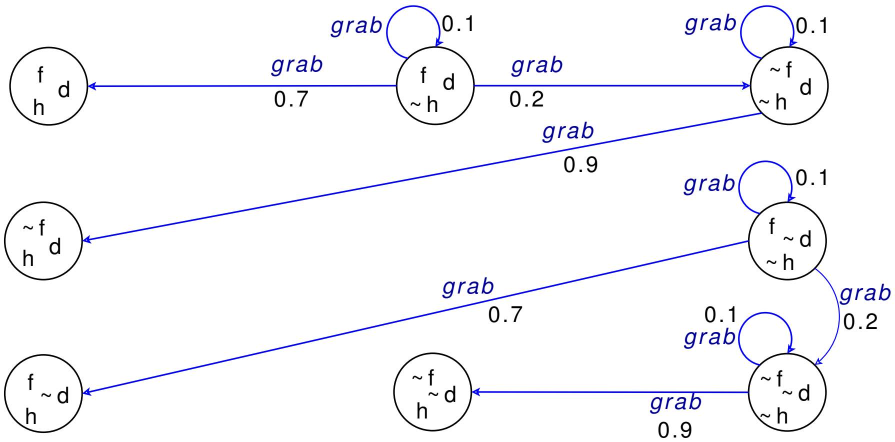


Figure 1: A transition diagram for the `grab` action. The letters f, d and h, respectively represent propositional literals `full`, `drank` and `holding`. And $\sim$ reads 'not'.

2. $R : \mathcal{A} \mapsto R_\alpha$, 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 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 2.4 (Truth Conditions). Let $\mathcal{S}$ be a SLAP structure, with $\alpha, \alpha' \in \mathcal{A}$ and $q, pr \in \mathbb{Q}_{[0,1]}$. Let $p \in \mathcal{P}$ 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 holds:

1. $\mathcal{S}, w \models \top$ for all $w \in W$;
2. $\mathcal{S}, w \models p \iff w(p) = 1$ 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. $\mathcal{S}, w \models [\alpha]_q\varphi \iff \big(\sum_{(w,w',pr)\in R_\alpha, \mathcal{S},w'\models\varphi} pr\big) = q$;
6. $\mathcal{S}, w \models \Box\Psi \iff$ for all $w' \in W$, $\mathcal{S}, w' \models \Psi$.

Looking at Figure 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 a 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 $\mathcal{K} \subset \mathcal{L}_{SLAP}$ 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 $\theta \in \mathcal{K}$, $\mathcal{S}, w \models \theta$, then $\mathcal{S}, w \models \Phi$. We also say that $\mathcal{K}$ *entails* Φ whenever $\mathcal{K} \models \Phi$.

Proposition 2.1. *Recall that $\mathcal{L}^{-\Box}_{SLAP}$ is all formulae in $\mathcal{L}_{SLAP}$ such that the formulae contain no $\Box$ operators. 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 straight-forward, appealing to basic logical equivalences.

2.3. 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 2.2. $\mathcal{K} \models \Phi \iff \bigwedge_{\theta\in\mathcal{K}} \theta \wedge \neg\Phi$ *is unsatisfiable.*

The proof is straightforward.

Note that $\neg\Box\Phi' \notin \mathcal{L}_{SLAP}$. This is why Φ 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 2.2 (with restricted arguments).

In the introduction, we mentioned that we are interested in whether $IC \rightarrow \Phi$ follows from $\bigwedge_{\phi \in BK} \Box\phi$, in general. Formally, this is stated as

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

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. For instance, a complete specifications of the probabilistic effects of action `grab` (Fig. 1) is

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

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

In terms of the decision procedure, the question of whether

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

is posed as the question of whether

$$\bigwedge_{\phi \in BK} \Box\phi \land \neg(IC \rightarrow \Phi)$$

is (un)satisfiable.

In the soundness and completeness proofs (§ 4.1 and § 4.2), even though formulae of the form $\neg\Phi$ are analysed, $\neg\Phi$ always has the form $\bigwedge_{\theta\in\mathcal{K}}\theta \wedge \neg\Phi'$, where Φ' does not mention the $\Box$ operator.

3. Decision Procedure for SLAP Entailment

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, Gasquet and Herzig [11]. 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.

3.1. The Tableau Phase

The necessary definitions and terminology are given next.

Definition 3.1. A *labeled formula* is a pair (x, Ψ), where $\Psi \in \mathcal{L}_{SLAP}$ is a formula and x is an integer called the *label* of Ψ.

Definition 3.2. A *node* Γ^j_k with superscript j (the *branch* index) and subscript k (the *node* index), is a set of labeled formulae.

Definition 3.3. The initial node, that is, Γ^0_0, to which the tableau rules must be applied, is called the *trunk*.

Definition 3.4. 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. If one has a tree with trunk $\Gamma^0_0 = \{(0, \Psi)\}$, we'll say one has *a tree for* Ψ.

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.

A tableau rule applied to node Γ^j_k creates one or more new nodes; its child(ren). If it creates one child, then it is identified as Γ^j_{k+1}. If Γ^j_k creates a second child, it is identified as $\Gamma^{j'}_0$, where j' is a fresh integer. That is, for every child created beyond the first, a new branch is started.

Definition 3.5. A node Γ is a *leaf* node of tree T if no tableau rule has been applied to Γ in T.

Definition 3.6. 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 some branch.

Definition 3.7. Γ is *higher* on a branch than Γ' if and only if Γ is an ancestor of Γ'.

Definition 3.8. 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 branches are closed, else it is open.

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

1. A rule may only be applied to an open leaf node.
2. A rule may not be applied to a formula if it has been applied to that formula higher in the tree, as defined in Definition 3.7.
3. rule $\bot$: If Γ^j_k contains (x, Ψ) and $(x, \neg\Psi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \bot)\}$.
4. rule $\neg$: If Γ^j_k contains $(x, \neg\neg\Psi)$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Psi)\}$.
5. rule $\wedge$: If Γ^j_k contains $(x, \Psi \wedge \Psi')$, then create node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Psi), (x, \Psi')\}$.
6. 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.
7. 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.
8. 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 $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(x, \Phi)\}$.

The second rule constrains rule application to prevent trivial re-applications of rules. For example, if rule $\Box$ were applied to $(0, \Box p_1) \in \Gamma^2_3$, then it may not be applied to $(0, \Box p_1) \in \Gamma^2_4$.

Definition 3.9. 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.

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 2 and 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

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

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 2 and 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—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 $\neg\neg(\neg f \wedge d \wedge \neg h) \wedge \neg[g]_{0.9}(\neg f \wedge h)$.

On the left-hand side of the Figure 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^0_6, Γ^1_0 and Γ^2_0 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.

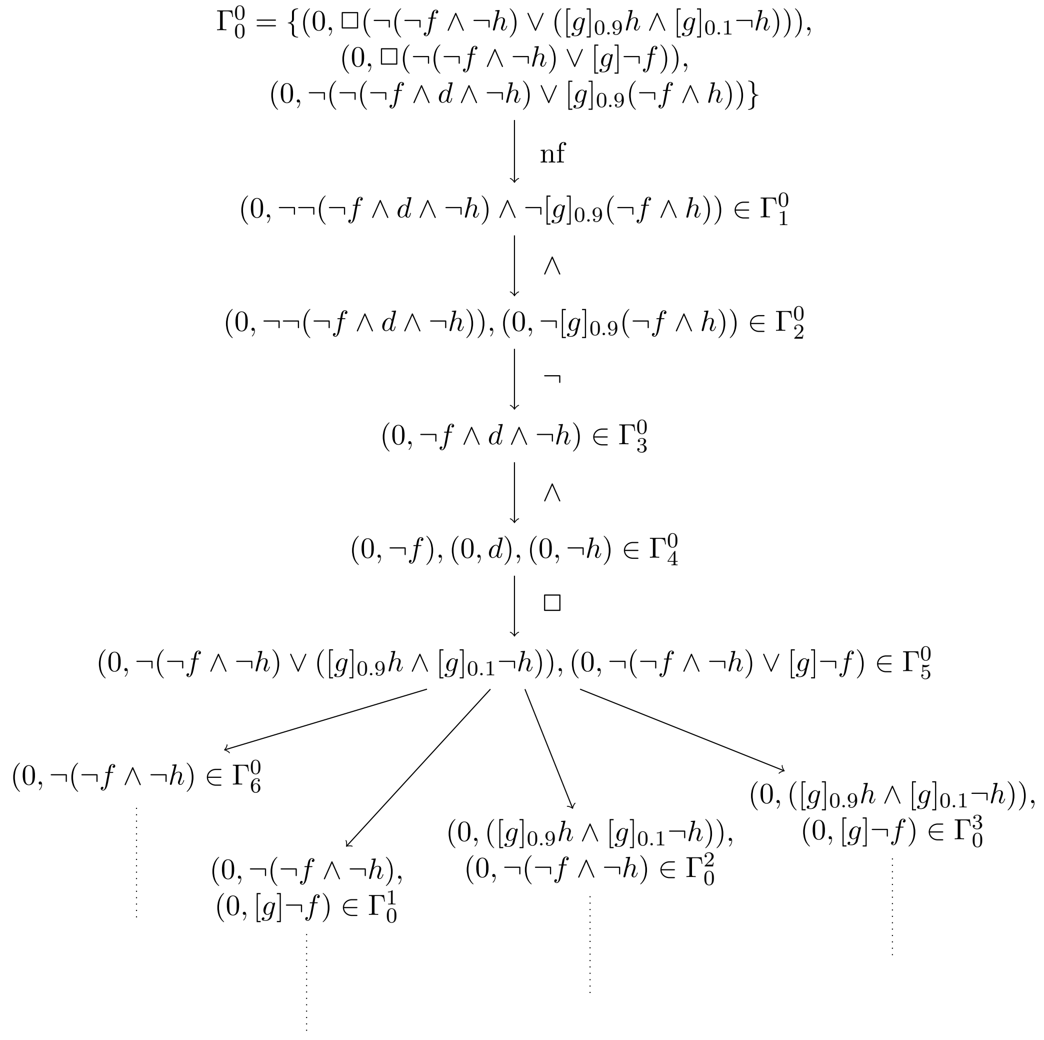
Determining whether the tree finally closes will be seen in the Example section after the SLI phase is described (§ 3.3). Each open leaf node will be involved in the SLI phase for a final decision.

3.2. 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.

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

$$\Gamma^0_0 = \{(0, [\texttt{grab}]_{0.9}\texttt{holding} \wedge [\texttt{grab}]_{0.6}\texttt{holding})\}$$


$\Gamma_0^0 = \{(0, \Box(\neg(\neg f \wedge \neg h) \vee ([g]_{0.9} h \wedge [g]_{0.1} \neg h))),$
$(0, \Box(\neg(\neg f \wedge \neg h) \vee [g]\neg f)),$
$(0, \neg(\neg(\neg f \wedge d \wedge \neg h) \vee [g]_{0.9}(\neg f \wedge h))\}$
nf
$(0, \neg\neg(\neg f \wedge d \wedge \neg h) \wedge \neg[g]_{0.9}(\neg f \wedge h)) \in \Gamma_1^0$
$\wedge$
$(0, \neg\neg(\neg f \wedge d \wedge \neg h)), (0, \neg[g]_{0.9}(\neg f \wedge h)) \in \Gamma_2^0$
$\neg$
$(0, \neg f \wedge d \wedge \neg h) \in \Gamma_3^0$
$\wedge$
$(0, \neg f), (0, d), (0, \neg h) \in \Gamma_4^0$
$\Box$
$(0, \neg(\neg f \wedge \neg h) \vee ([g]_{0.9} h \wedge [g]_{0.1} \neg h)), (0, \neg(\neg f \wedge \neg h) \vee [g]\neg f) \in \Gamma_5^0$
$(0, \neg(\neg f \wedge \neg h) \in \Gamma_6^0$
$(0, \neg(\neg f \wedge \neg h),$
$(0, [g]\neg f) \in \Gamma_0^1$
$(0, ([g]_{0.9} h \wedge [g]_{0.1} \neg h)),$
$(0, \neg(\neg f \wedge \neg h) \in \Gamma_0^2$
$(0, ([g]_{0.9} h \wedge [g]_{0.1} \neg h)),$
$(0, [g]\neg f) \in \Gamma_0^3$

continues below

Figure 2: First part of a tableau tree for $\Box(\neg f \wedge \neg h \rightarrow ([g]_{0.9} h \wedge [g]_{0.1} \neg h)) \wedge \Box(\neg f \wedge \neg h \rightarrow [g]\neg f) \wedge \neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h))$.

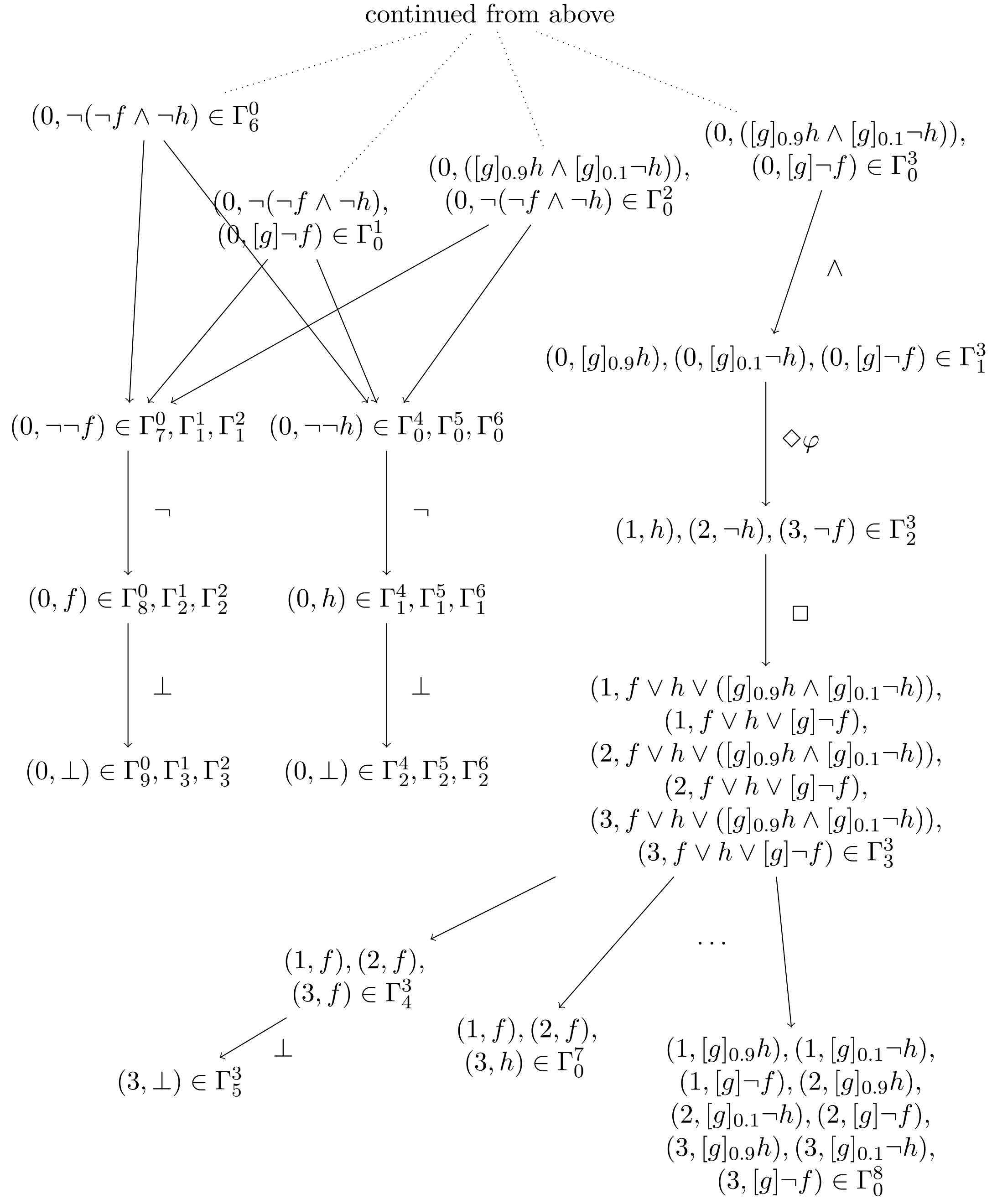

continued from above
$(0, \neg(\neg f \wedge \neg h) \in \Gamma^0_6$
$(0, \neg(\neg f \wedge \neg h),$
$(0, [g]\neg f) \in \Gamma^1_0$
$(0, ([g]_{0.9}h \wedge [g]_{0.1}\neg h)),$
$(0, \neg(\neg f \wedge \neg h) \in \Gamma^2_0$
$(0, ([g]_{0.9}h \wedge [g]_{0.1}\neg h)),$
$(0, [g]\neg f) \in \Gamma^3_0$
$\wedge$
$(0, [g]_{0.9}h), (0, [g]_{0.1}\neg h), (0, [g]\neg f) \in \Gamma^3_1$
$(0, \neg\neg f) \in \Gamma^0_7, \Gamma^1_1, \Gamma^2_1$
$(0, \neg\neg h) \in \Gamma^4_0, \Gamma^5_0, \Gamma^6_0$
$\Diamond\varphi$
$\neg$
$\neg$
$(1, h), (2, \neg h), (3, \neg f) \in \Gamma^3_2$
$(0, f) \in \Gamma^0_8, \Gamma^1_2, \Gamma^2_2$
$(0, h) \in \Gamma^4_1, \Gamma^5_1, \Gamma^6_1$
$\Box$
$\perp$
$\perp$
$(1, f \vee h \vee ([g]_{0.9}h \wedge [g]_{0.1}\neg h)),$
$(1, f \vee h \vee [g]\neg f),$
$(2, f \vee h \vee ([g]_{0.9}h \wedge [g]_{0.1}\neg h)),$
$(2, f \vee h \vee [g]\neg f),$
$(3, f \vee h \vee ([g]_{0.9}h \wedge [g]_{0.1}\neg h)),$
$(3, f \vee h \vee [g]\neg f) \in \Gamma^3_3$
$(0, \perp) \in \Gamma^0_9, \Gamma^1_3, \Gamma^2_3$
$(0, \perp) \in \Gamma^4_2, \Gamma^5_2, \Gamma^6_2$
$\ldots$
$(1, f), (2, f),$
$(3, f) \in \Gamma^3_4$
$\perp$
$(3, \perp) \in \Gamma^3_5$
$(1, f), (2, f),$
$(3, h) \in \Gamma^7_0$
$(1, [g]_{0.9}h), (1, [g]_{0.1}\neg h),$
$(1, [g]\neg f), (2, [g]_{0.9}h),$
$(2, [g]_{0.1}\neg h), (2, [g]\neg f),$
$(3, [g]_{0.9}h), (3, [g]_{0.1}\neg h),$
$(3, [g]\neg f) \in \Gamma^8_0$

Figure 3: Last part of the tableau tree for $\Box(\neg f \wedge \neg h \rightarrow ([g]_{0.9}h \wedge [g]_{0.1}\neg h)) \wedge \Box(\neg f \wedge \neg h \rightarrow [g]\neg f) \wedge \neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h))$.

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, Halpern and Megiddo [12] 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 (§ 3.3) 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. 3.13).

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

Definition 3.10. $W(\Gamma, x) \overset{def}{=} \{w \in C \mid w \models \ell$ for all $(x, \ell) \in \Gamma$ where ℓ is a propositional literal$\}$.

Definition 3.11. $X(\Gamma) \overset{def}{=} \{0, 1, \ldots, x'\}$ are all the labels mentioned in Γ.

Definition 3.12. $W(\Gamma) \overset{def}{=} \bigcup_{x \in X(\Gamma)} W(\Gamma, x)$.

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

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

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 (1) represents two distinct systems of equations.

Definition 3.13. 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 (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 2.3 on page 6.

3.3. The SLI Phase

Definition 3.14. 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 3.15. $F(\Gamma, \alpha, x) \overset{def}{=} \{[\alpha]_q\varphi \mid (x, [\alpha]_q\varphi) \in \Gamma\} \cup \{\neg[\alpha]_q\varphi \mid (x, \neg[\alpha]_q\varphi) \in \Gamma\}$.

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 3.16. A tree is called *finished* after the SLI phase is completed.

Note that all branches of a finished tree are saturated.

Definition 3.17. 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 2 and 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 $w_1 \models \mathtt{full} \land \mathtt{drank} \land \mathtt{holding}$, w_2 as 110 where $w_2 \models \mathtt{full} \land \mathtt{drank} \land \neg\mathtt{holding}$, …, w_8 as 000 where $w_8 \models \neg\mathtt{full} \land \neg\mathtt{drank} \land \neg\mathtt{holding}$. We shall refer to the open leaf node on the RHS in Figures 2 and 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. $F(\Gamma, \mathtt{grab}, 0) = \{[\mathtt{grab}]_{0.9}\mathtt{holding}, [\mathtt{grab}]_{0.1}\neg\mathtt{holding}, [\mathtt{grab}]\neg\mathtt{full}, \neg[\mathtt{grab}]_{0.9}(\neg\mathtt{full} \land \mathtt{holding})\}$.

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

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

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.

4. Properties of the Decision Procedure

4.1. Soundness

Theorem 4.1 (Soundness). *If $\vdash \Psi$ then $\models \Psi$. (Contrapositively, if $\not\models \Psi$ then $\not\vdash \Psi$.)*

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.

Lemma 4.1. *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, $\{(x, \neg\neg\Psi)\} \subseteq \Gamma_k$, 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, $\{(x, \Psi \wedge \Psi')\} \subseteq \Gamma_k$, 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, $\{(x, \Psi \vee \Psi')\} \subseteq \Gamma_k$, 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, $\{(0, \neg[\alpha]_0\varphi)\} \subseteq \Gamma_k$ or $\{(0, [\alpha]_q\varphi)\} \subseteq \Gamma_k$ 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 $\mathcal{S}, w' \models \Phi'$.

- 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 $\mathcal{S}, w''' \models \Phi'$.

- 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 $F(\Gamma, \alpha, x) \overset{def}{=} \{[\alpha]_q\varphi \mid (x, [\alpha]_q\varphi) \in \Gamma\} \cup \{\neg[\alpha]_q\varphi \mid (x, \neg[\alpha]_q\varphi) \in \Gamma\}$.

  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 $\mathcal{S}, w \models \Phi'$.

Corollary 4.1. *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.*

The corollary follows due to the special case when $k = 0$ of Γ_k mentioned in the proof of Lemma 4.1.

It is known that the first-order theory of rational numbers (linear arithmetic; without multiplication) is decidable; the Fourier-Motzkin method [13] and Dines' paper [14], for example, are proofs of this, and Ferrante and Rackoff's method is a more efficient (almost polynomial) variant [15]. Any system of equations and disequalities as they appear in this work, can easily be stated as an applicable first-order theory (see, e.g. Kroening and Strichman [16] 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 Corollary 4.1, every execution of a rule or procedure in the decision procedure is sound.

4.2. Completeness

Theorem 4.2 (Completeness). *If $\models \Psi$ then $\vdash \Psi$. (Contrapositively, if $\not\vdash \Psi$ then $\not\models \Psi$.)*

Let $\psi = \neg\Psi$. Then $\not\vdash \Psi$ 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 $\psi \in \mathcal{L}_{SLAP}$, a 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.

We now start with the description of the construction of a 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 $(x_1, x_2, \dots, x_n)$ such that $w_1 \in W(\Gamma, x_1)$, $w_2 \in W(\Gamma, x_2)$, …, $w_n \in W(\Gamma, x_n)$, where $(w_1, w_2, \dots, w_n) = W(\Gamma)^{\#}$. $\mathcal{S} = \langle W, R\rangle$ 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 $R_\alpha = \{(w_i, w_j, s_j^\alpha) \mid$
  - $w_i, w_j \in W(\Gamma)^{\#}$,
  - $x_i \in X(\Gamma)^{\#}$,
  - $(s_1, s_2, \dots, s_n) \in Z(F(\Gamma, \alpha, x_i))\}$.

Lemma 4.2. *$\mathcal{S}$ is a 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 4.3. *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, Φ_1), (x, Φ_2), …, $(x, \Phi_m) \in \Gamma$.*

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: $(0, t_1 \vee t_2 \vee \cdots \vee t_z)$, $(1, t_1 \vee t_2 \vee \cdots \vee t_z)$, $(2, t_1 \vee t_2 \vee \cdots \vee t_z)$, … $(x', t_1 \vee t_2 \vee \cdots \vee t_z)$. 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 T. For every label $x \in \{0, 1, 2, \ldots, x'\}$, $(x, t_k) \in T \subset \Gamma$ 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 $x \in X(\Gamma)$.

Proposition 4.1. *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 $w' \in W(\Gamma, x)$.*

By construction, $R_\alpha = \{(w, w_j, s_j) \mid x \in X(\Gamma), w \in W(\Gamma, x), w_j \in W(\Gamma)^\#$ 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 $w' \in W(\Gamma, x)$. Proposition 4.1 follows.

Lemma 4.4. *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. Let $\mathcal{S}$ be constructed as described above.

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 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 $\Box$, $(0, \Phi), (1, \Phi), \ldots, (x', \Phi) \in \Gamma$. 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 4.3, 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), $\ldots$, $(x, \Phi_m) \in \Gamma$.

  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 $W(\Gamma)$,

$$\mathcal{S}, w' \models \bigwedge_{\ell \in L(t)} \ell \text{ (for all } w' \in W(\Gamma)). \tag{2}$$

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 $\underline{\text{every}}$ $w_i \in W(\Gamma)^{\#}$. Let w_i be w'' and x_i be x.

Therefore, by (2) and the above argument for dynamic literals,

$$\mathcal{S}, w' \models t \quad \text{(for all } w' \in W(\Gamma)).$$

Corollary 4.2. *By Lemmata 4.2 and 4.4, 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 just after Corollary 4.1, Käufl [17] says that the Inf-Sup-Method developed by Bledsoe [18] and refined by Shostak [19] "is a complete decision procedure for systems of linear inequalities over the rational numbers." Theorem 4.2 follows directly from Corollary 4.2 and the fact that there exist complete methods for determining the feasibility of SLIs as they appear in this work.

4.3. Termination

Definition 4.1. 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 4.5. *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 $\bot$ 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 their 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. *The entailment decision procedure for SLAP terminates.*

PROOF. Due to Lemma 4.5, 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 label assignments in $SoLA(\Gamma)$ and a finite number of worlds in $W(\Gamma)$. 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 label assignment, for each world, for each action in $\mathcal{A}$. 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 [20, 16] and the process thus terminates in this phase.

Corollary 4.3. *The entailment problem for SLAP is decidable.*

Because the procedure is sound (Th. 4.1), complete (Th. 4.2) and terminating (Th. 4.3), entailment is decidable.

5. Related Work

SLAP found inspiration from the logic of action and plans $\mathcal{LAP}$ [11], especially for the $\Box$ operator for marking sentences globally applicable, that is, for marking sentences as *axioms*. Their tableau method was also a useful stating point for our decision procedure. However, $\mathcal{LAP}$ deals with uncertainty of action effects only with disjunction; this is a coarse-grained approach to dealing with uncertainty. Moreover, $\mathcal{LAP}$ allows nesting of modalities, which SLAP does not.

And SLAP found inspiration from the Master's dissertation of Van Diggelen [21] too, in which he 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," [21, p. 33]. An agent is assumed to always know in what world it is and observation is always certain (complete). $\mathcal{L}_{ProbDL}$ is introduced as a modal logic extended with probability theory and epistemic notions. Action-indexed *box* and *diamond* operators are defined, such that the $[\alpha]$ operator has a meaning very similar to ours, but the diamond operator is defined separately, not in terms of the box operator as we do.

Van Diggelen defines an accessibility relation $R : \mathcal{A} \rightarrow 2^{S \times S \times (0,1]}$ for $\mathcal{L}_{ProbDL}$, where S is a (non-empty) set of possible states. The 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. For the same reason, he has to define a diamond operator separately. At first we thought that this is a good idea, but later, we found that it complicates matters extremely: he introduces a probabilistic diamond operator with an unconventional (and in our view, unintuitive) semantic definition, with "unreal" states that are reached (and are inescapable) when an agent performs an impossible action.

Although SLAP uses probability theory, it is not for reasoning *about* probability; it is for reasoning about (probabilistic) actions. There have been many approaches/frameworks for reasoning *about* probability, but most of them are either not concerned with *dynamic environments* [22, 23, 24] or they *are* concerned with change, but they are not actually *logics* [25, 26, 27, 28]. We briefly discuss one of these:

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 early 90s [29]. The result is the Independent Choice Logic (ICL). In 1998 he wrote a paper about the current state of the logic [30]. Poole employs first-order logic as the base logic and then constrains it from various sides. The ICL is not a *logic* as such; it is a decision theoretic framework with some components referring to sets of 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.

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. 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 SLAP or planned extensions to SLAP is; our starting point is thus a simpler logic for a narrower application area.

Some probabilistic logics for reasoning about action and change do exist, but they are not suitable as bases for defining a minimalist logic with an MDP semantics. For instance, De Weerdt et al. [31] present a modal logic to deal with imprecision in robot actions and sensors. Their syntax and semantics is simple and intuitive, yet rich enough to reason about the imprecisions that robots typically need to deal with. They show this through some simple examples.

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 fully address domain specification. Also, their paper focuses on noisy sensing; no attention is given to nondeterminstic actions. All they say (almost) 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," [31].

Iocchi, et al. [32] developed a useful framework for reasoning about agents with sensing, qualitative nondeterminism and probabilistic uncertainty in action outcomes. It is the logic $\mathcal{E}+$. The application area of $\mathcal{E}+$ is plan generation for agents with nondeterministic and probabilistic uncertainy. A major differences of $\mathcal{E}+$ to SLAP is that $\mathcal{E}+$ is based on a fragment of the autoepistemic description logic $\mathcal{ALCK}_{\mathcal{NF}}$ [Donini et al. 2002] for modeling dynamic systems, whereas SLAP is a multi-modal logic. 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 SLAP decision procedure can determine the truth of arbitrary entailments, but we cannot express sequences of actions or plans.

Many popular frameworks for reasoning about action employ or are based on the situation calculus [33]. Reified situations make the meaning of formulae perspicuous. For instance, the framework proposed by Bacchus, Halpern and Levesque [34] and the logic $\mathcal{ESP}$ by Lakemeyer and Levesque [35] 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. The decidability of SLAP sets it apart from first-order logics for reasoning about action (including all logics based on the situation calculus).

6. Conclusion and Future Work

We presented a non-nesting logic (SLAP) for modeling probabilistic transition systems. The logic is based on modal logic with possible worlds semantics. The paper proves that determining whether a 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. Our next research goal is the development SLAOP, a logic for specifying partially observable Markov decision processes (POMDPs) [36, 37]. SLAP's significance lies in its role as a foundation for defining SLAOP. Investigations indicate that the combination of stochastic actions and observations requires solutions to systems of (weakly) nonlinear inequalities and that the particular kind of systems generated in SLAOP are decidable with respect to feasibility. Our next aim is thus to prove that there exists a decidable (w.r.t. validity) logic sufficient for specifying POMDPs.

Iocchi, et al. [32] also say that one of their aims is to extended $\mathcal{E}+$ 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.

Appendix A. More on the Decidability of the SLI Phase

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

$$\begin{array}{rll}
c_{1,1}pr_1 + c_{1,2}pr_2 + \cdots + c_{1,n}pr_n & = q_1 & \wedge \\
c_{2,1}pr_1 + c_{2,2}pr_2 + \cdots + c_{2,n}pr_n & = q_2 & \wedge \\
\vdots & & \\
c_{h,1}pr_1 + c_{h,2}pr_2 + \cdots + c_{h,n}pr_n & = q_h & \wedge \\
c_{h+1,1}pr_1 + c_{h+1,2}pr_2 + \cdots + c_{h+1,n}pr_n & \neq q_{h+1} & \wedge \\
c_{h+2,1}pr_1 + c_{h+2,2}pr_2 + \cdots + c_{h+2,n}pr_n & \neq q_{h+2} & \wedge \\
\vdots & & \\
c_{h+k,1}pr_1 + c_{h+k,2}pr_2 + \cdots + c_{h+k,n}pr_n & \neq q_{h+k} & \wedge \\
pr_1 + pr_2 + \cdots + pr_n & = q*, &
\end{array} \tag{A.1}$$

where each of the first $h + k$ conjuncts ((in)equalities) represents an element in $F(\Gamma, \alpha, x)$ and $q* = 0$ or $q* = 1$ in the last conjunct. 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 represented by the system $\mathbf{Cpr} = \mathbf{q}$ of linear equations, where $\mathbf{C}$ is a $(h+1) \times n$ matrix, $\mathbf{pr}$ is a n-dimensional vector and $\mathbf{q}$ is a $(h+1)$-dimensional vector. Formula $A(\alpha, x)$ (System (A.1)) can then be written as

$$\mathbf{Cpr} = \mathbf{q} \;\wedge\; \bigwedge_{i=1}^{k} \sum_{j=1}^{n} c_{i,j}pr_{i,j} \neq q_i. \tag{A.2}$$

Remark Appendix A.1. A disequation $a_1x_1 + \cdots + a_nx_n \neq b$ is equisatisfiable with

$$(a_1x_1 + \cdots + a_nx_n < b) \quad \vee \quad (-a_1x_1 - \cdots - a_nx_n < -b). \tag{A.3}$$

But Formula (A.3) is not in $T_{\mathbb{Q}}$, although each of the two disjuncts is. By Remark Appendix A.1, System (A.2) 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 $B_1, B_2, \ldots, B_{2^k}$ are 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 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 [16] 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. 18) is feasible.

References

[1] G. Hughes, M. Cresswell, A New Introduction to Modal Logic, Routledge, New York, NY, 1996.

[2] A. Chagrov, M. Zakharyaschev, Modal Logic (Oxford Logic Guides, Vol. 35), Oxford University Press, Oxford, England, 1997.

[3] P. Blackburn, M. D. Rijke, Y. Venema, Modal Logic, Cambridge University Press, Cambridge, UK, 2001.

[4] P. Blackburn, J. Van Benthem, F. Wolter (Eds.), Handbook of Modal Logic, Vol. 3 of Studies in Logic and Practical Reasoning, Elsevier, Amsterdam, The Netherlands / Oxford, UK, 2007.

[5] R. E. Bellman, A markov decision process, Journal of Mathematical Mech. 6 (1957) 679–684.

[6] R. Howard, Dynamic Programming and Markov Processes, MIT Press, Cambridge, MA, 1960.

[7] M. Puterman, Markov Decision Processes: Discrete Dynamic Programming, Wiley, New York, NY, 1994.

[8] G. Rens, T. Meyer, G. Lakemeyer, On the logical specification of probabilistic transition models, in: Proc. of 11th Intl. Symposium on Logical Formalizations of Commonsense Reasoning (COMMONSENSE 2013), 2013.

[9] B. Chellas, Modal Logic: an introduction, Cambridge University Press, Cambridge, MA, 1980.

[10] S. Popkorn, First Steps in Modal Logic, Cambridge University Press, 1994.

[11] M. Castilho, O. Gasquet, A. Herzig, Formalizing action and change in modal logic I: The frame problem, Journal of Logic and Computation 9 (5) (1999) 701–735.

[12] R. Fagin, J. Y. Halpern, N. Megiddo, A logic for reasoning about probabilities, Information and Computation 87 (1990) 78–128.

[13] T. Motzkin, Beiträge zur theorie der linearen ungleichungen, Ph.D. thesis, Universitat Zurich (1936).

[14] L. Dines, Systems of linear inequalities, The Annals of Mathematics 20 (3) (1919) 191–199.

[15] J. Ferrante, C. Rackoff, A decision procedure for the first order theory of real addition with order, SIAM J. Comput. 4 (1) (1975) 69–76.

[16] D. Kroening, O. Strichman, Decision Procedures: An Algorithmic Point of View, Texts in Theoretical Computer Science, Springer Verlag, 2008.

[17] T. Käufl, Reasoning about systems of linear inequalities, in: E. Lusk, R. Overbeek (Eds.), Proc. of 9th Intl. Conf. on Automated Deduction, Vol. 310 of LNCS, Springer Verlag, 1988, pp. 563–572.

[18] W. W. Bledsoe, A new method for proving certain Presburger formulas, in: Proceedings of the 4th international joint conference on Artificial intelligence - Volume 1, IJCAI'75, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1975, pp. 15–21.
URL `http://dl.acm.org/citation.cfm?id=1624626.1624629`

[19] R. E. Shostak, On the SUP-INF method for proving Presburger formulas, J. ACM 24 (4) (1977) 529–543. doi:10.1145/322033.322034.
URL `http://doi.acm.org/10.1145/322033.322034`

[20] G. B. Dantzig, Linear Programming and Extensions, Princeton University Press, 1963 & 1998.

[21] J. Van Diggelen, Using modal logic in mobile robots, Master's thesis, Cognitive Artificial Intelligence, Utrecht University (2002).

[22] F. Bacchus, Representing and Reasoning with Uncertain Knowledge, MIT Press, Cambridge, MA, 1990.

[23] R. Fagin, J. Y. Halpern, Reasoning about knowledge and probability, Journal of the ACM 41 (2) (1994) 340–367.

[24] A. Shirazi, E. Amir, Probabilistic modal logic, in: Proc. of 22nd Natl. Conf. on Artificial Intelligence (AAAI-07), AAAI Press, 2007, pp. 489–494.

[25] D. Poole, Decision theory, the situation calculus and conditional plans, Linköping Electronic Articles in Computer and Information Science 8 (3).

[26] C. Boutilier, R. Reiter, M. Soutchanski, S. Thrun, Decision-theoretic, high-level agent programming in the situation calculus, in: Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00), AAAI Press, Menlo Park, CA, 2000, pp. 355–362.

[27] B. Bonet, H. Geffner, Planning and control in artificial intelligence: A unifying perspective, Applied Intelligence 14 (3) (2001) 237–252.

[28] 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).

[29] D. Poole, The independent choice logic for modeling multiple agents under uncertainty, Artificial Intelligence, Special Issue on Economic Principles of Multi-Agent Systems 94 (1997) 7–56.

[30] D. Poole, The independent choice logic and beyond, in: L. D. Raedt, P. Frasconi, K. Kersting, S. Muggleton (Eds.), Probabilistic Inductive Logic Programming: Theory and Application, Vol. 4911 of Lecture Notes in Artificial Intelligence, Springer Verlag, 2008, pp. 222–243.

[31] M. De Weerdt, F. De Boer, W. Van der Hoek, J.-J. Meyer, Imprecise observations of mobile robots specified by a modal logic, in: Proc. of Fifth annual conference of the Advanced School for Computing and Imaging (ASCI-99), 1999, pp. 184–190.

[32] L. Iocchi, T. Lukasiewicz, D. Nardi, R. Rosati, Reasoning about actions with sensing under qualitative and probabilistic uncertainty, ACM Transactions on Computational Logic 10 (1) (2009) 5:1–5:41.

[33] R. Reiter, Knowledge in action: logical foundations for specifying and implementing dynamical systems, MIT Press, Massachusetts/England, 2001.

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

[35] A. Gabaldon, G. Lakemeyer, $\mathcal{ESP}$: A logic of only-knowing, noisy sensing and acting, in: Proc. of 22nd Natl. Conf. on Artificial Intelligence (AAAI-07), AAAI Press, 2007, pp. 974–979.

[36] G. E. Monahan, A survey of partially observable Markov decision processes: Theory, models, and algorithms, Management Science 28 (1) (1982) 1–16.

[37] W. Lovejoy, A survey of algorithmic methods for partially observed Markov decision processes, Annals of Operations Research 28 (1991) 47–66.