A Logic for Specifying Stochastic Actions and Observations

Gavin Rens[1], Thomas Meyer[1], and Gerhard Lakemeyer[2]

[1] Centre for Artificial Intelligence Research, University of KwaZulu-Natal, and CSIR Meraka, South Africa
{grens,tmeyer}@csir.co.za
[2] RWTH Aachen University, Germany
gerhard@cs.rwth-aachen.de

Abstract. We present a logic inspired by partially observable Markov decision process (POMDP) theory for specifying agent domains where the agent's actuators and sensors are noisy (causing uncertainty). The language features modalities for actions and predicates for observations. It includes a notion of probability to represent the uncertainties, and the expression of rewards and costs are also catered for. One of the main contributions of the paper is the formulation of a sound and complete decision procedure for checking validity of sentences: a tableau method which appeals to solving systems of equations. The tableau rules eliminate propositional connectives, then, for all open branches of the tableau tree, systems of equations are generated and checked for feasibility. This paper presents progress made on previously published work.

1 Introduction and Related Work

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. In situations where the oil-can is full, the robot gets five units of reward for gabbing the can, and it gets ten units of reward for a drink action.

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'.

The oil-drinking domain is (partially) formalized as follows. The robot has the set of (intended) actions $\mathcal{A} = \{\texttt{grab}, \texttt{drink}, \texttt{weigh}, \texttt{replace}\}$ with expected meanings. The robot can make observations only from the set $\Omega = \{\texttt{obsNil}, \texttt{obsLight}, \texttt{obsMedium}, \texttt{obsHeavy}\}$. 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`. 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 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?

Modal logic is considered to be well suited to reasoning about beliefs and changing situations [6, 14]. Partially observable Markov decision process (POMDP) theory [30, 17] has proven to be a good general framework for formalizing dynamic stochastic systems. Our goal is to integrate logic with stochastic actions and observations, taking the semantics of POMDPs in particular. To our knowledge, there exists no such logic; see the discussion of related work below. This paper though, concerns work that is a step towards that goal. Here we present the Specification Logic of Actions and Observations with Probability (SLAOP). With SLAOP, POMDP models can be represented compactly.

The present version of SLAOP is an extension of the Specification Logic of Actions with Probability (SLAP) [24, 27], but with an improved completeness proof due to a new decision procedure. 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 the results of [26] to add observations as first-class objects.

A *preliminary* version of SLAOP has been presented at a doctoral consortium [23]. Since then, significant progress has been made. We mention only some of the major changes. Firstly, the present version of SLAOP 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. § 3.2) is sufficient for specifying perception probabilities (cf. § 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.

A formal approach how to specify probabilistic transition models with SLAP has been published [24], and there, a solution to the frame problem for SLAP is also presented. That frame solution can easily be employed for SLAOP. A decision procedure for validity checking in SLAP is presented in a journal article [27]. The procedure for SLAP is simpler than for SLAOP because it does not use the 'label assignment' approach (cf. § 4.3). We opted for a decision procedure with label assignments for SLAOP because the proof of completeness is then easier to understand (see the accompanying technical report [25]), and sentences of a certain form which are not allowed in SLAP are allowed in SLAOP (see § 4.3).

Related work is discussed next. Then Section 3 presents the syntax and semantics of SLAOP. Section 4 presents the two-phase decision procedure. Section 5 provides examples of application of the decision procedure. Some concluding remarks are made in Section 6.

2 Related Work

Several frameworks exist for reasoning about probabilistic inference in static domains [1, 9, 13, 16, 29, 33]. 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. In SLAOP, the focus is more on how stochastic actions change the physical state of a system. Some of these logics are concerned with how knowledge changes as new information is gained, however, the information received is not seen as an observation *object*. 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. SLAOP can express the fact that information (in the form of observation objects) may be incorrect to some degree. This ability of SLAOP is carried over from the SLAP logic [24, 27].

Poole's Independent Choice Logic using the situation calculus (ICL_{SC}) [20] is a relatively rich framework, with acyclic logic programs which may contain variables, quantification and function symbols. For certain applications, SLAOP may be preferred due to its comparative simplicity. And because SLAOP's semantics is very close to that of standard POMDP theory, it may be easier to understand by people familiar with POMDPs. Finally, decidability of inferences made in the ICL_{SC} are, in general, not guaranteed.

Bonet and Geffner [3] present a framework with heuristic search algorithms for modeling and solving MDPs and POMDPs. It seems similar to the ICL_{SC}

in its application area. Their framework uses high-level logical representations, but it is not presented as a logic, nor does it empoy logical entailment.

DTGolog [5] is a programming language, rather than a logic, and it does not deal with stochastic observations.

PODTGolog [22] is another logic programming framework which does deal with stochastic observations, but it does not have a well defined semantics.

Many popular frameworks for reasoning about action, employ or are based on the situation calculus [22]. Reified situations make the meaning of formulae perspicuous. However, the situation calculus seems too rich and expressive for our purposes, and it would be desirable to remain decidable, hence the restriction to a propositional modal framework. The validity problem for SLAOP is decidable, which sets it apart from first-order logics for reasoning about action (including the situation calculus) or reasoning with probabilities (including BHL's approach [2] and $\mathcal{ESP}$ [11]). In other words, having a decidable formalism to reason about POMDP's is considered an asset and would set 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.

Iocchi *et al.* [15] present a logic called $\mathcal{E}+$ for reasoning about agents with sensing, qualitative nondeterminism and probabilistic uncertainty in action outcomes. Planning with sensing and uncertain actions is also dealt with. 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.

There are some logics that come closer to what we desire [8, 34, 11, 33], that is, 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 [4, 35, 28], 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.

Systems of linear inequalities are at the heart of Nilsson's probabilistic logic [19], which has been extended with stochastic actions by Thiébaux *et al.* [32]. Fagin, Halpern and Megiddo [10] use a similar idea to prove that the axiomatization of their logic for reasoning about probabilities is complete. None of these deals with observations.

3 Specification Logic of Actions and Observations with Probability

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

3.1 Syntax

The vocabulary of our 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* $\mathbb{R}$,[3]
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 $\mathbb{R}_{[0,1]}$. 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 $q \in \mathbb{R}_{[0,1]}$, and predicates $(\varsigma \mid \alpha : q)$, one for each pair in $\Omega \times \mathcal{A}$ and $q \in \mathbb{R}_{[0,1]}$.

Definition 1. *Let $f \in \mathcal{F}$, $\alpha \in (\mathcal{A} \cup V_{\mathcal{A}})$, $\varsigma \in (\Omega \cup V_{\Omega})$, $v \in (V_{\mathcal{A}} \cup V_{\Omega})$, $q \in \mathbb{R}_{[0,1]}$ and $r \in \mathbb{R}$. The language of SLAOP, denoted $\mathcal{L}_{SLAOP}$, is the least set of Ψ defined by the grammar:*

$$
\begin{aligned}
\varphi ::=&\ f \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi.\\
\Phi ::=&\ \varphi \mid \alpha = \alpha \mid \varsigma = \varsigma \mid \mathit{Reward}(r) \mid \mathit{Cost}(\alpha, r) \mid [\alpha]_q\varphi \mid (\varsigma \mid \alpha : q) \mid\\
&\ (\forall v)\Phi \mid \neg\Phi \mid \Phi \wedge \Phi.\\
\Psi ::=&\ \Phi \mid \Box\Phi \mid \neg\Psi \mid \Psi \wedge \Psi.
\end{aligned}
$$

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.

(For SLAP, $\Phi ::= \varphi \mid [\alpha]_q\varphi \mid \neg\Phi \mid \Phi \wedge \Phi$.) 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 transitions 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'.

[3] In SLAP [27] and the previous version of SLAOP [23], rational numbers were used. Due to our completeness proof relying on Tarski's quantifier elimination method [31] which involves real numbers, we use real numbers here.

$[\alpha]_q\varphi$ 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'.

$\langle\alpha\rangle\varphi$ 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—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. 3 for the formal definition). $(\exists v)\Phi$ abbreviates $\neg(\forall v)\neg\Phi$.

3.2 Semantics

SLAOP extends SLAP. SLAP structures are non-standard: They have 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 fluent. Let C be the set of all possible functions w. We call C the *conceivable worlds*.

SLAP structures are comparable to Markov decision processes (MDPs) [21] 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.[4]

Definition 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 $R_{\alpha} : (W \times W) \mapsto \mathbb{R}_{[0,1]}$ is a total function from pairs of worlds into the reals; That is, R is a mapping that provides an accessibility*

[4] Specification of the initial belief-state is required at a higher level of reasoning. It is left for future work.

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 $Q_\alpha : (W \times O) \mapsto \mathbb{R}_{[0,1]}$ 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 $\sum_{o \in O} Q_\alpha(w^+, o) = 0$;*
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 $pr \in \mathbb{R}_{[0,1]}$ 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 $pr \in \mathbb{R}_{[0,1]}$ 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 $w \in C$. 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 $w \in C$.

Definition 3 (Truth Conditions). *Let $\mathcal{S}$ be a SLAOP structure, with $\alpha, \alpha' \in \mathcal{A}$, $q, pr \in \mathbb{R}_{[0,1]}$ 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 φ is *valid* in a SLAOP structure (denoted $\mathcal{S} \models \varphi$) if $\mathcal{S}, w \models \varphi$ for every $w \in W$. φ is *SLAOP-valid* (denoted $\models \varphi$) if φ is true in every structure $\mathcal{S}$. If $\models \theta \leftrightarrow \psi$, we say θ and ψ are *semantically equivalent* (abbreviated $\theta \equiv \psi$).

φ is *satisfiable* if $\mathcal{S}, w \models \varphi$ 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 $w \models \varphi$ instead of $\mathcal{S}, w \models \varphi$ when φ is a propositional formula.

Let $\mathcal{K}$ be a finite subset of $\mathcal{L}_{SLAOP}$. 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 $\mathcal{S}, w \models \bigwedge_{\theta \in \mathcal{K}} \theta$ then $\mathcal{S}, w \models \psi$. We shall also say that $\mathcal{K}$ *entails* ψ whenever $\mathcal{K} \models \psi$. If $\{\theta\} \models \psi$ then we simply write $\theta \models \psi$. In fact, $\mathcal{K} \models \Psi$ if and only if $\models \bigwedge_{\theta \in \mathcal{K}} \theta \rightarrow \Psi$ (i.e., $\mathcal{K}$ entails Ψ iff $\bigwedge_{\theta \in \mathcal{K}} \theta \rightarrow \Psi$ 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 $Def(\varphi)$ be all the definitive formulae which entail φ, that is, $Def(\varphi) = \{\delta \in \mathcal{L}_{SLAOP} \mid \delta \text{ is definitive and } \delta \models \varphi\}$.

4 Decision Procedure for SLAOP Entailment

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.

4.1 The Tableau Phase

The necessary definitions and terminology are given next.

A *labeled formula* is a pair (x, Ψ), where $\Psi \in \mathcal{L}_{SLAOP}$ is a formula and x is an integer called the *label* of Ψ. A *node* Γ^j_k with superscript j (the *branch* index) and subscript k (the *node* index), is a set of labeled formulae. The initial node, that is, Γ^0_0, to which the tableau rules must be applied, is called the *trunk*.

Definition 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_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.

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

Definition 5. *Γ is* higher *on a branch than Γ' if and only if Γ is an ancestor of Γ'.*

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.

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. A rule may only be applied to an open leaf node. To constrains rule application to prevent trivial re-applications of rules, a rule may not be applied to a formula if it has been applied to that formula higher in the tree, as in Definition 5. For example, if rule $\Box$ were applied to $\{(0, \Box p_1), (1, \neg[go]_0 p_2)\} \subset \Gamma_3^2$, then it may not be applied to $\{(0, \Box p_1), (1, \neg[go]_0 p_2)\} \subset \Gamma_4^2$.

Let Γ_k^j be a leaf node.

- rule $\bot$: If Γ_k^j contains (n, Φ) and $(n, \neg\Phi)$, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \bot)\}$.
- rule $\neg$: If Γ_k^j contains $(n, \neg\neg\Phi)$, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \Phi)\}$.
- rule $\wedge$: If Γ_k^j contains $(n, \Phi \wedge \Phi')$, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \Phi), (n, \Phi')\}$.
- rule $\vee$: If Γ_k^j contains $(n, \neg(\Phi \wedge \Phi'))$, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \neg\Phi)\}$ and node $\Gamma_0^{j'} = \Gamma_k^j \cup \{(n, \neg\Phi')\}$, where j' is a fresh integer.
- rule $=$: If Γ_k^j contains $(n, c = c')$ and c and c' are distinct constants, or if Γ_k^j contains $(n, \neg(c = c'))$ and c and c' are identical constants, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \bot)\}$.
- rule $\Diamond\varphi$: If Γ_k^j contains $(0, \neg[\alpha]_0\varphi)$ or $(0, [\alpha]_q\varphi)$ for $q > 0$, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \varphi)\}$, where n is a fresh integer.
- rule obs: If Γ_k^j contains $(x, \neg[\alpha]_0\varphi)$ or $(x, [\alpha]_q\varphi)$ for $q > 0$ and some x, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(0, \Box(\delta_1 \rightarrow (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \Box(\delta_2 \rightarrow (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)) \vee \cdots \vee \Box(\delta_n \rightarrow (\exists v^\varsigma)\neg(v^\varsigma \mid \alpha : 0)))\}$, where $\delta_i \in Def(\varphi)$.
- rule $\Box$: If Γ_k^j contains $(0, \Box\Phi)$ and (n, Φ') for any $n \geq 0$, and if it does not yet contain (n, Φ), then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \Phi)\}$.
- rule $\Diamond$: If Γ_k^j contains $(0, \neg\Box\Phi)$, then create node $\Gamma_{k+1}^j = \Gamma_k^j \cup \{(n, \neg\Phi)\}$, where n is a fresh integer.

One might wonder why there is not a rule to deal with the case when Γ_k^j contains $(x, [\alpha]_q\varphi)$ and $(x, \neg[\alpha]_q\varphi)$, or no rule for when Γ_k^j contains $(x, (\varsigma \mid \alpha : r))$ and $(x, (\varsigma \mid \alpha : r'))$ where $r \neq r'$. As will be seen in Section 4.2, these and similar cases are dealt with.

Definition 6. *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.*

Once the tableau phase is completed, inconsistencies are sought for each open branch of the saturated tree. 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. Before the second phase can be explained, we need to explain how a system of inequalities (SI) can be generated from a set of dynamic and perception literals.

4.2 Systems of Inequalities

Definition 7. $W(\Gamma, n) \stackrel{def}{=} \{w \in C \mid w \models \ell \text{ for all } (n, \ell) \in \Gamma \text{ where } \ell \text{ is a propositional literal}\}$. $W(\Gamma) \stackrel{def}{=} \bigcup_{x \in \{0,1,\ldots,n'\}} W(\Gamma, x)$, *where* n' *is the largest label mentioned in* Γ.

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 $pr_k^{\alpha} \in \mathbb{R}_{[0,1]}$. One can generate

$$c_{i,1}pr_1^{\alpha} + c_{i,2}pr_2^{\alpha} + \cdots + c_{i,n}pr_n^{\alpha} = q_i \text{ and } c_{i,1}pr_1^{\alpha} + c_{i,2}pr_2^{\alpha} + \cdots + c_{i,n}pr_n^{\alpha} \neq q_i,$$

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.

Adding an equation

$$pr_1^{\alpha} + pr_2^{\alpha} + \cdots + pr_n^{\alpha} = \lceil pr_1^{\alpha} + pr_2^{\alpha} + \cdots + pr_n^{\alpha} \rceil$$

will ensure that either $\sum_{w^+ \in W(\Gamma)} R_{\alpha}(w^-, w^+) = 1$ or $\sum_{w^+ \in W(\Gamma)} R_{\alpha}(w^-, w^+) = 0$, as stated in Definition 2 on page 7.

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

$$pr_j^{\sigma} = q_j \text{ and } pr_j^{\sigma} \neq q_j$$

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 $pr_j^{\sigma} \in \{pr_1^{\varsigma}, \ldots, pr_2^{\varsigma}, \ldots, pr_m^{\varsigma}\}$.

Adding an equation

$$pr_1^{\varsigma} + pr_2^{\varsigma} + \cdots + pr_2^{\varsigma} + \cdots + pr_m^{\varsigma} = \lceil pr_1^{\varsigma} + pr_2^{\varsigma} + \cdots + pr_2^{\varsigma} + \cdots + pr_m^{\varsigma} \rceil.$$

ensures that either $\sum_{o \in O} Q_{\alpha}(w^+, o) = 1$ or $\sum_{o \in O} Q_{\alpha}(w^+, o) = 0$, as stated in Definition 2 on page 7.

Let $\Delta(\alpha)$ be a set of dynamic literals mentioning α and let $\Omega(\alpha)$ be a set of perception literals involving α. Let $S(\Delta(\alpha))$ and $S(\Omega(\alpha))$ be the systems formed from $\Delta(\alpha)$, respectively, $\Omega(\alpha)$. Let $\mathbf{v}$ be the vector of all variables mentioned in $S(\Delta(\alpha))$ or $S(\Omega(\alpha))$. $Z(\Delta(\alpha))$ 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 $pr_i^\alpha \in \mathbf{v}$ for $i = 1, 2, \ldots, n$, respectively, assigning s_j^ς to $pr_j^\varsigma \in \mathbf{v}$ 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 $[\texttt{replace}]_{0.43}(\texttt{full} \wedge \neg\texttt{holding})$ and $\neg[\texttt{replace}]_{0.43}(\texttt{full} \wedge \neg\texttt{holding})$. Then $S(\Delta(\texttt{replace}))$ will contain

$$\begin{aligned} 0 + pr_2^\alpha + 0 + pr_4^\alpha + 0 + 0 + 0 + 0 &= 0.43 \\ 0 + pr_2^\alpha + 0 + pr_4^\alpha + 0 + 0 + 0 + 0 &\neq 0.43. \end{aligned}$$

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 $S(\Delta(\texttt{replace}))$.

The interested reader can refer to the technical report [25] for a more thorough explication of the generation of SIs.

4.3 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.

In SLAP, sentences of the form $\neg\Box\Phi$ are not in the language. The reason is that the decision procedure for SLAP [27] would not notice certain contradictions which may occur due to such sentences being allowed. In SLAOP, sentences of the form $\neg\Box\Phi$ are in the language, because the label assignment procedure described below picks up the contradictions which may occur.

Informally, x mentioned in Γ could represent any one of the worlds in $W(\Gamma, x)$. Now suppose $(x, \Phi), (x', \Phi') \in \Gamma$ 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:

- $SoLA(\Gamma) \stackrel{def}{=} \{(0{:}w^1, 1{:}w^2, \ldots, x'{:}w^{x'}) \mid w^x \in W(\Gamma, x)\}$, where 0, 1, …, x' are all the labels mentioned in Γ. We shall call an element of $SoLA(\Gamma)$ a *label assignment*. $LA(\Gamma)$ denotes an element of $SoLA(\Gamma)$.
- $E(\Gamma, x) \stackrel{def}{=} \{(x, \Phi) \in \Gamma \mid \Phi$ 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 $\alpha\}$.
- $E(\Gamma, LA, w) \stackrel{def}{=} \bigcup_{x:w \in LA(\Gamma)} E(\Gamma, x)$.
- $F(\Gamma, \alpha, x) \stackrel{def}{=} \{[\alpha]_q\varphi \mid (x, [\alpha]_q\varphi) \in \Gamma\} \cup \{\neg[\alpha]_q\varphi \mid (x, \neg[\alpha]_q\varphi) \in \Gamma\}$.
- $F(\Gamma, \alpha, LA, w) \stackrel{def}{=} \bigcup_{x:w \in LA(\Gamma)} F(\Gamma, \alpha, x)$.
- $G(\Gamma, \alpha, x) \stackrel{def}{=} \{(\varsigma \mid \alpha : q) \mid (x, (\varsigma \mid \alpha : q)\varphi) \in \Gamma\} \cup \{\neg(\varsigma \mid \alpha : q) \mid (x, \neg(\varsigma \mid \alpha : q)) \in \Gamma\}$.
- $G(\Gamma, \alpha, LA, w) \stackrel{def}{=} \bigcup_{x:w \in LA(\Gamma)} G(\Gamma, \alpha, x)$.

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.

Do the following for every $LA \in SoLA(\Gamma^j_k)$. If one of the following three 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 $w \in W(\Gamma^j_k)$, $Z(F(\Gamma^j_k, \alpha, LA, w)) = \emptyset$ or $Z(G(\Gamma^j_k, \alpha, LA, w)) = \emptyset$.

If every $LA \in SoLA(\Gamma^j_k)$ is marked as "unsat", then create new leaf node $\Gamma^j_{k+1} = \Gamma^j_k \cup \{(0, \bot)\}$.

That is, if for all logically correct ways of assigning possible worlds to labels (i.e., for all the label assignments in $SoLA(\Gamma^j_k)$), no assignment (LA) satisfies all formulae in Γ^j_k, then Γ^j_k is unsatisfiable.

Definition 8. *A tree is called* finished *after the label assignment phase is completed.*

Definition 9. *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 $\nvdash \Psi$.*

Theorem 1 (Decidability). *Determining whether a sentence is SLAOP-valid is decidable.*

Proof. The proof is sketched; an accompanying technical report [25] presents the full proof. The proof shows that the decision procedure is sound, complete and terminating, thus decidable.

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. This is shown using induction on the height of a node in a tableau tree, and looking at each tableau rule and the label assignment phase.

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 $\psi \in \mathcal{L}_{SLAOP}$, 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. That is, we show (i) how to construct a structure $\mathcal{S}$ from the information contained in the leaf node Γ of any open branch of a finished tree and (ii) that for all $(x, \Phi) \in \Gamma$, $\mathcal{S}, w \models \Phi$ for $x{:}w \in LA$, for the label assignment LA which is known to exist. Point (ii) relies on induction on the structure of the formulae in Γ.

Termination. Finally, by showing that all trees will become saturated and that the label assignment phase always terminates, it follows that the whole procedure terminates. In particular, rule obs cannot cause cycles because $\Box(\delta_1 \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)) \vee \Box(\delta_2 \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0)) \vee \cdots \vee \Box(\delta_n \rightarrow (\exists v^{\varsigma})\neg(v^{\varsigma} \mid \alpha : 0))$ is not dynamic; it can thus not make rule obs applicable again. That is, rule obs can only cause other rules to become applicable; rules which add $\bot$ to the new node, and rules with the subformula property.

5 Examples

The following abbreviations for constants will be used: `grab` $:= g$, `weigh` $:= w$, `full` $:= f$, `drank` $:= d$, `holding` $:= h$, `obsHeavy` $:= oH$, `obsMedium` $:= oM$ and `obsLight` $:= oL$.

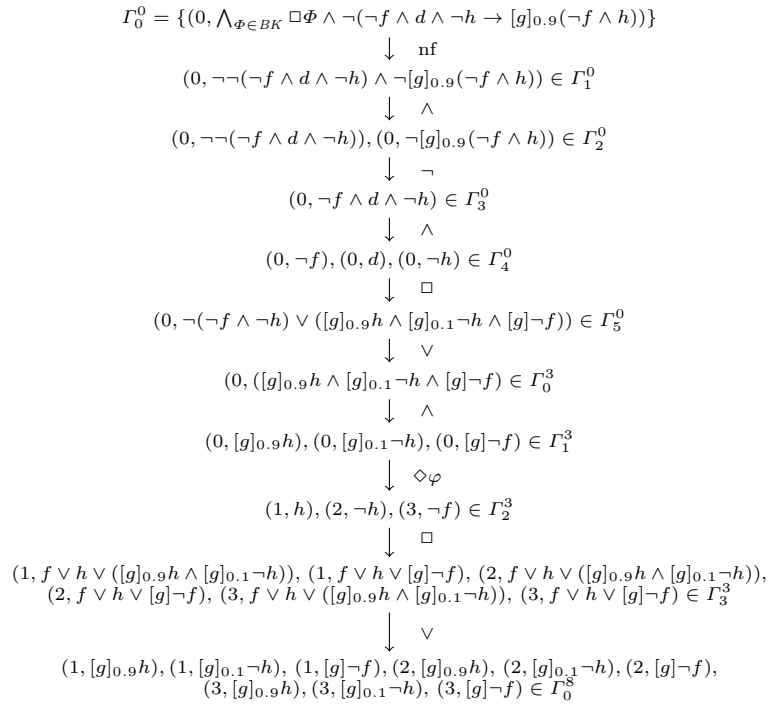

Fig. 1. One branch of a tree for proving that $\{\Box\Phi \mid \Phi \in BK\}$ entails $\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h)$.

In Figures 1 and 2, 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.

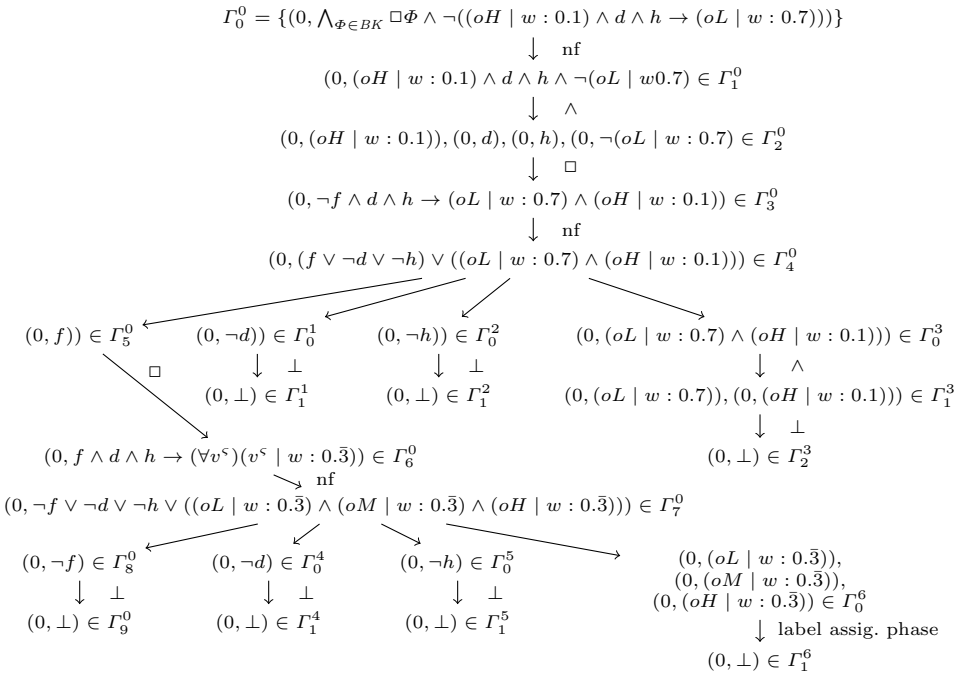

Fig. 2. A tree for proving that $\{\Box\Phi \mid \Phi \in BK\}$ entails $(oH \mid w : 0.1) \wedge d \wedge h \rightarrow (oL \mid w : 0.7)$.

Suppose the following *domain axioms*[5] are part of the robot's background knowledge BK for the oil-drinking scenario.

$$
\begin{aligned}
&f \wedge d \wedge h \rightarrow (\forall v^{\varsigma})(v^{\varsigma} \mid w : 0.\bar{3})\\
&f \wedge \neg d \wedge h \rightarrow (oL \mid w : 0.1) \wedge (oH \mid w : 0.7)\\
&((f \wedge \neg d) \vee (\neg f \wedge d)) \wedge h \rightarrow (oM \mid w : 0.2)\\
&\neg f \wedge d \wedge h \rightarrow (oL \mid w : 0.7) \wedge (oH \mid w : 0.1)\\
&\neg f \wedge \neg d \wedge h \rightarrow (oL \mid w : 0.5) \wedge (oM \mid w : 0.3) \wedge (oH \mid w : 0.2)\\
&\neg f \wedge \neg h \rightarrow [g]_{0.9} h \wedge [g]_{0.1} \neg h \wedge [g] \neg f.
\end{aligned}
$$

For the first example, we claim that $\{\Box\Phi \mid \Phi \in BK\} \models \neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h)$. Figure 1 shows only one branch of a tree for

$$\bigwedge_{\Phi \in BK} \Box\Phi \wedge \neg(\neg f \wedge d \wedge \neg h \rightarrow [g]_{0.9}(\neg f \wedge h)). \tag{1}$$

[5] 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.

For the claim to hold, the tree for (1) must close. We'll only show that the branch in Figure 1 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 $w_2 \models f \wedge d \wedge \neg h$, ..., w_8 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 $W(\Gamma, 0) = \{010\}$, $W(\Gamma, 1) = \{111, 101, 011, 001\}$, $W(\Gamma, 2) = \{110, 100, 010, 000\}$ and $W(\Gamma, 3) = \{011, 010, 001, 000\}$, 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 $F(\Gamma, grab, 0) \subseteq F(\Gamma, \texttt{grab}, LA, 010)$ for all $LA \in SoLA(\Gamma)$. And note that $F(\Gamma, grab, 0)$ 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

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

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 $\{\Box\Phi \mid \Phi \in BK\} \models (oH \mid w : 0.1) \wedge d \wedge h \rightarrow (oL \mid w : 0.7)$. Figure 2 shows the closed tree for

$$\bigwedge_{\Phi \in BK} \Box\Phi \wedge \neg((oH \mid w : 0.1) \wedge d \wedge h \rightarrow (oL \mid w : 0.7)).$$

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 $(oH \mid w : 0.\overline{3})$, which will cause all SIs to be infeasible. Hence, the label assignment phase will create a new node containing $(0, \perp)$ at the end of the branch.

6 Conclusion

A decidable logic with a semantics closely related to partially observable Markov decision processes (POMDPs) was presented. The logic 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.

Two examples were provided in this paper, which give an indication of how SLAOP-validity is computed. In a sequent paper, we would like to explain the formal approach of how SLAOP is used to give a complete specification of a domain.

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 future, to make the language more expressive, the syntax and semantics of these kinds of sentences can be replaced with sentences which have a meaning 'probability is less that, less than or equal to, etc. q.'

An important next step for SLAOP would be to add the ability to express sequences of actions, and then evaluate the part of the sentence occurring after the sequence.

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 that, less than or equal to, etc. some given value. Generating POMDP policies is also on the cards for the future of SLAOP.

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 SI phase has the potential for being very expensive. These are linear systems of equations; one could thus investigate Linear Programming methods [7, 18, 12] 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 paper—may be intractable, it is important to have a decision procedure as a launchpad for tackling the computational complexity. We would like implement some extended version of SLAOP. Determining the complexity of an optimized entailment decision procedure may be attempted before an implementation, though.

References

1. Bacchus, F.: Representing and Reasoning with Uncertain Knowledge. MIT Press, Cambridge, MA (1990)
2. Bacchus, F., Halpern, J.Y., Levesque, H.J.: Reasoning about noisy sensors and effectors in the situation calculus. Artificial Intelligence 111(1–2), 171–208 (1999)
3. Bonet, B., Geffner, H.: Planning and control in artificial intelligence: A unifying perspective. Applied Intelligence 14(3), 237–252 (2001)
4. Boutilier, C., Poole, D.: Computing optimal policies for partially observable decision processes using compact representations. In: Proc. of 13th Natl. Conf. on Artificial Intelligence. pp. 1168–1175 (1996)

5. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S.: 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), pp. 355–362. AAAI Press, Menlo Park, CA (2000)
6. Chellas, B.: Modal Logic: an introduction. Cambridge University Press, Cambridge, MA (1980)
7. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press (1963 & 1998)
8. De Weerdt, M., De Boer, F., Van der Hoek, W., Meyer, J.J.: 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). pp. 184–190 (1999)
9. Fagin, R., Halpern, J.Y.: Reasoning about knowledge and probability. Journal of the ACM 41(2), 340–367 (1994)
10. Fagin, R., Halpern, J.Y., Megiddo, N.: A logic for reasoning about probabilities. Information and Computation 87, 78–128 (1990)
11. Gabaldon, A., Lakemeyer, G.: $\mathcal{ESP}$: A logic of only-knowing, noisy sensing and acting. In: Proc. of 22nd Natl. Conf. on Artificial Intelligence (AAAI-07). pp. 974–979. AAAI Press (2007)
12. Gass, S.I.: Linear programming : methods and applications. Dover Publications, fifth edn. (2010)
13. Halpern, J.Y.: Reasoning about Uncertainty. The MIT Press, Cambridge, MA (2003)
14. Hughes, G., Cresswell, M.: A New Introduction to Modal Logic. Routledge, New York, NY (1996)
15. Iocchi, L., Lukasiewicz, T., Nardi, D., Rosati, R.: Reasoning about actions with sensing under qualitative and probabilistic uncertainty. ACM Transactions on Computational Logic 10(1), 5:1–5:41 (2009)
16. Kooi, B.: Probabilistic dynamic epistemic logic. Journal of Logic, Language and Information 12(4), 381–408 (2003)
17. Monahan, G.E.: A survey of partially observable Markov decision processes: Theory, models, and algorithms. Management Science 28(1), 1–16 (1982)
18. Murty, K.G.: Linear programming. John Wiley and sons, revised edn. (1983)
19. Nilsson, N.: Probabilistic logic. Artificial Intelligence 28, 71–87 (1986)
20. Poole, D.: Decision theory, the situation calculus and conditional plans. Linköping Electronic Articles in Computer and Information Science 8(3) (1998)
21. Puterman, M.: Markov Decision Processes: Discrete Dynamic Programming. Wiley, New York, NY (1994)
22. Reiter, R.: Knowledge in action: logical foundations for specifying and implementing dynamical systems. MIT Press, Massachusetts/England (2001)
23. Rens, G., Lakemeyer, G., Meyer, T.: A logic for specifying agent actions and observations with probability. In: Kersting, K., Toussaint, M. (eds.) Proc. of 6th Starting AI Researchers' Symposium (STAIRS 2012). Frontiers in Artificial Intelligence and Applications, vol. 241, pp. 252–263. IOS Press (2012), url: http://www.booksonline.iospress.nl/Content/View.aspx?piid=31509
24. Rens, G., Meyer, T., Lakemeyer, G.: On the logical specification of probabilistic transition models. In: Proc. of 11th Intl. Symposium on Logical Formalizations of Commonsense Reasoning (COMMONSENSE 2013) (May 2013)
25. Rens, G., Meyer, T., Lakemeyer, G.: A sound and complete decision procedure for SLAOP. Tech. rep., Centre for Artificial Intelligence Research (University of KwaZulu-Natal, and CSIR Meraka), South Africa

(July 2013), url: http://www.cair.za.net/sites/default/files/outputs/The-SLAOP-decision-procedure.pdf
26. Rens, G., Varzinczak, I., Meyer, T., Ferrein, A.: A logic for reasoning about actions and explicit observations. In: Li, J. (ed.) AI 2010: Advs. in Artif. Intell. Proc. of 23rd Australasian Joint Conf. LNAI, vol. 6464, pp. 395–404. Springer Verlag, Berlin/Heidelberg (December 2010)
27. Rens, G., Meyer, T., Lakemeyer, G.: SLAP: Specification logic of actions with probability. Journal of Applied Logic (2013), www.sciencedirect.com/science/article/pii/S157086831300075X, url = http://dx.doi.org/10.1016/j.jal.2013.09.001
28. Sanner, S., Kersting, K.: Symbolic dynamic programming for first-order POMDPs. In: Proc. of 24th Natl. Conf. on Artificial Intelligence (AAAI-10). pp. 1140–1146. AAAI Press (2010)
29. Shirazi, A., Amir, E.: Probabilistic modal logic. In: Proc. of 22nd Natl. Conf. on Artificial Intelligence (AAAI-07). pp. 489–494. AAAI Press (2007)
30. Smallwood, R., Sondik, E.: The optimal control of partially observable Markov processes over a finite horizon. Operations Research 21, 1071–1088 (1973)
31. Tarski, A.: A decision method for elementary algebra and geometry. Tech. rep., The RAND Corporation, Santa Monica, Calif. (1957)
32. Thiébaux, S., Hertzberg, J., Schoaff, W., Schneider, M.: A stochastic model of actions and plans for anytime planning under uncertainty. International Journal of Intelligent Systems 10(2), 155–183 (1995)
33. Van Benthem, J., Gerbrandy, J., Kooi, B.: Dynamic update with probabilities. Studia Logica 93(1), 67–96 (2009)
34. Van Diggelen, J.: Using Modal Logic in Mobile Robots. Master's thesis, Cognitive Artificial Intelligence, Utrecht University (2002)
35. Wang, C., Schmolze, J.: Planning with POMDPs using a compact, logic-based representation. In: Proc. of 17th IEEE Intl. Conf. on Tools with Artif. Intell. (ICTAI'05). pp. 523–530. IEEE Computer Society, Los Alamitos, CA, USA (2005)