A Logic for Specifying Agent Actions and Observations with Probability[1]

Gavin RENS [a], Gerhard LAKEMEYER [b] and Thomas MEYER [a]
[a] *Centre for Artificial Intelligence Research, University of KwaZulu-Natal, South Africa; and CSIR Meraka, South Africa*
[b] *RWTH Aachen University, Informatik, Germany*

Abstract. We propose a non-standard modal logic for specifying agent domains where the agent's actuators and sensors are noisy, causing uncertainty in action and perception. The logic is multi-modal, indexed with actions; the logic is also augmented with *observation* objects to facilitate knowledge engineers dealing with explicit observations in the environment, and it includes a notion of probability. A tableau method is provided for proving decidability of the proposed logic. It is our conjecture that the tableau rules are complete with respect to the semantics. The proof does not yet exist, however, we discuss the current approach of the proof and provide some examples to motivate our conjecture.

Keywords. Logic, POMDP, stochastic actions and observations, domain specification, tableau method

1. Introduction and Motivation

In the physical real world, or in complex engineered systems, things are not black-and-white. We live in a world where there can be shades of truth and degrees of belief. Part of the problem is that agents' actuators and sensors are noisy, causing uncertainty in their action and perception. Agents inhabiting such complex and uncertain environments have to cope with the uncertainty. Thus we—agent designers—have to provide the agents with the coping mechanisms. We refer to the real worlds in which robots live, and man-made systems in which intelligent agents are deployed, as *stochastic domains*.

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'.

Logic-based artificial intelligence for agent reasoning is well established. In particular, a domain expert choosing to represent domains with a *logic* can take advantage of

[1] An earlier version of this logic has been presented at the Ninth International Workshop on Non-Monotonic Reasoning, Action and Change (NRAC'11) in Barcelona, Spain, 2011 [1]

the progress made in this sub-field of cognitive robotics [2] to specify the dynamics of stochastic domains.

Modal logic is considered to be well suited to reasoning about beliefs and changing situations [3,4,5,6]. Many popular frameworks for reasoning about action, employ or are based on the situation calculus [7]. 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 modal framework.

Partially observable Markov decision process (POMDP) theory [8,9,10,11,12] has proven to be a good general framework for formalizing dynamic stochastic systems. 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 for each agent action and the resulting state, a probability distribution over observations; and b^0 is the initial probability distribution over all states in S.

Our goal is to combine modal logic with POMDP theory so as to model agents as POMDPs, specifically for reasoning tasks in cognitive robotics. That goal-logic will be called *LUAP*. This paper though, concerns work that is a step towards LUAP. Here we present the Specification Logic of Actions and Observations with Probability (SLAOP), a 'sub-logic' of LUAP. SLAOP is a modal logic with actions and observations as first-class objects [13]. To establish a correspondence between POMDPs and SLAOP, SLAOP must view observations as objects at the same semantic level as actions. SLAOP can accommodate models of stochastic actions and observations via probabilities. The notion of *utility* (for rewards) can also be expressed in SLAOP. With SLAOP, POMDP states can be represented compactly, that is, an explicit enumeration of states is not required. To some extent with SLAOP, and more so with LUAP, one will be able to reason about aspects of POMDPs using theorem-proving tools (e.g., tableaux).

Whereas SLAOP is a language for *specifying* stochastic domains, LUAP will reason *with* the domain specification written with SLAOP. An engineer using LUAP will be able to specify POMDPs, including belief states; belief states cannot be specified with SLAOP. Our aim for the future will be to provide an algorithm for updating belief states. The belief update algorithm will be a core component of the proof system of LUAP, and proving validity of formulae in the syntax of SLAOP will be an important task in the belief update algorithm.

Although SLAOP uses probability theory, it is not for reasoning *about* probability; it is for reasoning about (probabilistic) actions and observations. There have been many approaches/frameworks for reasoning *about* probability, but most of them are either not concerned with *dynamic environments* [14,15,16,17] or they *are* concerned with change, but they are not actually *logics* [18,19,20,21]. Some probabilistic logics for reasoning about action and change do exist [22,23], but they are not modal and lack some desirable attributes, for example, decidability, a solution to the frame problem, non-deterministic actions, or catering for sensing. There are some logics that come closer to what we desire [24,25,26,27], that is, they are modal and 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 [28,29,30]. But again, they do not employ *modal* logic or they do not incorporate principals of cognitive robotics in a way that we would like to see in a representation/specification language.

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.

The domain is (partially) formalized as follows. The robot has the set of (intended) actions $\mathfrak{A} = \{grab, drink, weigh, replace\}$ with expected meanings. The robot can perceive observations only from the set $\Omega = \{obsNil,\ obsLight,\ obsMedium,\ obsHeavy\}$. 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: $\mathfrak{P} = \{full, drank, holding\}$ 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:

- Is it so that the probability of perceiving that the oil-can is light is 0.7 when the can is not full, and I have drunk the oil, and I am holding the can? That is, does $[obsLight \mid weigh]_{0.7}(\neg full \wedge drank \wedge holding)$ 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 $(\neg full \wedge \neg holding) \rightarrow ([grab]_{0.9}(\neg full \wedge holding) \wedge [grab]_{0.1}(\neg full \wedge \neg holding))$ follow from *BK*?

In a previous paper [1], we introduced SLAOP and showed how one can specify a stochastic domain by using the language of SLAOP, with the 'oil-can scenario' as a running example. In this paper, we present some of the work done towards proving that SLAOP is decidable, which would set it apart from first-order logics for reasoning about action (including the situation calculus [7]) or reasoning with probabilities (including $\mathcal{ESP}$ [26]). 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.

Section 2 presents the syntax and semantics of SLAOP. Section 3 presents the tableau method and Section 4 provides examples of application of the tableau method. Some concluding remarks are made in Section 5.

2. Specification Logic of Actions and Observations with Probability

2.1. Syntax

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

1. a finite set of *propositional variables* (simply, *propositions*) $\mathfrak{P} = \{p_1, \ldots, p_n\}$,

2. a finite set of names of atomic *actions* $\mathfrak{A} = \{\alpha_1, \ldots, \alpha_n\}$,
3. a finite set of names of atomic *observations* $\Omega = \{\varsigma_1, \ldots, \varsigma_n\}$,
4. a countable set of names $\mathfrak{Q} = \{q_1, q_2, \ldots\}$ of all *rational numbers* in $\mathbb{Q}$.

The setting is multi-modal, in which we have modal operators $[\alpha]$, one for each $\alpha \in \mathfrak{A}$, and modal operators $[\varsigma \mid \alpha]_q$, one for each pair in $\Omega \times \mathfrak{A}$.

Definition 2.1 *Let $\alpha, \alpha' \in \mathfrak{A}$, $\varsigma, \varsigma' \in \Omega$, $q, r, c \in \mathfrak{Q}$ and $p \in \mathfrak{P}$. The language of SLAOP, denoted $\mathscr{L}_{SLAOP}$, is the least set of Φ defined by the grammar:*

$$\varphi ::= p \mid \top \mid \neg\varphi \mid \varphi \wedge \varphi.$$

$$\Phi ::= \varphi \mid \neg\Phi \mid \Phi \wedge \Phi \mid [\alpha]_q\varphi \mid [\varsigma \mid \alpha]_q\varphi \mid \alpha = \alpha' \mid \varsigma = \varsigma' \mid Reward(r) \mid Cost(\alpha, c).$$

$[\alpha]_q\varphi$ is read 'The probability of reaching a world in which φ holds after executing α, is equal to q'. $[\alpha]$ abbreviates $[\alpha]_1$ and $\langle\alpha\rangle\varphi$ abbreviates $\neg[\alpha]\neg\varphi$. $[\varsigma \mid \alpha]_q\varphi$ can be read 'The probability of perceiving ς in a world in which φ holds is equal to q, given α was performed'. $\neg[\varsigma \mid \alpha]_0\varphi$ can be written as $\langle\varsigma \mid \alpha\rangle\varphi$ and is read 'It is possible to perceive ς in a φ-world, given α was performed'. We may write $[\varsigma \mid \alpha]\varphi$ instead of $[\varsigma \mid \alpha]_1\varphi$.

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. $Reward(r)$ can be read 'The reward for being in the current situation is r units' and we read $Cost(\alpha, c)$ as 'The cost for executing α is c units'.

Note that formulae with nested modal operators, like $[\cdot]_q[\cdot]_q\varphi$, $[\cdot]_q[\cdot]_q[\cdot]_q\varphi$, et cetera, are not in $\mathscr{L}_{SLAOP}$. 'Single-step' or 'flat' formulae are sufficient to *specify* transition and perception probabilities. The logic called LUAP, to be defined in future, will allow an agent to query the probability of some propositional formula φ after an *arbitrary* sequence of actions and observations. As usual, we treat $\bot, \vee, \rightarrow$ and $\leftrightarrow$ as abbreviations.

2.2. Semantics

Our semantics follows that of multi-modal logic **K** [31]. However, structures (alias, possible worlds models [32,33]) are non-standard. Standard modal logic structures 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 [34,3, e.g.]. We shall say that modal logics—and their extensions—with such standard structures, have *point-based* semantics.

As mentioned in Section 1, the development of SLAOP is to provide a logic that can represent POMDPs for cognitive robotics. The addition of observation objects is one step towards this goal; another important step is to set up a correspondence between states in POMDP theory and worlds in the logic. That is, given a specification that uniquely identifies a state, there should be a uniquely identifiable world in the structure of the logic. 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. Hence, SLAOP does not have a point-based semantics: 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. Let $w \in W$ and let $w : \mathfrak{P} \mapsto \{0,1\}$ be a total function that assigns a truth value to each proposition. Let C (*conceivable worlds*) be the set of all possible functions w. We shall say that modal logics—and their extensions—with such structures, have *world-based* semantics.

Definition 2.2 *A SLAOP structure is a tuple $\mathfrak{S} = \langle W, R, O, N, Q, U\rangle$ such that*

1. *$W \subset C$ a non-empty set of* possible worlds*;*
2. *R is a mapping that provides an accessibility relation $R_\alpha : W \times W \times \mathbb{Q} \cap [0,1]$ for each action $\alpha \in \mathfrak{A}$; Given some $w^- \in W$ as the first component of a triple in R_α, we require that $\sum_{(w^-,w^+,pr)\in R_\alpha} pr = 1$; Also, if $(w^-,w^+,pr),(w^-,w^+,pr') \in R_\alpha$, then $pr = pr'$;*
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 is a mapping that provides a perceivability relation $Q_\alpha : O \times W \times \mathbb{Q} \cap [0,1]$ for each action $\alpha \in \mathfrak{A}$; Given some $w^+ \in W$ such that $(w^-,w^+,pr') \in R_\alpha$, it is required that $\sum_{(o,w^+,pr)\in Q_\alpha} pr = 1$; Also, if $(o,w^+,pr),(o,w^+,pr') \in Q_\alpha$, then $pr = pr'$;*
6. *U is a pair $\langle Re, Co\rangle$, where $Re : W \mapsto \mathbb{Q}$ is a reward function and Co is a mapping that provides a cost function $Co_\alpha : C \mapsto \mathbb{Q}$ for each $\alpha \in \mathfrak{A}$;*

R_α defines which worlds w^+ are accessible via action α performed in world w^- and the transition probability $pr \in \mathbb{Q} \cap [0,1]$. Q_α defines which observations o are perceivable in worlds w^+ accessible via action α and the observation probability $pr \in \mathbb{Q} \cap [0,1]$.

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 rational 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 rational number representing the cost of executing α in the world w. It must be defined for each action $\alpha \in \mathfrak{A}$ and each $w \in C$.

Definition 2.3 (Truth Conditions) *Let $\mathfrak{S}$ be a SLAOP structure, with $\alpha, \alpha' \in \mathfrak{A}$, $\varsigma, \varsigma' \in \Omega$, $q, r, c \in \mathfrak{Q}$. Let $p \in \mathfrak{P}$ and let φ be any sentence in $\mathscr{L}_{SLAOP}$. We say φ is* satisfied *at world w in structure $\mathfrak{S}$ (written $\mathfrak{S}, w \models \varphi$) if and only if the following holds:*

1. $\mathfrak{S}, w \models \top$ for all $w \in W$*;*
2. $\mathfrak{S}, w \models p \iff w(p) = 1$ for $w \in W$*;*
3. $\mathfrak{S}, w \models \neg\varphi \iff \mathfrak{S}, w \not\models \varphi$*;*
4. $\mathfrak{S}, w \models \varphi \wedge \varphi' \iff \mathfrak{S}, w \models \varphi$ and $\mathfrak{S}, w \models \varphi'$*;*
5. $\mathfrak{S}, w \models \alpha = \alpha' \iff \alpha, \alpha' \in \mathfrak{A}$ are the same element*;*
6. $\mathfrak{S}, w \models \varsigma = \varsigma' \iff \varsigma, \varsigma' \in \Omega$ are the same element*;*
7. $\mathfrak{S}, w \models Reward(r) \iff Re(w) = r$*;*
8. $\mathfrak{S}, w \models Cost(\alpha, c) \iff Co_\alpha(w) = c$*;*
9. $\mathfrak{S}, w \models [\varsigma \mid \alpha]_q \varphi \iff (\forall w')$ if $(\exists pr)(w, w', pr) \in R_\alpha$ and $\mathfrak{S}, w' \models \varphi$ then $(N(\varsigma), w', q) \in Q_\alpha$*;*
10. $\mathfrak{S}, w \models [\alpha]_q \varphi \iff \left(\sum_{(w,w',pr)\in R_\alpha, \mathfrak{S}, w' \models \varphi} pr\right) = q$.

There should always be *some* observation (associated with an action) in a world, given that action was performed to reach that world. If this were not so, an agent could reach a world and become 'unconscious' due to having no observations. Conversely, notice that if there is an observation in a world, there must have been an action that caused the agent to be there. Therefore, it is required that the following set of axioms be stated in any and all domain specifications employing SLAOP: $\{\langle\alpha\rangle\varphi \leftrightarrow \bigvee_{\varsigma\in\Omega}\langle\varsigma \mid \alpha\rangle\varphi \mid \alpha \in \mathfrak{A}\}$.

A formula φ is *valid* in a SLAOP structure (denoted $\mathfrak{S} \models \varphi$) if $\mathfrak{S}, w \models \varphi$ for every $w \in W$. φ is *SLAOP-valid* (denoted $\models \varphi$) if φ is true in every structure $\mathfrak{S}$. φ is *satisfiable* if $\mathfrak{S}, w \models \varphi$ for some $\mathfrak{S}$ and $w \in W$. The truth of a propositional formula is independent of a SLAOP structure. We may thus write $w \models \varphi$ instead of $\mathfrak{S}, w \models \varphi$ when φ is a propositional formula.

Let $\mathscr{K} \subset \mathscr{L}_{SLAOP}$. If, for all $\theta \in \mathscr{K}$, $\models \psi$ whenever $\models \theta$, we say $\mathscr{K}$ *logically entails* ψ (abbreviated $\mathscr{K} \models \psi$). If $\mathscr{K}$ logically entails ψ and $\mathscr{K}$ contains a single sentence θ, then we omit the brackets and write $\theta \models \psi$. If $\models \theta \leftrightarrow \psi$, we say θ and ψ are *logically equivalent* (abbreviated $\theta \equiv \psi$).

3. The Tableau Method

In modal logics, tableau calculi are well suited as decision procedures for validity. If we could design a tableau method and prove that it is sound and complete with respect to the semantics and prove that the tableau method always terminates, then as a consequence, SLAOP would be decidable. The tableau method we propose is adapted from Castilho, Gasquet and Herzig [35]. It is based on a labeled formulae calculus. The necessary terminology is given next.

The tableau calculus for SLAOP, with all its rules, is referred to as $\mathscr{C}_{SLAOP}$.

A *labeled formula* is a pair (n, φ), where φ is a formula and n is an integer called the *label* of φ. A *skeleton* Σ is a binary relation $\Sigma \subseteq \mathfrak{A} \times \mathbb{N}$. Elements (α, n') of the relation are denoted $\stackrel{\alpha}{\to} n'$. A *node* N^j_k is a pair $\langle \Gamma^j_k, \Sigma^j_k \rangle$ with superscript j the *branch* index and subscript k the *node* index, where Γ^j_k is a set of labeled formulae and Σ^j_k is a skeleton. The initial node to which $\mathscr{C}_{SLAOP}$ must be applied, that is, N^0_0, is called the *trunk*. A *tree* T is a set of nodes. A tree must include N^0_0 and only nodes resulting from the application of *tableau rules* to the trunk and subsequent nodes. A *branch* $B^j(T)$ of some tree T is the set of nodes with the same *branch* index j: $B^j(T) = \{N^j_k \in T \mid k = 0, 1, 2, \ldots\}$.

When we say '...where x is a fresh integer', we mean that x is the smallest positive integer not yet used (for a label, branch index or node index, as the case may be) in the current tree.

A tableau rule applied to node N^j_k creates one or more new nodes; its child(ren). If it creates one child, then it is identified as N^j_{k+1}. If N^j_k creates a second child, it is identified as $N^{j'}_0$, where j' is a fresh integer. That is, for every child created beyond the first, a new branch is started. Node N^j_k is a *leaf* node of tree T if there is no node $N^j_{k'}$ in branch $B^j(T)$ such that $k' > k$. A node $\langle \Gamma, \Sigma \rangle$ is *closed* if $(i, \bot) \in \Gamma$ for some i. 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 $B^0(T), \ldots, B^n(T)$ are closed, else it is open.

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

- A rule may only be applied to an open leaf node.

- rule $\bot$: If Γ_k^j contains (n,Φ) and $(n,\neg\Phi)$, then create node $\langle\Gamma_k^j \cup \{(n,\bot)\},\Sigma_k^j\rangle$.
- rule $\neg$: If Γ_k^j contains (n,Φ), where Φ contains $\neg\neg$, then create node $\langle\Gamma_k^j \cup \{(n,\Phi')\},\Sigma_k^j\rangle$, where Φ' is Φ without $\neg\neg$.
- rule $\wedge$: If Γ_k^j contains $(n,\Phi\wedge\Phi')$, then create node $\langle\Gamma_k^j \cup \{(n,\Phi),(n,\Phi')\}\},\Sigma_k^j\rangle$.
- rule $\vee$: If Γ_k^j contains $(n,\neg(\Phi\wedge\Phi'))$, then create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(n,\neg\Phi)\},\Sigma_k^j\rangle$ and node $N_0^{j'} = \langle\Gamma_k^j \cup \{(n,\neg\Phi')\},\Sigma_k^j\rangle$, where j' is a fresh integer.
- rule $=$: If Γ_k^j contains $(0,c=c')$ and in fact constants c and c' do not refer to the same constant, or if Γ_k^j contains $(0,c\neq c')$ and in fact constants c and c' do refer to the same constant, then create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\bot)\},\Sigma_k^j\rangle$.
- rule *Re*: If Γ_k^j contains $(0,Reward(r))$ and $(0,Reward(r'))$ such that $r\neq r'$, then create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\bot)\},\Sigma_k^j\rangle$.
- rule *Co*: If Γ_k^j contains $(0,Cost(\alpha,c))$ and $(0,Cost(\alpha,c'))$ such that $c\neq c'$, then create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\bot)\},\Sigma_k^j\rangle$.
- rule $\neg[\varsigma]$: If Γ_k^j contains $(0,[\varsigma \mid \alpha]_q\Phi)$ and $(0,\neg[\varsigma \mid \alpha]_q\Phi')$ where $\Phi\wedge\Phi'\not\equiv\bot$, create node $\langle\Gamma_k^j \cup \{(0,\bot)\},\Sigma_k^j\rangle$.
- rule $[\varsigma]_q$: If Γ_k^j contains $(0,[\varsigma \mid \alpha]_q\Phi)$ and $(0,[\varsigma' \mid \alpha]_{q'}\Phi')$ where $\Phi\wedge\Phi'\not\equiv\bot$, then

  1. if $q\neq q'$, create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\neg(\varsigma=\varsigma'))\},\Sigma_k^j\rangle$.
  2. if $q+q'>1$, create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\varsigma=\varsigma')\},\Sigma_k^j\rangle$.

- rule *obs*: If Γ_k^j contains $(0,[\varsigma_1 \mid \alpha]_{q_1}\Phi_1),(0,[\varsigma_2 \mid \alpha]_{q_2}\Phi_2),\ldots,(0,[\varsigma_m \mid \alpha]_{q_m}\Phi_m)$ such that ς_x is not the same as ς_y for all x and y ($1\leq x,y\leq m$ and $x\neq y$) and $\bigwedge_{i=1}^m\Phi_i\not\equiv\bot$, then

  1. if $\sum_{i=1}^m q_i = 1$, then create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,[\varsigma_1' \mid \alpha]_0\Phi'),(0,[\varsigma_2' \mid \alpha]_0\Phi'),\ldots,(0,[\varsigma_m' \mid \alpha]_0\Phi')\},\Sigma_k^j\rangle$, where $\varsigma'\in\Omega\setminus\{\Phi_z\in\Omega \mid z=1,2,\ldots,m\}$ and Φ' is $\bigwedge_{i=1}^m\Phi_i$.
  2. if $\sum_{i=1}^m q_i<1$, then create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\neg[\varsigma_1' \mid \alpha]_0\Phi'\vee\neg[\varsigma_2' \mid \alpha]_0\Phi')\vee\cdots\vee\neg[\varsigma_m' \mid \alpha]_0\Phi')\},\Sigma_k^j\rangle$, where $\varsigma'\in\Omega\setminus\{\Phi_z\in\Omega \mid z=1,2,\ldots,m\}$ and Φ' is $\bigwedge_{i=1}^m\Phi_i$.
  3. if $\bigcup_{i=1}^n\varsigma_i=\Omega$, then if $\sum_{i=1}^m q_i\neq 1$, create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\bot)\},\Sigma_k^j\rangle$.

- rule $\rightarrow\langle\alpha\rangle$: If Γ_k^j contains $(0,[\alpha]_q\Phi)$ for $0<q\leq 1$, then create node $\langle\Gamma_k^j \cup \{(0,\neg[\alpha]\neg\Phi)\},\Sigma_k^j\rangle$.
- rule $\rightarrow\langle\varsigma\rangle$: If Γ_k^j contains $(0,[\varsigma \mid \alpha]_q\Phi)$ for $0<q\leq 1$, then create node $N_{k+1}^j = \langle\Gamma_k^j \cup \{(0,\neg[\varsigma \mid \alpha]_0\Phi)\},\Sigma_k^j\rangle$.
- rule $\Diamond$: If Γ_k^j contains $(0,\neg[\alpha]\Phi)$, then create node $\langle\Gamma_k^j \cup \{(n,\neg\Phi)\},\Sigma_k^j \cup \{0 \xrightarrow{\alpha} n\}\rangle$, where n is a fresh integer.
- rule $\Box$: If Γ_k^j contains $(0,[\alpha]\Phi)$ and Σ contains $\xrightarrow{\alpha} n$, then create node $\langle\Gamma_k^j \cup \{(n,\Phi)\},\Sigma_k^j\rangle$.
- rule *rng*: If Γ_k^j contains $(0,[\alpha]_q\Phi)$ such that $q<0$ or $q>1$, then create node $\langle\Gamma_k^j \cup \{(0,\bot)\},\Sigma_k^j\rangle$.

- rule $1-q$: If Γ_k^j contains $(0,[\alpha]_q\Phi)$, then create node $\langle\Gamma_k^j\cup\{(0,[\alpha]_{1-q}\neg\Phi)\},\Sigma_k^j\rangle$.
- rule $\neg[\alpha]$: If Γ_k^j contains $(0,[\alpha]_q\Phi)$ and $(0,\neg[\alpha]_q\Phi')$ where $q<1$, create node $\langle\Gamma_k^j\cup\{(0,\neg[\alpha](\Phi\leftrightarrow\Phi'))\},\Sigma_k^j\rangle$.
- rule $[\alpha]_q$: If Γ_k^j contains $(0,[\alpha]_q\Phi)$ and $(0,[\alpha]_{q'}\Phi')$, then

  1. create node $\langle\Gamma_k^j\cup\{(0,[\alpha]\neg(\Phi\wedge\Phi')\to[\alpha]_{q+q'}(\Phi\vee\Phi'))\},\Sigma_k^j\rangle$.
  2. create node $\langle\Gamma_k^j\cup\{(0,[\alpha](\Phi\to\Phi')\to[\alpha]_{q'-q}(\Phi'\wedge\neg\Phi))\},\Sigma_k^j\rangle$.
  3. if $q>q'$, create node $\langle\Gamma_k^j\cup\{(0,\langle\alpha\rangle(\Phi\wedge\neg\Phi'))\},\Sigma_k^j\rangle$.

- rule *dne*: If Γ_k^j contains $(0,[\alpha]_q\Phi)$, $(0,[\alpha]_{q'}\Phi')$ and $(0,[\alpha]_{q''}\Phi'')$, then

  1. if $\Phi''\equiv\Phi\wedge\Phi'$, then: if $0\leq q+q'-q''\leq 1$, create node $\langle\Gamma_k^j\cup\{(0,[\alpha]_{q+q'-q''}\neg(\neg\Phi\wedge\neg\Phi'))\},\Sigma_k^j\rangle$, else create node $\langle\Gamma_k^j\cup\{(0,\perp)\},\Sigma_k^j\rangle$.
  2. if $\Phi''\equiv\Phi\vee\Phi'$, then: if $0\leq q+q'-q''\leq 1$, create node $\langle\Gamma_k^j\cup\{(0,[\alpha]_{q+q'-q''}(\Phi\wedge\Phi'))\},\Sigma_k^j\rangle$, else create node $\langle\Gamma_k^j\cup\{(0,\perp)\},\Sigma_k^j\rangle$.
  3. if $\Phi'\equiv\Phi\wedge\chi$ and $\Phi''\equiv\Phi\vee\chi$ for some $\chi\not\equiv\Phi$, then: if $0\leq q'+q''-q\leq 1$, create node $\langle\Gamma_k^j\cup\{(0,[\alpha]_{q'+q''-q}\chi)\},\Sigma_k^j\rangle$, else create node $\langle\Gamma_k^j\cup\{(0,\perp)\},\Sigma_k^j\rangle$.

If one has a tree with trunk $N_0^0=\langle\{(0,\Psi)\},\emptyset\rangle$, we'll say one has *a tree for* Ψ. Note that $(n,\Phi)\notin\Gamma$ for $n>0$ when Φ is a dynamic formula. Hence, in tableau rules explicitly concerning dynamic formulae, the labeled formula 'triggering' the rule has label 0.

Remark 3.1 *For rule Rl applicable to any labeled formula* (n,Φ)*, if the rule says to create a new node* $\langle\Gamma\cup F,\Sigma\rangle$ *while F is already in* Γ*, then Rl may not be applied to* (n,Φ)*. Also, if rule* $\Diamond$ *has been applied to* $(0,\Diamond\Phi)$*, don't apply it to* $(0,\Diamond\Phi)$ *again.*

The above remark constrains rule application to prevent trivial re-applications of rules.

A branch is *saturated* if and only if any rule that can be applied to its leaf node has been applied. If a tree for $\neg\Psi$ is closed, we write $\vdash\Psi$. If there is a tree for $\neg\Psi$, with a saturated open branch, we write $\not\vdash\Psi$.

Theorem 3.1 (Soundness) *If* $\vdash\Psi$ *then* $\models\Psi$.

We proved soundness; the proof is omitted here. We conjecture that $\mathscr{C}_{SLAOP}$ always terminates, and although it does not seem difficult to prove, it is a work in progress. However, the proof of completeness is difficult: it requires that a SLAOP structure be constructed from the information in a tableau tree whenever the tree indicates that a model exists for the input sentence—while the SLAOP structure must adhere to probability theory, given the notions of probability expressed in the input sentence.

Conjecture 3.1 (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 tree in a saturated tableau for ψ. It thus suffices to construct for any open saturated tree for $\psi\in\mathscr{L}_{SLAOP}$, a SLAOP structure $\mathfrak{S}$ in which there is a world w in $\mathfrak{S}$ such that ψ is true in $\mathfrak{S}$ at w.

4. Examples

This section includes three examples of $\mathscr{C}_{SLAOP}$ at work, all involving our oil-can scenario. Limited space prevents us from providing a full specification of the scenario. We assume that the full domain specification is contained by the agent's background knowledge BK. In particular, the following domain axioms, which are required in the example proofs below, are in BK.

- $[obsLight \mid weigh]_{0.7}(\neg full \wedge drank)$ gives the probability of weighing the oil-can and finding that it is light in worlds where the can is not full and the oil has been drunk,
- $(full \wedge \neg drank \wedge holding) \rightarrow ([drink]_{0.85}(\neg full \wedge drank \wedge holding) \wedge [drink]_{0.15}(\neg full \wedge \neg drank \wedge holding))$ gives the probabilities of reaching the only two worlds reachable from the world where *full* and *holding* are true and *drank* is false.
- $holding \rightarrow [drink]holding$ expresses that the agent doesn't drop the oil-can when drinking,
- $(full \wedge \neg holding) \rightarrow ([grab]_{0.7}(full \wedge holding) \wedge [grab]_{0.2}(\neg full \wedge \neg holding) \wedge [grab]_{0.1}(full \wedge \neg holding))$ is another specification of transition probabilities given the can is full and the agent is not holding it when it grabs it,
- $(full \wedge drank \wedge \neg holding) \rightarrow [grab]\ drank$ expresses the agent's belief that if it has drunk the oil, then if it grabs the can, the agent will still think it has drunk the oil.

Please refer to our previous paper [1] for an explanation of domain specification using SLAOP.

In these examples, it will be determined whether a sentence $IC \rightarrow \varphi$ is logically entailed by BK, where φ is an arbitrary sentence of interest and IC is the agent's initial condition.

Tables 1, 2 and 3 depict the tableaux of the different examples. To shorten and clarify the proofs, we shall use syntactic abbreviations, and we shall not show every rule application, as long as the steps remain clear. The 'Comment' column mentions the rule applied and the numbers in the 'Comment' column refer to the line to which the rule was applied. That is, "rl. $R\ell$:x" means that rule $R\ell$ was applied to a formula in line x. Also, in the 'Comment' column, "bk." indicates that the sentence in that line is from BK.

Standard logical equivalences will be used to transform formulae into more 'normal' forms: "nf.: x" in the 'Comment' column means that 'normal forming' was applied to line x. If there is not enough space in the 'Comment' column, the comment will be written just adjacent to the applicable node.

Furthermore, the following abbreviations for constants will be used: $grab := g$, $drink := d$, $weigh := w$, $full := f$, $drank := d$, $holding := h$, and $obsLight := oL$.

Table 1. Proof that $BK \models (full \wedge drank) \rightarrow \neg[obsLight \mid weight]_{0.1}\neg full$.

Line	Γ & Σ	Comment
1	$(0, f \wedge d)$, $(0, \neg\neg[oL \mid w]_{0.1}\neg f)$	trunk
2	$(0, f \wedge d)$, $(0, [oL \mid w]_{0.1}\neg f)$	rl.$\neg$:1
3	$(0, [oL \mid w]_{0.7}(\neg f \wedge d))$	bk.
4	$(0, \neg(oL = oL)$	rl.$[\varsigma]_q$1:2,3
5	$(0, \bot)$	rl.=:4

In the proof given in Table 1, the agent's initial condition is expressed as $(0, f \wedge d)$. Note however, that $BK \models \iota \rightarrow \neg[obsLight \mid weight]_{0.1}\neg full$ for any initial condition ι. This is because observation probabilities depend on the action executed and the world reached, not the world in which the action was executed.

Table 2. Proof that $BK \models (full \wedge \neg drank \wedge holding) \rightarrow [drink]_{0.15}(full \vee \neg drank)$.

Line	Γ & Σ					Comment
1	$(0, f \wedge \neg d \wedge h)$, $(0, \neg[d]_{0.15}(f \vee \neg d))$					trunk
2	$(0, f)$, $(0, \neg d)$, $(0, h)$, $(0, \neg[d]_{0.15}(f \vee \neg d))$					rl.$\wedge$:1
3	$(0, (f \wedge \neg d \wedge h) \rightarrow ([d]_{0.85}(\neg f \wedge d \wedge h) \wedge [d]_{0.15}(\neg f \wedge \neg d \wedge h)))$					bk.
4	$(0, \neg(f \wedge \neg d \wedge h) \vee ([d]_{0.85}(\neg f \wedge d \wedge h) \wedge [d]_{0.15}(\neg f \wedge \neg d \wedge h)))$					nf.:3
5	$(0, \neg f)$	$(0, d)$	$(0, \neg h)$	$(0, [d]_{0.85}(\neg f \wedge d \wedge h))$, $(0, [d]_{0.15}(\ldots))$		rl.$\vee$,$\neg$,$\wedge$:4
6	$(0, \bot)$	$(0, \bot)$	$(0, \bot)$	$(0, [d]_{0.15}\neg(\neg f \wedge d \wedge h))$		rl.$1-q$:5
7	rl.$\bot$:2,5	rl.$\bot$:2,5	rl.$\bot$:2,5	$(0, \neg[d](\neg(\neg f \wedge d \wedge h) \leftrightarrow (f \vee \neg d))$		rl.$\neg[\alpha]$:1,6
8				$(1, \neg f \wedge d \wedge h)$, $(1, f \vee \neg d)$, $\xrightarrow{d} 1$	$(1, f \vee \neg d \vee \neg h)$, $(1, \neg f \wedge d)$, $\xrightarrow{d} 1$	nf.& rl.$\Diamond$:7
9					$(0, h \rightarrow [d]h)$	bk.
				$\vdots$	$\vdots$	
				$(1, \bot)$, rl.$\bot$:8	$(1, \bot)$	rl.$\bot$:8,9

Table 3. Proof that $BK \models (full \wedge drank \wedge \neg holding) \rightarrow [grab]_{0.7}(full \wedge drank \wedge holding)$.

Line	Γ & Σ						Comment
1	$(0, f \wedge d \wedge \neg h)$, $(0, \neg[g]_{0.7}(f \wedge d \wedge h))$						trunk
2	$(0, f)$, $(0, d)$, $(0, \neg h)$, $(0, \neg[g]_{0.7}(f \wedge d \wedge h))$						rl.$\wedge$:1
3	$(0, (f \wedge \neg h) \rightarrow ([g]_{0.7}(f \wedge h) \wedge [g]_{0.2}(\neg f \wedge \neg h) \wedge [g]_{0.1}(f \wedge \neg h)))$						bk.
4	$(0, \neg f)$	$(0, h)$	$(0, [g]_{0.7}(f \wedge h) \wedge [g]_{0.2}(\neg f \wedge \neg h) \wedge [g]_{0.1}(f \wedge \neg h)))$				nf.& rl.$\vee$,$\neg$:3
5	$(0, \bot)$	$(0, \bot)$	$(0, (f \wedge d \wedge \neg h) \rightarrow [g]\ d)$				bk.
6	rl.$\bot$:2,4	rl.$\bot$:2,4	$(0, \neg f)$	$(0, \neg d)$	$(0, h)$	$(0, [g]\ d)$	nf.& rl.$\vee$,$\neg$:5
7			$(0, \bot)$	$(0, \bot)$	$(0, \bot)$	$(0, [g]_{0.7}(f \wedge h))$	rl.$\wedge$:4
8			rl.$\bot$:2,6	rl.$\bot$:2,6	rl.$\bot$:2,6	continues in table below	

Line	Γ & Σ				Comment
	$\vdots$				
9	$(0, \neg[g]((f \wedge d \wedge h) \leftrightarrow (f \wedge h))$				rl.$\neg[\alpha]$:1,7
10	$(1, \neg((f \wedge d \wedge h) \leftrightarrow (f \wedge h))$, $\xrightarrow{g} 1$				rl.$\Diamond$:9
11	$(1, \neg(f \wedge d \wedge h) \wedge (f \wedge h)$			$(1, (f \wedge d \wedge h) \wedge \neg(f \wedge h)$	rl.$\wedge$,$\vee$:10
12	$(1, \neg f \vee \neg d \vee \neg h)$, $(1, f)$, $(1, h)$			$(1, f)$, $(1, d)$, $(1, h)$, $(1, \neg f \vee \neg h)$	rl.$\wedge$,$\vee$:11
13	$(1, \neg f)$, $(1, f)$	$(1, \neg d)$, $(1, f)$, $(1, h)$	$(1, \neg h)$, $(1, h)$	$\vdots$	rl.$\vee$:12
14	$(1, \bot)$	$(1, \neg d)$, $(1, f)$, $(1, h)$, $(1, d)$	$(1, \bot)$	$(1, \bot)$	rl.$\Box$:6
15	rl.$\bot$:13	$(1, \bot)$	rl.$\bot$:13	rl.$\bot$:12	rl.$\bot$:14

5. Concluding Remarks

We introduced a formal language for specifying partially observable Markov decision processes (POMDPs), specifically for robots that must deal with uncertainty in affection and perceptions. The formal language is based on multi-modal logic and accepts basic principals of cognitive robotics. We have also included notions of probability to represent the uncertainty to represent POMDPs for the intended application. Beyond the usual elements of logics for reasoning about action and change, the logic presented here adds *observations* as first-class objects, and a means to represent *utility functions*. An approach to specifying a robot and its environment was laid out elsewhere [1].

Our research thus far has shown that SLAOP's tableau method is sound. Ultimately, we want to prove that the method is decidable, however, this will depend on whether it is complete and terminating. Proving completeness is difficult and has not yet been achieved. Our approach for the completeness proof is via a tableau method for deciding the validity of sentences. Proofs of validity, like those in the previous section, supports our intuition that SLAOP is complete. A secondary purpose for designing a tableau method is as a starting point for designing an implementation of SLAOP.

Acknowledgement

Part of this research was done while the first author was in Germany on a DAAD (German Academic Exchange Service) scholarship.

References

[1] G. Rens, T. Meyer, A. Ferrein, and G. Lakemeyer. A logic for specifying partially observable stochastic domains. In S. Sardina and S. Vassos, editors, *Proceedings of the Ninth International Workshop on Non-Monotonic Reasoning, Action and Change (NRAC'11)*, pages 15–22, Melbourne 3000, Australia, July 2011. School of Computer Science and Information Technology, RMIT University.

[2] H. J. Levesque and G. Lakemeyer. Cognitive Robotics. In B. Porter F. Van Harmelen, V. Lifshitz, editor, *The Handbook of Knowledge Representation*, pages 869–886. Elsevier Science, 2008.

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

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

[5] P. Blackburn, M. De Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, UK, 2001.

[6] P. Blackburn, J. Van Benthem, and F. Wolter, editors. *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*. Elsevier, Amsterdam, The Netherlands / Oxford, UK, 2007.

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

[8] K. Aström. Optimal control of Markov decision processes with incomplete state estimation. *J. Math. Anal. Appl.*, 10:174–205, 1965.

[9] R. Smallwood and E. Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973.

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

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

[12] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *J. Artif. Intell. Res. (JAIR)*, 11:1–94, 1999.

[13] G. Rens, I. Varzinczak, T. Meyer, and A. Ferrein. A logic for reasoning about actions and explicit observations. In Jiuyong Li, editor, *AI 2010: Advances in Artificial Intelligence. Proceedings of the 23rd Australasian Joint Conference*, volume 6464 of *Lecture Notes in Artificial Intelligence*, pages 395–404, Berlin/Heidelberg, December 2010. Springer-Verlag.

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

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

[16] J. Y. Halpern. *Reasoning about Uncertainty*. The MIT Press, Cambridge, MA, 2003.

[17] A. Shirazi and E. Amir. Probabilistic modal logic. In *Proc. of 22nd Natl. Conf. on Artificial Intelligence (AAAI-07)*, pages 489–494. AAAI Press, 2007.

[18] D. Poole. Decision theory, the situation calculus and conditional plans. *Linköping Electronic Articles in Computer and Information Science*, 8(3), 1998.

[19] C. Boutilier, R. Reiter, M. Soutchanski, and 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)*, pages 355–362. AAAI Press, Menlo Park, CA, 2000.

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

[21] 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.

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

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

[24] M. De Weerdt, F. De Boer, W. Van der Hoek, and J.-J. Meyer. Imprecise observations of mobile robots specified by a modal logic. In *Proc. of ASCI-99*, pages 184–190, 1999.

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

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

[27] J. Van Benthem, J. Gerbrandy, and B. Kooi. Dynamic update with probabilities. *Studia Logica*, 93(1):67–96, 2009.

[28] C. Boutilier and D. Poole. Computing optimal policies for partially observable decision processes using compact representations. In *Proc. of 13th Natl. Conf. on Artificial Intelligence*, pages 1168–1175, 1996.

[29] C. Wang and J. Schmolze. Planning with POMDPs using a compact, logic-based representation. In *Proc. of 17th IEEE Intl. Conf. on Tools with Artif. Intell. (ICTAI'05)*, pages 523–530, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[30] S. Sanner and K. Kersting. Symbolic dynamic programming for first-order POMDPs. In *Proc. of 24th Natl. Conf. on Artificial Intelligence (AAAI-10)*, pages 1140–1146. AAAI Press, 2010.

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

[32] S. Kripke. A completeness theorem in modal logic. *Journal of Symbolic Logic*, 24(1):1–14, 1959.

[33] J. K. K. Hintikka. *Knowledge and belief*. Cornell University Press, Ithaca, NY, 2nd edition, 1962.

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

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