A Logic for Specifying Agent Actions
and Observations with Probability

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 ac-
tions 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 rep-
resent 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 partic-
ular, 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; \( \Omega \) 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 \(A = \{\text{grab, drink, weigh, replace}\}\) with expected meanings. The robot can per-
ceive observations only from the set \(\Omega = \{\text{obsNil, obsLight, obsMedium, obsHeavy}\}\).
Intuitively, when the robot performs a weigh action (i.e., it activates its ‘weight’ sensor)
it will perceive either \(\text{obsLight, obsMedium or obsHeavy}\); for other actions, it will per-
ceive \(\text{obsNil}\). The robot experiences its world (domain) through three Boolean features:
\(P = \{\text{full, drank, holding}\}\) 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
\([\text{obsLight} | \text{weigh}]_{0.7}(\neg\text{full} \land \text{drank} \land \text{holding})\) 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 \((\neg\text{full} \land \neg\text{holding}) \rightarrow ([\text{grab}]_{0.9}(\neg\text{full} \land \text{holding}) \land [\text{grab}]_{0.1}(\neg\text{full} \land
\neg\text{holding}))\) 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
\(\mathcal{EP}\) [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:

- a finite set of \(p_1, \ldots, p_n\),

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. a countable set of names $\Omega = \{q_1, q_2, \ldots\}$ of all rational numbers in $\mathbb{Q}$.

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

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

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

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

$[\alpha]_q \varphi$ is read ‘The probability of reaching a world in which $\varphi$ holds after executing $\alpha$, is equal to $q$. [$\alpha$] abbreviates $[\alpha]_1$, and $(\alpha)\varphi$ abbreviates $\neg [\alpha] \neg \varphi$. $[\varsigma]_q \varphi$ can be read ‘The probability of perceiving $\varsigma$ in a world in which $\varphi$ holds is equal to $q$, given $\alpha$ was performed’. $\neg [\varsigma]_q \varphi$ can be written as $[\neg \varsigma]_q \varphi$ and is read ‘It is possible to perceive $\varsigma$ in a $\varphi$-world, given $\alpha$ was performed’. We may write $[\varsigma]_q \alpha \varphi$ instead of $[\varsigma]_q \alpha_1 \varphi$.

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 $\text{Reward}$ and $\text{Cost}$ as special predicates. $\text{Reward}(r)$ can be read ‘The reward for being in the current situation is $r$ units’ and we read $\text{Cost}(\alpha, c)$ as ‘The cost for executing $\alpha$ is $c$ units’.

Note that formulae with nested modal operators, like $[\alpha]_q [\varsigma]_q \varphi$, $[\alpha]_q [\alpha]_q \varphi$, et cetera, are not in $\mathcal{L}_{\text{SLAOP}}$. ‘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 $\varphi$ after an arbitrary sequence of actions and observations. As usual, we treat $\bot, \lor, \rightarrow$ and $\leftarrow$ as abbreviations.

### 2.2. Semantics

Our semantics follows that of multi-modal logic $\mathbf{K}$ [31]. However, structures (alias, possible worlds models [32,33]) are non-standard. Standard modal logic structures are tuples $(W, R, V)$, 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 \( w: \wp \mapsto \{0, 1\} \) 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 \( \mathcal{S} = \langle W, R, O, N, Q, U \rangle \) such that

1. \( W \subset C \) a non-empty set of possible worlds;
2. \( R \) is a mapping that provides an accessibility relation \( R_a : W \times W \times \mathbb{Q} \cap [0, 1] \) for each action \( \alpha \in \mathfrak{A} \): Given some \( w^- \in W \) as the first component of a triple in \( R_a \), we require that \( \sum_{(w^-, w^+, pr) \in R_a} pr = 1 \); Also, if \( (w^-, w^+, pr), (w^-, w'^+, pr') \in R_a \), 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 \( \Omega \), 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 \( (w^-, w^+, pr) \in R_a \), 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 \( Re : W \mapsto Q \) is a reward function and \( Co \) is a mapping that provides a cost function \( Co_\alpha : C \mapsto Q \) for each \( \alpha \in \mathfrak{A} \);

\( R_a \) defines which worlds \( w^+ \) are accessible via action \( \alpha \) performed in world \( w^- \) and the transition probability \( pr \in \mathbb{Q} \cap [0, 1] \). \( Q_\alpha \) defines which observations \( o \) are perceivable in worlds \( w^+ \) accessible via action \( \alpha \) 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 \( \alpha \) 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 \( \mathcal{S} \) be a SLAOP structure, with \( \alpha, \alpha' \in \mathfrak{A} \), \( \varsigma, \varsigma' \in \Omega \), \( q, r, c \in \mathfrak{Q} \). Let \( p \in \wp \) and let \( \varphi \) be any sentence in \( \mathcal{L}_{SLAOP} \). We say \( \varphi \) is satisfied at world \( w \) in structure \( \mathcal{S} \) (written \( \mathcal{S}, w \models \varphi \)) if and only if the following holds:

1. \( \mathcal{S}, w \models \top \) for all \( w \in W \);
2. \( \mathcal{S}, w \models p \iff w(p) = 1 \) for \( w \in W \);
3. \( \mathcal{S}, w \models \neg \varphi \iff \mathcal{S}, w \not\models \varphi \);
4. \( \mathcal{S}, w \models \varphi \land \varphi' \iff \mathcal{S}, w \models \varphi \) and \( \mathcal{S}, w \models \varphi' \);
5. \( \mathcal{S}, w \models \alpha = \alpha' \iff \alpha, \alpha' \in \mathfrak{A} \) are the same element;
6. \( \mathcal{S}, w \models \varsigma = \varsigma' \iff \varsigma, \varsigma' \in \Omega \) are the same element;
7. \( \mathcal{S}, w \models \text{Reward}(r) \iff \text{Re}(w) = r \);
8. \( \mathcal{S}, w \models \text{Cost}(\alpha, c) \iff Co_\alpha(w) = c \);
9. \( \mathcal{S}, w \models [\varsigma | \alpha]_q \varphi \iff (\exists w') (\exists pr)(w, w', pr) \in R_a \) and \( \mathcal{S}, w \models \varphi \)
   then \( (N(\varsigma), w', q) \in Q_a \);
10. \( \mathcal{S}, w \models [\alpha]_q \varphi \iff (\sum_{(w, w', pr) \in R_a} \mathcal{S}, w' \models \varphi) pr = q \).
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: \( \{ (\alpha) \varphi \leftrightarrow \bigvee_{z \in \Omega} (z | \alpha) \varphi \mid \alpha \in \mathcal{A} \} \).

A formula \( \varphi \) is valid in a SLAOP structure (denoted \( \mathcal{S} \models \varphi \)) if \( \mathcal{S}, w \models \varphi \) for every \( w \in W \). \( \varphi \) is SLAOP-valid (denoted \( \models \varphi \)) if \( \varphi \) is true in every structure \( \mathcal{S} \). \( \varphi \) is satisfiable if \( \mathcal{S}, w \models \varphi \) for some \( \mathcal{S} \) 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 \( \mathcal{S}, w \models \varphi \) when \( \varphi \) is a propositional formula.

Let \( \mathcal{X} \subseteq \text{SLAOP} \). If, for all \( \theta \in \mathcal{X} \), \( \models \psi \) whenever \( \models \theta \), we say \( \mathcal{X} \) logically entails \( \psi \) (abbreviated \( \mathcal{X} \models \psi \)). If \( \mathcal{X} \) logically entails \( \psi \) and \( \mathcal{X} \) contains a single sentence \( \theta \), then we omit the brackets and write \( \theta \models \psi \). If \( \models \theta \leftrightarrow \psi \), we say \( \theta \) and \( \psi \) 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 \( \mathcal{C}_{\text{SLAOP}} \).

A labeled formula is a pair \((n, \varphi)\), where \( \varphi \) is a formula and \( n \) is an integer called the label of \( \varphi \). A skeleton \( \Sigma \) is a binary relation \( \Sigma \subseteq \mathcal{A} \times \mathbb{N} \). Elements \((\alpha, n')\) of the relation are denoted \( \stackrel{\alpha}{\rightarrow} n' \). A node \( N_k^j \) is a pair \((\Gamma_k^j, \Sigma_k^j)\) with superscript \( j \) the branch index and subscript \( k \) the node index, where \( \Gamma_k^j \) is a set of labeled formulae and \( \Sigma_k^j \) is a skeleton. The initial node to which \( \mathcal{C}_{\text{SLAOP}} \) 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 \( B^i(T) \) of some tree \( T \) is the set of nodes with the same branch index \( j \): \( B^i(T) = \{ N_k^j \in T \mid k = 0, 1, 2, \ldots \} \).

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 \( N_k^j \) creates one or more new nodes; its child(ren). If it creates one child, then it is identified as \( N_{k+1}^j \). If \( N_k^j \) creates a second child, it is identified as \( N_0^{j'} \), where \( j' \) is a fresh integer. That is, for every child created beyond the first, a new branch is started. Node \( N_k^j \) is a leaf node of tree \( T \) if there is no node \( N_{k'}^j \) in branch \( B^i(T) \) 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 \( B^i(T), \ldots, B^i(T) \) are closed, else it is open.

Let \( N_k^j = \langle \Gamma_k^j, \Sigma_k^j \rangle \) be a leaf node. The tableau rules for SLAOP follow.

- A rule may only be applied to an open leaf node.
• rule $\bot$: If $\Gamma^j_k$ contains $(n, \Phi)$ and $(n, \neg \Phi)$, then create node $(\Gamma^0_k \cup \{(n, \bot)\}, \Sigma^j_k)$.

• rule $\neg$: If $\Gamma^j_k$ contains $(n, \Phi)$, where $\Phi$ contains $\neg \neg$, then create node $(\Gamma^j_k \cup \{(n, \Phi')\}, \Sigma^j_k)$, where $\Phi'$ is $\Phi$ without $\neg \neg$.

• rule $\wedge$: If $\Gamma^j_k$ contains $(n, \Phi \wedge \Phi')$, then create node $(\Gamma^j_k \cup \{(n, \Phi), (n, \Phi')\}, \Sigma^j_k)$.

• rule $\vee$: If $\Gamma^j_k$ contains $(n, \neg (\Phi \wedge \Phi')$), then create node $N^j_{k+1} = (\Gamma^j_k \cup \{(n, \neg \Phi)\}, \Sigma^j_k)$ and node $N^j_0 = (\Gamma^j_k \cup \{(n, \neg \Phi')\}, \Sigma^j_k)$, where $j'$ is a fresh integer.

• rule $==$: If $\Gamma^j_k$ contains $(0, c = c')$ and in fact constants $c$ and $c'$ do not refer to the same constant, or if $\Gamma^j_k$ contains $(0, c \neq c')$ and in fact constants $c$ and $c'$ do refer to the same constant, then create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.

• rule $\text{Re}$: If $\Gamma^j_k$ contains $(0, \text{Reward}(r))$ and $(0, \text{Reward}(r'))$ such that $r \neq r'$, then create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.

• rule $\text{Co}$: If $\Gamma^j_k$ contains $(0, \text{Cost}(\alpha, c))$ and $(0, \text{Cost}(\alpha, c'))$ such that $c \neq c'$, then create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.

• rule $\neg [\zeta]$: If $\Gamma^j_k$ contains $(0, [\zeta | \alpha]_0 \Phi)$ and $(0, \neg [\zeta | \alpha]_0 \Phi')$ where $\Phi \wedge \Phi' \neq \bot$, then create node $(\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.

• rule $[\zeta]$: If $\Gamma^j_k$ contains $(0, [\zeta | \alpha]_0 \Phi)$ and $(0, [\zeta' | \alpha]_0 \Phi')$ where $\Phi \wedge \Phi' \neq \bot$, then

1. if $q \neq q'$, create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \neg (\zeta = \zeta'))\}, \Sigma^j_k)$.
2. if $q + q' > 1$, create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \zeta = \zeta')\}, \Sigma^j_k)$.

• rule $\text{obs}$: If $\Gamma^j_k$ contains $(0, [\zeta_1 | \alpha]_{y_1} \Phi_1), (0, [\zeta_2 | \alpha]_{y_2} \Phi_2), \ldots, (0, [\zeta_m | \alpha]_{y_m} \Phi_m)$ such that $\zeta_\alpha$ is not the same as $\zeta_\alpha$ for all $x$ and $y$ ($1 \leq x, y \leq m$ and $x \neq y$) and $\bigwedge_{i=1}^m \Phi_i \neq \bot$, then

1. if $\sum_{i=1}^m q_i = 1$, then create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, [\zeta'_1 | \alpha]_0 \Phi'), (0, [\zeta'_2 | \alpha]_0 \Phi'), \ldots, (0, [\zeta'_m | \alpha]_0 \Phi')\}, \Sigma^j_k)$, where $\zeta' \in \Omega \setminus \{\Phi_0 \in \Omega \mid z = 1, 2, \ldots, m\}$ and $\Phi' = \bigwedge_{i=1}^m \Phi_i$.
2. if $\sum_{i=1}^m q_i < 1$, then create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \neg [\zeta_1 | \alpha]_0 \Phi) \vee \neg [\zeta_2 | \alpha]_0 \Phi) \vee \cdots \vee \neg [\zeta_m | \alpha]_0 \Phi)\}, \Sigma^j_k)$, where $\zeta' \in \Omega \setminus \{\Phi_0 \in \Omega \mid z = 1, 2, \ldots, m\}$ and $\Phi' = \bigwedge_{i=1}^m \Phi_i$.
3. if $\bigvee_{i=1}^m \zeta_i = \Omega$, then if $\sum_{i=1}^m q_i \neq 1$, create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.

• rule $\to (\alpha)$: If $\Gamma^j_k$ contains $(0, [\alpha]_0 \Phi)$ for $0 < q \leq 1$, then create node $(\Gamma^j_k \cup \{(0, \neg [\alpha]_0 \Phi)\}, \Sigma^j_k)$.

• rule $\to (\zeta)$: If $\Gamma^j_k$ contains $(0, [\zeta | \alpha]_0 \Phi)$ for $0 < q \leq 1$, then create node $N^j_{k+1} = (\Gamma^j_k \cup \{(0, \neg [\zeta | \alpha]_0 \Phi)\}, \Sigma^j_k)$.

• rule $\exists$: If $\Gamma^j_k$ contains $(0, [\alpha]_0 \Phi)$, then create node $(\Gamma^j_k \cup \{(n, \Phi)\}, \Sigma^j_k \cup \{0 \alpha \rightarrow n\})$, where $n$ is a fresh integer.

• rule $\Box$: If $\Gamma^j_k$ contains $(0, [\alpha]_0 \Phi)$ and $\Sigma$ contains $\alpha \rightarrow n$, then create node $(\Gamma^j_k \cup \{(n, \Phi)\}, \Sigma^j_k \cup \{0 \alpha \rightarrow n\})$.

• rule $\text{rng}$: If $\Gamma^j_k$ contains $(0, [\alpha]_0 \Phi)$ such that $q < 0$ or $q > 1$, then create node $(\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$. 
• rule $1-q$: If $\Gamma^j_k$ contains $(0, [\alpha]_q \Phi)$, then create node $(\Gamma^j_k \cup \{(0, [\alpha]_{1-q} \neg \Phi)\}, \Sigma^j_k)$.

• rule $\neg [\alpha]$: If $\Gamma^j_k$ contains $(0, [\alpha]_q \Phi)$ and $(0, \neg [\alpha]_q \Phi')$ where $q < 1$, create node $(\Gamma^j_k \cup \{(0, \neg [\alpha]([\Phi \leftrightarrow \Phi']))\}, \Sigma^j_k)$.

• rule $[\alpha]_q$: If $\Gamma^j_k$ contains $(0, [\alpha]_q \Phi)$ and $(0, [\alpha]_q \Phi')$, then
  1. create node $(\Gamma^j_k \cup \{(0, [\alpha]_q (\Phi \land \Phi'))\}, \Sigma^j_k)$.
  2. create node $(\Gamma^j_k \cup \{(0, [\alpha]_q (\Phi \rightarrow \Phi'))\} \cup \{(\alpha, q, (\Phi' \land \neg \Phi))\}, \Sigma^j_k)$.
  3. if $q > q'$, create node $(\Gamma^j_k \cup \{(0, [\alpha]_q (\Phi \land \neg \Phi'))\}, \Sigma^j_k)$.

• rule $\text{dne}$: If $\Gamma^j_k$ contains $(0, [\alpha]_q \Phi)$, $(0, [\alpha]_q \Phi')$ and $(0, [\alpha]_q \Phi'')$, then
  1. if $\Phi'' \equiv \Phi \land \Phi'$, then: if $0 \leq q + q' - q'' \leq 1$, create node $(\Gamma^j_k \cup \{(0, [\alpha]_{q+q'-q''} \neg (\neg \Phi \land \neg \Phi'))\}, \Sigma^j_k)$, else create node $(\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.
  2. if $\Phi'' \equiv \Phi \lor \Phi'$, then: if $0 \leq q + q' - q'' \leq 1$, create node $(\Gamma^j_k \cup \{(0, [\alpha]_{q+q'-q''} (\Phi \land \Phi'))\} \cup \{(\alpha, q, (\Phi' \lor \Phi'))\}, \Sigma^j_k)$, else create node $(\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.
  3. if $\Phi'' \equiv \Phi \land \chi$ and $\Phi' \equiv \Phi \lor \chi$ for some $\chi \neq \Phi$, then: if $0 \leq q + q' - q \leq 1$, create node $(\Gamma^j_k \cup \{(0, [\alpha]_{q+q'-q} \chi)\}, \Sigma^j_k)$, else create node $(\Gamma^j_k \cup \{(0, \bot)\}, \Sigma^j_k)$.

If one has a tree with trunk $N_0 = \{\{0, \Phi\}, \emptyset\}$, we’ll say one has a tree for $\Psi$. Note that $(n, \Phi) \not\in \Gamma$ for $n > 0$ when $\Phi$ 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 $\text{RI}$ applicable to any labeled formula $(n, \Phi)$, if the rule says to create a new node $(\Gamma \cup F, \Sigma)$ while $F$ is already in $\Gamma$, then $\text{RI}$ may not be applied to $(n, \Phi)$. 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 $\mathcal{S}_{\text{SLAO}}$ 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 SLAO structure be constructed from the information in a tableau tree whenever the tree indicates that a model exists for the input sentence—while the SLAO 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 $\psi$. It thus suffices to construct for any open saturated tree for $\psi \in \mathcal{S}_{\text{SLAO}}$, a SLAO structure $\mathcal{S}$ in which there is a world $w$ in $\mathcal{S}$ such that $\psi$ is true in $\mathcal{S}$ at $w$. 
4. Examples

This section includes three examples of $\mathcal{CSLAOP}$ 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 $BK$.

- $[\text{obsLight} \mid \text{weigh}]_{0.7}(\neg \text{full} \land \text{drank})$ 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,
- $(\text{full} \land \neg \text{drank} \land \text{holding}) \rightarrow ([\text{drink}]_{0.85}(\neg \text{full} \land \text{drank} \land \text{holding}) \land [\text{drink}]_{0.15}(\neg \text{full} \land \neg \text{drank} \land \text{holding}))$ gives the probabilities of reaching the only two worlds reachable from the world where $\text{full}$ and $\text{holding}$ are true and $\text{drank}$ is false.
- $\text{holding} \rightarrow [\text{drink}] \text{holding}$ expresses that the agent doesn’t drop the oil-can when drinking,
- $(\text{full} \land \neg \text{holding}) \rightarrow ([\text{grab}]_{0.7}(\text{full} \land \text{holding}) \land [\text{grab}]_{0.2}(\neg \text{full} \land \neg \text{holding}) \land [\text{grab}]_{0.1}(\text{full} \land \neg \text{holding}))$ is another specification of transition probabilities given the can is full and the agent is not holding it when it grabs it.
- $(\text{full} \land \text{drank} \land \neg \text{holding}) \rightarrow [\text{grab}] \text{drank}$ 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 $IC \rightarrow \phi$ is logically entailed by $BK$, where $\phi$ 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: $\text{grab} := g$, $\text{drink} := d$, $\text{weigh} := w$, $\text{full} := f$, $\text{drank} := h$, and $\text{obsLight} := oL$.

<table>
<thead>
<tr>
<th>Line</th>
<th>$\Gamma$ &amp; $\Sigma$</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$(0, f \land d), (0, \neg\neg(oL \mid w)[0.1\neg f])$</td>
<td>trunk</td>
</tr>
<tr>
<td>2</td>
<td>$(0, f \land d), (0, oL \mid w)[0.1\neg f]$</td>
<td>rl.$\neg$:1</td>
</tr>
<tr>
<td>3</td>
<td>$(0, oL \mid w)[0.7(\neg f \land d)]$</td>
<td>bk.</td>
</tr>
<tr>
<td>4</td>
<td>$(0, \neg(oL = oL))$</td>
<td>rl.$[\Sigma]_{oL}1.2.3$</td>
</tr>
<tr>
<td>5</td>
<td>$(0, \perp)$</td>
<td>rl.$=4$</td>
</tr>
</tbody>
</table>
In the proof given in Table 1, the agent's initial condition is expressed as \((0, f \land d)\). Note however, that \(BK \models 1 \rightarrow \neg[\text{obsLight} \mid \text{weight}]_{0.1} \rightarrow \text{full}\) for any initial condition 1. 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 \(BK \models (\text{full} \land \neg \text{drank} \land \text{holding}) \rightarrow [\text{drink}]_{0.15}(\text{full} \lor \neg \text{drank}).\)

<table>
<thead>
<tr>
<th>Line</th>
<th>(\Gamma &amp; \Sigma)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((0, f \land d \land h), (0, \neg[d]_{0.15}(f \lor \neg d)))</td>
<td>trunk</td>
</tr>
<tr>
<td>2</td>
<td>((0, f), (0, \neg d), (0, h), (0, \neg[d]_{0.15}(f \lor \neg d)))</td>
<td>rl&amp;:1</td>
</tr>
<tr>
<td>3</td>
<td>((0, (f \land d \land h) \lor (d)<em>{0.05}(\neg f \land d \land h) \land (d)</em>{0.15}(\neg f \land d \land h)))</td>
<td>bk.</td>
</tr>
<tr>
<td>4</td>
<td>((0, \neg(f \land d \land h) \lor (d)<em>{0.05}(\neg f \land d \land h) \land (d)</em>{0.15}(\neg f \land d \land h)))</td>
<td>nf:3</td>
</tr>
<tr>
<td>5</td>
<td>((0, \neg f), (0, d), (0, \neg h), (0, \neg[d]<em>{0.05}(\neg f \land d \land h)), (0, \neg[d]</em>{0.15}(\text{...})))</td>
<td>rl&amp;\lor&amp;:4</td>
</tr>
<tr>
<td>6</td>
<td>((0, \bot), (0, \bot), (0, \bot), (0, \neg[d]_{0.15}(\neg f \land d \land h)))</td>
<td>rl&amp;:1&amp;:5</td>
</tr>
<tr>
<td>7</td>
<td>rl&amp;&amp;:2.5, rl&amp;&amp;:2.5, rl&amp;&amp;:2.5, ((0, \neg[d]_{0.15}((\neg f \land d \land h) \lor (f \lor \neg d))))</td>
<td>rl&amp;\lor&amp;&amp;:5</td>
</tr>
<tr>
<td>8</td>
<td>((1, \neg f \land d \land h), (1, \neg [f \lor \neg d], 1, 1))</td>
<td>bk.</td>
</tr>
<tr>
<td>9</td>
<td>((0, h \lor [d]_{k}))</td>
<td>rl&amp;&amp;:8, rl&amp;&amp;:9</td>
</tr>
</tbody>
</table>

Table 3. Proof that \(BK \models (\text{full} \land \text{drank} \land \neg \text{holding}) \rightarrow [\text{grasp}]_{0.7}(\text{full} \land \text{drank} \land \text{holding}).\)

<table>
<thead>
<tr>
<th>Line</th>
<th>(\Gamma &amp; \Sigma)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>((0, f \land d \land \neg h), (0, \neg[g]_{0.7}(f \land d \land h)))</td>
<td>trunk</td>
</tr>
<tr>
<td>2</td>
<td>((0, f), (0, d), (0, \neg h), (0, \neg[g]_{0.7}(f \land d \land h)))</td>
<td>rl&amp;:1</td>
</tr>
<tr>
<td>3</td>
<td>((0, (f \land d \land h) \lor (g)<em>{0.7}(f \land d \land h) \lor (g)</em>{0.2}((f \land d \land h) \land (f \land \neg h))))</td>
<td>bk.</td>
</tr>
<tr>
<td>4</td>
<td>((0, \neg f), (0, h), (0, \neg[g]<em>{0.7}(f \land d \land h)), (0, \neg[g]</em>{0.2}((f \land d \land h) \land (f \land \neg h))))</td>
<td>nf&amp; rl&amp;\lor&amp;:2</td>
</tr>
<tr>
<td>5</td>
<td>((0, \bot), (0, \bot), (0, \bot), (0, \neg[d]_{0.7}(f \land d \land h)))</td>
<td>bk.</td>
</tr>
<tr>
<td>6</td>
<td>rl&amp;&amp;:2.4, rl&amp;&amp;:2.4, ((0, \neg f), (0, \neg d), (0, h), (0, \neg[g]_{0.7}(f \land d \land h)))</td>
<td>nf&amp; rl&amp;\lor&amp;:5</td>
</tr>
<tr>
<td>7</td>
<td>rl&amp;&amp;:2.6, rl&amp;&amp;:2.6, rl&amp;&amp;:2.6, continues in table below</td>
<td>rl&amp;&amp;:4</td>
</tr>
<tr>
<td>8</td>
<td>rl&amp;&amp;:2.6, rl&amp;&amp;:2.6, rl&amp;&amp;:2.6, continues in table below</td>
<td>continues in table below</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Line</th>
<th>(\Gamma &amp; \Sigma)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>((0, \neg[g]((f \land d \land h) \lor (f \land h)))</td>
<td>rl&amp;&amp;:1.7</td>
</tr>
<tr>
<td>10</td>
<td>((1, \neg[f \land d \land h] \lor (f \land h)), 1))</td>
<td>rl&amp;&amp;:9</td>
</tr>
<tr>
<td>11</td>
<td>((1, (\neg f \land d \land h) \land (f \land h)))</td>
<td>rl&amp;&amp;:10</td>
</tr>
<tr>
<td>12</td>
<td>((1, \neg[f \land d \land h] \lor (f \land h)), (1, f), (1, h))</td>
<td>rl&amp;&amp;:11</td>
</tr>
<tr>
<td>13</td>
<td>((1, (1, f), (1, d), (1, d), (1, h)), (1, h))</td>
<td>rl&amp;&amp;:12</td>
</tr>
<tr>
<td>14</td>
<td>((1, \bot), (1, \bot), (1, \bot), (1, \bot), (1, \bot), (1, \bot))</td>
<td>rl&amp;&amp;:13</td>
</tr>
<tr>
<td>15</td>
<td>rl&amp;&amp;:13, rl&amp;&amp;:13, rl&amp;&amp;:13, rl&amp;&amp;:12, rl&amp;&amp;:14</td>
<td>rl&amp;&amp;:14</td>
</tr>
</tbody>
</table>
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


