First-Order Models for POMDPs

Siddharth Srivastava
Computer Science Division
University of California, Berkeley

Stuart Russell
Computer Science Division
University of California, Berkeley

Avi Pfeffer
Charles River Analytics

Abstract

Interest in relational and first-order languages for probability models has grown rapidly in recent years, and with it the possibility of extending such languages to handle decision processes—both fully and partially observable. We examine the problem of extending a first-order, open-universe language to describe POMDPs and identify non-trivial representational issues in describing an agent's capability for observation and action—issues that were avoided in previous work only by making strong and restrictive assumptions. We present a solution based on ideas from modal logic, and show how to handle cases like being able to act upon an object that has been detected through one's observations.

1 Introduction

Relational and first-order languages for probability models (as well as their close relatives the probabilistic programming languages) constitute an important development for AI in general and for machine learning in particular [Getoor and Taskar, 2007]. The availability of such expressive languages should make it possible to write not just complex probability models but also complex decision models—the foundation for rational behavior. To achieve this goal, probabilistic languages can be extended with information about actions, observations, and rewards. As we show, however, such extensions raise significant difficulties. We argue that as machine learning and probabilistic AI researchers grapple with these more expressive representations, it will be necessary to use techniques from logical AI in conjunction with probabilistic techniques to produce representations that are both expressive enough to model the real world and mean exactly what is intended. In this paper, we use such techniques to develop a representation of first-order decision models.

We begin in §2 by describing partially observable Markov decision processes (POMDPs), which are a very general form of decision model. As a "warm-up" for the real problem, we show how POMDPs may be defined as directed graphical models by extending dynamic Bayesian networks (DBNs). We focus in particular on the *observation model*, which gives the distribution over anticipated sensor readings given the current world state. When we move to the relational or first-order setting, we have to contend with the fact that observations, rather than being values of variables, become sentences describing properties of objects using names for those objects. It turns out to be quite tricky to say in a formal way what a given sensor can or cannot supply in the way of observation sentences; expressing the agent's capability for action is also tricky, especially when the agent can act on objects whose existence has been determined only through its sensors. Consider the following example:

Consider an airport security system whose sensors include passport scanners at check-in kiosks, boarding pass scanners, X-ray scanners, etc. A person passing through the airport generates observations from each of these scanners, while each scanner generates a sequence of observations from the sequence of persons passing through it. Thus, the passport scanner at location A may generate observations of the form $\text{IDName}(p_{A,1}) = \text{Bond}$, $\text{IDNumber}(p_{A,1}) = 174666007$, $\text{HeightOnID}(p_{A,1}) = 185cm, \ldots$; *a boarding-pass scanner at B may generate a sequence of the form* $\text{Destination}(p_{B,1}) = \ldots$, $\text{IDNumber}(p_{B,1}) = 174666007$, *and finally, an X-ray scanner at C may generate observations of the form* $\text{MeasuredHeight}(p_{C,1}) = 171cm$, $\text{MeasuredHeight}(p_{C,2}) = \ldots$. *In these observation streams, the symbols* $p_{A,i}, p_{B,i}$ *and* $p_{C,i}$ *are placeholder identifiers (essentially skolem constants or "gensyms" in Lisp terminology). Although each use of a given symbol necessarily corresponds to the same in-*

dividual, different symbols may or may not correspond to different individuals; thus, it is possible that $p_{A,1}$ and $p_{C,2}$ refer to the same person, while it is also possible that $p_{A,1}$ and $p_{B,1}$ refer to different people even though they are carrying a passport and a boarding pass with the same ID number.

Such a scenario can be modeled probabilistically by a first-order, open-universe language such as BLOG [Milch *et al.*, 2005], which enables reasoning about identity uncertainty. To make *decisions*—such as searching or arresting a given individual—we need a POMDP with rewards, actions, and an observation model. Informally, we might say, "Everyone in the security line will get scanned": $\forall x \quad InLine(x) \rightarrow Scanned(x)$, and "For everyone who gets scanned, we will observe a measured height":

$$\forall x Scanned(x) \rightarrow Observable(MeasuredHeight(x)) \quad (1)$$

So far, so good. Now, suppose we know, "Johnny and his mother are in the security line." While it is true, in a sense, that we will get a measured height for Johnny's mother, it is *not* true that the X-ray scanner will tell us $MeasuredHeight(Mother(Johnny)) = 171cm$. Technically, the problem arises because we are trying to substitute $Mother(Johnny)$ for x in the universally quantified sentence (1), but one occurrence of x is inside $Observable(\cdot)$, which is a *modal operator*. Practically, the problem arises because the sensor doesn't know who Johnny's mother is. The same issue can arise on the action side: telling the security guard to "Arrest Johnny's mother" doesn't work if the guard cannot execute that action as stated.

We provide a general solution to the problem of representing sensing and action in open universe models (§3). Our solution is based on the idea of defining models of sensors and actuators and a sensori-motor stream that connects them (§3.2). We avoid the pitfalls of standard solutions by defining meta-predicates for concepts such as the ability to observe something or the ability to do something (§3.3). We define a notion of syntactic entailment for these meta-predicates and a semantics for them and show that syntactic entailment is sound and complete with respect to the semantics (§3.4). We also present several examples showing how our representation captures the intuitive meaning of situations (§4). We also discuss the ways in which previous attempts to define first-order POMDPs have used strongly restricted languages to avoid these problems altogether. These restrictions do not allow an agent to walk into a room, see something, and pick it up (§5).

2 POMDPs

Definition 1 *(POMDP)* A POMDP is defined as $\langle S, A, O, T, \Omega, R \rangle$, where S, A, O are finite sets of state, action and observation symbols; $T(S_{t+1} = s' \mid S_t = s, A_t = a)$ defines the transition model, i.e., the probability of reaching state s' if action a is applied in state s; $\Omega(O_{t+1} = o \mid S_{t+1} = s', A_t = a)$ defines the sensor model, i.e., the probability of receiving an observation o when state s' is reached via action a, and $R : S \times A \rightarrow \mathbb{R}$ defines the reward that the agent receives on applying a given action at a given state.

A POMDP defines a decision-theoretic planning problem for an agent. At every step, the agent executes an action from A, then the environment enters a new state (according to T), then the agent receives an observation and a reward according to Ω and R. Typically, rewards are aggregated via an infinite discounted sum, so the optimal policy maximizes the expected value of $\sum_{i=1\ldots\infty} \gamma^i \cdot r(i)$ where $r(i)$ is the reward obtained at timestep i and $\gamma < 1$ is a constant.

A DBN [Dean and Kanazawa, 1989] describes a factored, homogeneous, first-order Markov process as a "two-slice" Bayesian network showing how variables at time $t+1$ depend on variables at t. At each time step, any subset of variables may be observed. A *dynamic decision network* or DDN [Russell and Norvig, 1995] represents a POMDP using the the following additional node types:

- An *action variable* A_t whose values are the possible actions at time t. For now, assume this set of actions is fixed. The POMDP's transition model is represented through the conditional dependence of variables at $t+1$ on the value of A_t and other variables at t.
- A reward variable R_t, which depends deterministically on A_t and other variables at time t.
- A set of Boolean *observability variables* $ObsX_t$, one for each ordinary variable X_t. Each observability variable is necessarily observed at each time step, and $ObsX_t$ is true iff X_t is observed. Observability variables may have as parents other observability variables, ordinary variables, or the preceding action variable. [1] The DBN with X and $ObsX$ variables defines Ω.

Russell and Norvig's description of DDNs [Russell and Norvig, 1995] does not mention the need for a model, and implicitly assumes that a fixed subset of variables is always observed. (In our model, an always-

[1] Observability variables capture the full range of possibilities in the spectrum from missing-completely-at-random to not-missing-at-random (NMAR) data [Little and Rubin, 2002].

observable X_t has an $ObsX_t$ with a deterministic prior set to 1.) It should be clear, however, that different observability models correspond to different POMDPs.

3 First-Order Representation

Structures, States and Transitions Given a set of types $\mathcal{T} = \{\tau_1, \ldots \tau_k\}$, we define a first-order vocabulary as a set of function symbols with their type signatures. Constant symbols are represented as zero-ary function symbols and predicates as Boolean functions. Given a first-order vocabulary, a structure is defined as a tuple $\langle \mathcal{U}, \mathcal{I} \rangle$ where $\mathcal{U} = \langle \mathcal{U}_1, \ldots \mathcal{U}_k \rangle$ and each $\mathcal{U}_i$ is a set of elements of type $\tau_i \in \mathcal{T}$. The interpretation $\mathcal{I}$ has, for each function symbol in the vocabulary, a function of the corresponding type signature over $\mathcal{U}_1, \ldots, \mathcal{U}_k$. WLOG, we assume that every vocabulary is defined over a set of types that includes the type *Timestep*. We assume the set of values for *Timestep* to be the set of natural numbers. Functions whose last argument is of type timestep are called *fluents*. We represent states and possible worlds as structures.

We represent actions using sets of formulas defining the value of a fluent at timestep $t+1$ in terms of non-fluents and fluents at timestep t. The reward function can be expressed as a fluent in a similar manner. This approach is similar to writing successor state axioms in situation calculus [Reiter, 2001]. In order to define probabilistic action effects these ideas can be easily extended to include a probability distribution for all possible action effects. Sanner et al. [Sanner and Boutilier, 2009] describe one such approach. In order to make the discussion independent of any particular language of implementation, we will formalize our ideas in the language of first-order logic (FOL). We note however that languages for writing generative first-order probabilistic models like BLOG can easily express sentences necessary to formulate probabilistic action updates. In particular, in BLOG the update for a fluent $f(x_1, \ldots, x_k)$ can be expressed as a dependency of the form *if* φ_1 *hen* $CPD_1()$ *elseif* φ_2 *then* $CPD_2()$ *; else* $CPD_n()$*;*, where the conditions φ_i are arbitrary FOL formulas with free variables from $\{x_1, \ldots, x_n\}$ and CPDs are probability distributions conditioned on expressions over $\{x_1, \ldots, x_n\}$. E.g., the following BLOG expression describes the update on the predicate *atScanner(x, t)*:

$atScanner(x, t+1)$ {*if applied_sendToScanner(x,t)*
then $\sim$ *followedInstructionCPD()*;
elseif atScanner(x, t)
then $\sim$ *leftScannerCPD()*;
else $\sim$ *defaultAtScannerCPD()*}

The first clause of this update states that if *sendToScanner* was applied at t then the likelihood that x is at scanner at $t+1$ is given by a probability distribution *followedInstructionCPD*. Successive lines can be understood similarly. We define a set $\mathcal{A}$ of functions that indicate whether an action is applied and denote the function corresponding to action a as *applied_a*. These formulations do not address a crucial point: which terms can be used in action arguments? As implied in the introduction, an agent's sensors may need to provide information about argument values. We formalize this below.

3.1 Challenges in Defining Sensor and Actuator Models

For clarity in discussion, we will motivate the problems in expressing observation and action models as well as their solutions in non-deterministic, partially observable situations where probabilistic information about possible action effects or observations is not available. This will be particularly helpful because the biggest problems in formulating FO-POMDPs are those of designing and utilizing appropriate FOL constructs. As we noted for the case of actions above, our solutions generalize easily to probabilistic settings. This is noted explicitly where needed.

Generalizing the *obsX* idea discussed in §2 to first-order representations raises several questions. First, consider the language in which observations are expressed. Suppose the sensor reports an observation: *MeasuredHeight*(p_{17})*=171cm*. Is p_{17} an element of the universe or a constant symbol? The two possibilities correspond to model-theoretic and proof-theoretic frameworks respectively. Both present numerous ambiguities. In order to plan in partially observable settings the agent maintains a belief state, which is in general, a representation of the set of possible real states. On receiving the observation, the agent must update its belief state to eliminate all possible worlds where the interpretation of *MeasuredHeight* for the element p_{17} is not 171cm. In a model-theoretic framework, the observed sentence reflects a fact about the interpretation of *MeasuredHeight*. The problem with this framework is that such information is almost impossible to obtain, because the agent doesn't know that p_{17} is the one being sensed.

In a proof-theoretic framework, belief is updated using inference rules over axioms about possible observations. A natural generalization of the *obsX* idea is to write rules of the form: $\forall \bar{x}\ \varphi(\bar{x}) \rightarrow observable(\psi(\bar{x}))$, where φ and ψ are arbitrary FOL formulas. WLOG, ψ and φ can be considered to be predicates defined using FOL formulas with variables in $\bar{x}$ as free variables. The interpretation of this formula would be "if

$\varphi(\bar{x})$ holds, then $\psi(\bar{x})$ is observed". Problems with this framework that were discussed in the introduction are consequences of the nature of universal instantiation in FOL. The axiom of universal instantiation in FOL states that if $\forall x\ \alpha(x)$ is true, then for any ground term t, $\alpha(t/x)$ (the version of $\alpha(x)$ where all free occurrences of x are replaced by t) must also be true. Therefore, the same erroneous conclusions follow if we let $\alpha(x)$ be any FOL formula that states "if x is at the scanner, then the measured height of x is observable". Intuitively, as noted in the introduction, we would like to be able to refer to the same object (e.g. "Wilma" versus "mother of Jonny") in different terms, such that one's height is observable and the other's is not, even though the precondition (*scanned*) holds for both.

3.2 System-Independent Sensor and Actuator Specifications

We begin our formal solution to the problems noted above by defining uninterpreted sensor and actuator specifications denoting the types of values that sensors generate and actuators accept.

Definition 2 *(Sensor Specification)* A Sensor specification S is a tuple $\langle \bar{T}_S, \tau_S \rangle$ where $\bar{T}_S$ is a vector of types and τ_S is a type.

$\bar{T}_S$ defines the type-vector of observation values that S produces and τ_S defines the type of new symbols that it may generate and communicate to an actuator. An X-ray sensor can be specified as $\langle \langle \textit{PersonRef, Real} \rangle, \langle \textit{PersonRef} \rangle \rangle$. Such a sensor can generate new symbols of type *PersonRef* in the form $\textit{new}(pr_{17})$, or tuples of the form $\langle pr_{17}, 171cm \rangle$. The interpretation of these values is modelled using generative models in the next section. Intuitively, the X-ray scanner may generate *PersonRef* symbols for every person that passes through it, and return a mapping from these symbols to the measured heights of the persons they represent.

Definition 3 *(Actuator Specification)* An actuator specification A is a tuple $\langle a_1, \bar{t}_1 \rangle$ denoting an action name and a vector of its argument types.

An actuator may be specified as $\langle \textit{takeSnapshotOfSrc}, \langle \textit{PersonRef} \rangle \rangle$, denoting a camera that can take pictures when given an argument of type *PersonRef*. In many settings sensors can communicate knowledge necessary for an actuator to be able to execute an action. For example, the X-ray scanner may setup a communication channel with real-time information about the position of a person. The scanner indicates that information about a *PersonRef*, e.g.. pr_{17} is available at t by generating the signal $\textit{known}(pr_{17},\ t)$. We formalize these channels as sensori-motor streams. These streams characterize the truly feasible observation and action signal sequences under given specifications.

Definition 4 *(Sensori-Motor Stream)* Let $S = \langle \bar{T}_S, \tau_S \rangle$ be a sensor specification and $A = \{\langle a_1, \bar{t}_1 \rangle, \ldots \langle a_k, \bar{t}_k \rangle\}$ be an actuator specification. A sensori-motor stream for S and A, denoted *stream(S, A)* is a timestep-indexed sequence of sets of tuples $\textit{signals}(t)$ such that each element of $\textit{signals}(t)$ is either:

- An observation value in the specification S OR
- A new symbol declaration of the form $\textit{new}(u)$ where u is a term of type τ_S OR
- A signal of the form $\textit{known}_{SA}(u)$ where $\textit{new}(u)$ occurs in $\cup_{i=0,\ldots,t-1} \textit{signals}(i)$ OR
- An effector signal of the form $a_i(\bar{r})$ where $\langle a_i, \textit{type}(\bar{r}) \rangle \in A$; $\textit{type}(\bar{r})$ is the vector of types of elements of $\bar{r}$, taken in order; and every term u occurring in $\bar{r}$ is of the form $f(u')$ where $\textit{known}_{SA}(u') \in \textit{signals}(t)$ and f is a deterministic function (one whose interpretation is a unique function in all possible structures).

3.3 Generative Models for Sensor and Actuator Specifications

In order to express generative models for given sensor and actuator specifications, we define a set of meta-predicates. These meta-predicates signify properties of the language's predicates, such as when a certain predicate's value will be observed.

Definition 5 *(Sensor Model)* A sensor model for a given sensor specification S consists of conditional dependency statements for the following predicates:

- $\textit{observable}(p(x_1, \ldots, x_n), t)$ where types of $x_1, \ldots, x_n$ correspond to the vector of observation types in S, t is of type timestep and p is a predicate in the generative model's vocabulary.
- $\textit{new}(x_1, t)$ where x_1 the type that can be generated by S and t is of type timestep.

As noted earlier, p could be a defined function/predicate. The meta-predicate *observable($p(\bar{x})$, t)* is true iff the agent receives an observation about $p(\bar{x})$ at timestep t. Both *new(x, t)* and *observable($p(\bar{x})$, t)* are themselves fully observable at t because the agent can always determine whether or not it has received an observation, and if so, the value of that observation.

Definition 6 *(Actuator Model)* An actuator model for an actuator specification A is a dependency statement for $\textit{doable}(a(x_1, ..., x_n), t)$, where a and types of $x_1, \ldots, x_n$ are as specified by A.

Sensori-motor streams are modelled using a *communication model*, which consists of dependency statements for *known()* predicates. The *known(x,t)* predicate is also observable at timestep t as it is true iff *known(x,t)* is generated by the sensor. Thus, *doable* is the only meta-predicate that need not always be observable since it may depend on other state dependent, partially observable properties.

Sensor-Actuator Initialization Sets of predefined symbols that are new, and those that are always available to actuators can be initialized by asserting expressions of the form $\wedge_{c \in \mathcal{C}_{new}} new(c, 0)$ and $\forall t \wedge_{c \in \mathcal{C}_{known}} known_{S_i A_j}(c, t)$ respectively. We will omit the subscripts from $known_{SA}()$ predicates when they are clear from the context.

Example 1 Consider an X-ray scanner specified as $\langle\langle$*PersonRef, Real*$\rangle, \langle$*PersonRef*$\rangle\rangle$ and a camera specified as $\{\langle$*takeSnapshotOfSrc*$, \langle$*PersonRef*$\rangle\rangle, \langle$*takeSnapshot*, $\langle$*Angle, Angle, Zoom*$\rangle\rangle\}$. FOL formulas modelling this example are presented in Fig. 1.

3.3.1 Syntactic Entailment and Semantics

In this section we formulate a notion of entailment over a set of logical axioms that use the meta-predicates *new*, *observable*, *known* and *doable*. Note that using person references as illustrated above does not solve the problem of incorrect inferences due to the axiom of universal instantiation. In particular, if the rules stated in Fig. 1 are treated as standard formulas in FOL we can substitute variables of type *PersonRef* with the function *record(Mother(Johnny), 12)*. The agent would then expect observations of the form *MsdHtOfSrc(record(Mother(Johnny)), 12)=...* at timestep 12. Clearly, the sensor specification in Eg. 1 does not allow such observations.

In order to take into account the state of knowledge of a sensor, we distinguish meta-predicates from the usual, object-predicates because meta-predicates denote properties of symbols rather than of objects. Intuitively, even if the agent believes that pr_{17}=*record(Mother(Johnny), t)*, truth of *observable(MsdHtOfSrc(pr_{17}), t)* does not imply *observable(MsdHtOfSrc(record(Mother(Johnny), t)), t)*. On the other hand, since object-predicates denote properties of objects, in every possible structure that satisfies pr_{17}=*record(Mother(Johnny), t)*, *MsdHtOfSrc(record(Mother(Johnny),t)) = MsdHtOfSrc(pr_{17})* holds.

We formalize this intuition using terminology from modal logic. The axiom of universal quantification is modified in modal logic to state that: if $\forall x\ \alpha(x)$ is true, then $\alpha(t/x)$ is true for terms t whose values are known. Such terms are referred to as rigid designators and represent the same object in every possible structure. In our framework, it is clear that the sensor will know the symbols that it itself generates in addition to any set of symbols that it may be initialized with, e.g. numbers and strings. E.g., if a sensor reports a new image as *new(img17)*, then the symbol *img17* is known to it.

Formally, each sensor is associated with a countably infinite set of rigid designator symbols in the vocabulary. The *new()* observation for a rigid designator makes that symbol available for future observations. Universal instantiation is modified so that bound variables that occur in meta-predicates can only be substituted with rigid designators. Formally, if x occurs in a meta-predicate in α, then the modified axiom of universal instantiation is $\forall x \alpha(x) \rightarrow \alpha(t_{rd}/x)$ where t_{rd} is a rigid designator of the type of x in the language. Given a theory T, we say that $T \vdash_{RD} \varphi$ iff φ follows from T using the axioms of FOL with universal instantiation applicable only on those quantifiers in a formula whose quantified variables occur only in object-level predicates (not meta-predicates) in that formula. For other quantifiers we use the modified version of universal instantiation.

Example 2 In Fig. 1, in equations 4 and 6 the variables y, and z range over the set of person references that have been generated by the sensor. Since *Mother(Johnny)* and *record(Mother(Johnny))* are not such symbols, incorrect inference resulting from substitution of these terms is avoided.

Semantics In order to define the semantics of the modified form of universal instantiation we need to describe the semantics of quantifiers over variables that occur in meta-predicates. As noted above, each sensor is associated with a set of rigid designator symbols. We require that every structure has a unique interpretation for each such symbol. This is easy to appreciate for rigid designators of standard types like numbers. Every structure's universe includes all real numbers; rigid designator symbols for reals (e.g., "7.3") will be interpreted as the corresponding real numbers (e.g., 7.3) in every structure. In general we will assume that each rigid designator symbol c has a canonical interpretation as the object $[[c]]$ of the universe.

Formally, given a structure S, $S \models \forall x\ \alpha(x)$ where x occurs in a meta-predicate in α iff $S \models \alpha(c/x)$ for every c that is a rigid designator of the type of x in S. If x does not occur in any meta-predicate in α, we use the usual FOL semantics of quantification. Given a theory T, we denote semantic entailment of a formula α with this semantics as as $T \models_{RD} \alpha$.

$$\forall \textit{PersonRef y, Timestep t } (\exists \textit{Person x } (\textit{atScanner(x, t)} \wedge \textit{record(x, t)=y}) \leftrightarrow \textit{new(y, t)}) \quad (2)$$

$$\forall \textit{Person x, x' PersonRef y, Timestep t,t' } (\textit{record(x, t)=y} \wedge \textit{record(x',t')=y} \rightarrow \textit{x=x'} \wedge \textit{t=t'}) \quad (3)$$

$$\forall \textit{PersonRef y, Timestep t, Real z } (\textit{new(y, t)} \wedge \exists \textit{Person x } (\textit{atScanner(x, t)} \wedge \textit{Height(x,t)>50} \wedge \textit{record(x,t)=y}) \wedge \textit{HeightSensorOnline(t)} \leftrightarrow \textit{observable(MsdHtOfSrc(y), t)}) \quad (4)$$

$$\forall \textit{PersonRef y, Timestep t } (\textit{new(y, t)} \wedge \exists \textit{Person x } (\textit{atScanner(x,t)} \wedge \textit{record(x,t)=y}) \leftrightarrow \textit{known(y,t)}) \quad (5)$$

$$\forall \textit{PersonRef y, Timestep t } (\textit{known(y,t)} \leftrightarrow \textit{doable(takeSnapshotOfSrc(y),t)}) \quad (6)$$

Figure 1: FOL formulas modelling a sensor, an actuator and communication between them (see example 1).

3.4 Properties of the Proposed Framework

A few properties of the proposed framework are presented below. First, we note that the framework developed above is sound and complete.

Theorem 1 *$T \vdash_{RD} \alpha$ iff $T \models_{RD} \alpha$.*

Proof The result follows by soundness and completeness of typed first-order logic. We can translate the formulation above into typed first-order logic, where rigid designator objects for each sensor constitute a distinct type and have unique names. In the translation, quantifiers over variables that occur in meta-predicates range over the objects of the type corresponding to rigid designators of the appropriate sensors or actuators. Syntactic and semantic definitions of entailment described above are equivalent to the corresponding notions for typed quantifiers. Then, soundness and completeness follows by soundness and completeness of typed first-order logic. □

This framework addresses all the fundamental problems raised towards the beginning of Sec. 3.1. In an observation *MeasuredHeight(p_{17})=171cm*, the soundness and completeness results above show that the results on an agent's belief will be the same regardless of whether p_{17} is treated as a rigid-designator symbol in the language or a rigid-designator object in the universe. Furthermore, since p_{17} must be a rigid designator, it must occur in the agent's belief state. The next theorem presents conditions under which a generative model will generate only valid sensori-motor streams, ensuring that the model conforms to specifications.

Theorem 2 *Let $\mathcal{S}$ be a set of sensor specifications, $\mathcal{A}$ a set of actuator specifications, and $\mathcal{SA}$ a set of sensori-motor stream specifications over a subset of $\mathcal{S} \times \mathcal{A}$. Suppose the model includes the following formulas:*

- $\text{doable}(a_k(x_1, \ldots x_m), t) \rightarrow \wedge_{k=1\ldots m} \vee_{\{S_j:S_jA_i \in \mathcal{SA}\}} \text{known}_{S_jA_i}(x_k, t)$.
- $\text{observable}(p(x_1, \ldots, x_m), t) \rightarrow \wedge_{k=1\ldots m} \exists t'\ new(x_k, t') \wedge t' \leq t$.
- $\text{known}(y, t) \rightarrow \exists t'\ new(y, t') \wedge t' \leq t$.

Then the set of observable, new, known *and* doable *statements that are provable in the theory will correspond to a subset of the union of the sensori-motor streams $\mathcal{SA}$.*

Proof The requirements for a sensori-motor stream are that arguments for known and observable be declared using *new* and that action arguments be communicated from a coordinating sensor. Stated conditions ensure this. □

In a probabilistic setting the consequent of Th. 2 continues to hold when the logical implications in the premises are made to hold with probability 1. E.g., $P(\wedge_{k=1\ldots m} \vee_{\{S_j:S_jA_i \in \mathcal{SA}\}} \textit{known}_{S_jA_i}(x_k, t) \mid \textit{doable}(a_k(x_1, \ldots x_m), t)) = 1$. These requirements can be included in a generative first-order model using defined predicates. The next simple yet significant result shows that if an object is referred to by multiple terms, even in our framework of meta-predicates that distinguish between such terms, the result of an action on any of them leads to the same object-level belief state.

Theorem 3 *Let t_1 and t_2 be terms of type τ and a an action whose argument is of type τ. Suppose the agent believes with certainty that $t_1 = t_2$. If doable($a(t_1)$) and doable($a(t_2)$) are both true in the agent's belief then the belief state over object-level predicates after $a(t_1)$ will be the same as the belief state obtained after applying $a(t_2)$.*

Proof In every possible structure in a belief state, interpretations of t_1 and t_2 are identical; a formula holds for t_1 iff it holds for t_2. Since the effects of action application are computed by evaluating FO formulas, the effects of $a(t_1)$ will be identical to that for $a(t_2)$ for all object-level predicates in the universe. □

4 Examples

In the previous section we showed how syntactic rules of inference defined above correspond with certain semantic notions. We now demonstrate using examples that these semantic notions correspond with desirable intuitive interpretations of sensor and actuator models.

Example 3 Let S_{skolem} be a sensor that returns a sequence of Lengths denoting heights of people (*Length* objects are positive reals attached with units, e.g. "171cm".). S_{skolem} can be modeled as $\forall Person\ x, Length\ y\ Height(x) = y \wedge atScanner(x) \leftrightarrow observable(\exists z\ MeasuredHeight(z) = y)$. In a probabilistic model, the observed value of y may include error and can be represented in general in terms of a conditional distribution that takes as its arguments each of the conjuncts on the other side of the bi-implication. The only variable in *observable*() in this expression is y; y can be substituted with any Length. Thus, we may substitute 171cm for y to obtain the fact that an observation $\exists z\ MeasuredHeight(z) = 171cm$ is received iff there is someone with height 171cm at the scanner. Such a model can also represent a scanner generating a stream of fingerprints which can be used in conjunction with a database to initiate actions when a match is (or isn't) detected.

Example 4 Let S_{Person} be a sensor that returns a mapping from objects of type Person to their heights. S_{Person} can be modelled as $\forall Person\ x\ atScanner(x) \leftrightarrow observable(Height(x))$. Instead of *PersonRef* as used in the running example, x's type is *Person* in this example. However, *Person* is not a predefined type like *Length* with a standard interpretation. Thus, x can be substituted with any constant of type *Person* that this sensor generated or was initialized with. Further, if the sensor returns an observation $Height(p_{17})=171cm$ then p_{17} is a rigid designator of type *Person* and every time the object represented by p_{17} is at the scanner, the scanner must be able to re-identify it and return a height observation. This is a natural consequence of two facts: the specification that the scanner returns symbols of type Person and the observability condition specified above. In practice, it is unusual to find scanners that return rigid designators for Persons. More realistic sensors can be specified using other type specifications. This example indicates that if a sensor cannot identify elements of type τ_1, it cannot return mappings from τ_1 to τ_2.

Finally, we illustrate that this system can be used to construct search trees for FOPOMDPs in a manner consistent with given sensor-actuator specifications. A search tree for a POMDP is a bipartite tree consisting of action nodes and observation nodes with an action node as the root. Each edge from an action node represents a possible action; each edge from an observation node n_O represents an observation that may be received when the action leading to n_O is executed. Solution techniques for POMDPs use search trees to evaluate the best action at each node of the tree, given a belief state at which one of the actions stemming from the root node must be applied. Fig. 2 shows an example of the search tree for the running example. For this illustration we assume that the symbols $P1, \ldots, Pk$ have been declared initially and that the action *sendToScanSrc(Pi, t)* can be performed on any of these symbols. This action has effect of the person represented by Pi being at the scanner with a high probability; it is also possible that this person could not go to the scanner and instead we receive an observation of another person at the scanner. The left-most branch from the observation node represents the high-probability outcome. Following every branch, the rules of inference imply that only one particular *takeSnapshotOfSrc(Pi)* action can be executed. Every path from root to a leaf in this tree constitutes a valid sensori-motor stream.

5 Related Work and Conclusions

Our formulation of action effects uses update rules similar to successor state axioms proposed by Reiter [2001]. However, usually employed assumptions like having a "closed initial database" in that line of work preclude the possibility of expressing identity uncertainty: distinct terms like *Mary* and *Mother(Johnny)* can never represent the same object. Sanner and Kersting [2010] present a framework for first-order POMDPs but under the assumption that all non-fluent terms are fully observable. This disallows uncertainty about statements like Mary=Mother(Johnny), which constitutes one of the key representational problems addressed in this paper. Sanner and Kersting suggest a *same-as*(t_1, t_2) predicate for representing identity uncertainty between fluent terms. However, it is not clear how this predicate can be used in conjunction with their unique names axioms for actions, which assert that instances of an action applied on distinct terms must be distinct. Furthermore, if the *same-as* predicate is used, a single object may be referred to by multiple terms, only one of which may be used by a sensor—thus leading to the problems addressed in this paper. Wang and Khardon [2010] present a relational representation for POMDPs while making the unique names and closed world assumptions: in their framework, action arguments have to be objects of the universe (implying that every object has a unique name in the language) and observations report properties of such objects.

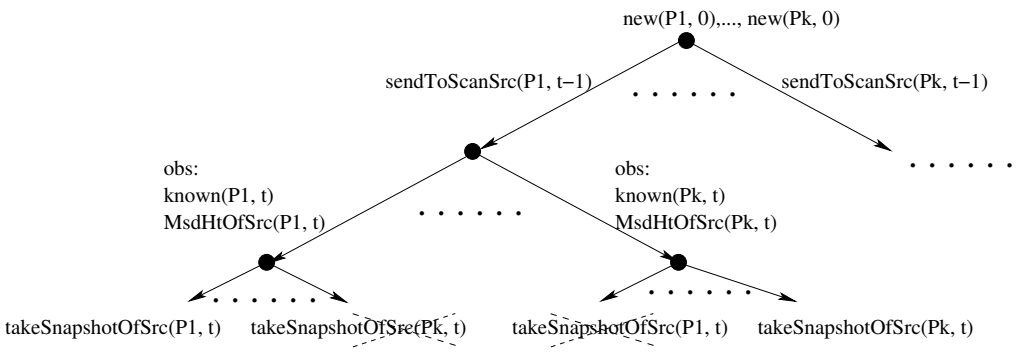

Figure 2: A 2-step search tree. Actions that are not doable are striked out.

To the best of our knowledge, Moore [1985] presented the earliest comprehensive FOL formulation of actions that did not make the unique names assumption and allowed terms in the language to be partially observable. In Moore's formulation actions can be executed by an agent only if they are "known" to it; an agent knows an action if its arguments are substituted by rigid designators. This notion of epistemic feasibility of an action was also used in later work by Morgenstern [1987] and Davis [1994, 2005]. These approaches use a significantly larger axiomatization to address the problem of syntactically proving and communicating facts about knowledge. In contrast, the main contributions of our work are on the unaddressed problems of expressing observability and action availability in a sound and complete framework that does not make unique-names or closed-world assumptions; conforms to a given sensori-motor specification; and addresses the fundamental question of how rigid designators are generated. These fundamental advances open the research frontier on developing autonomous agents that can plan and act on objects discovered through their sensors.

Acknowledgments

This research was supported in part by DARPA DSO grant FA8650-11-1-7153, DARPA contract W31P4Q-11-C-0083 and the NSF under grant number IIS-0904672.

References

Ernest Davis. Knowledge preconditions for plans. *J. Log. Comput.*, 4(5):721–766, 1994.

Ernest Davis. Knowledge and communication: A first-order theory. *Artif. Intell.*, 166(1-2):81–139, 2005.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Comput. Intell.*, 5(3):142–150, December 1989.

Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

Roderick J. A. Little and Donald B. Rubin. *Statistical analysis with missing data (second edition)*. Wiley, 2002.

Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic models with unknown objects. In *Proc. of IJCAI*, pages 1352–1359, 2005.

Robert C Moore. *A Formal Theory of Knowledge and Action*, pages 319–358. Ablex, 1985.

Leora Morgenstern. Knowledge preconditions for actions and plans. In *Proc. of IJCAI*, pages 867–874, 1987.

Raymond Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. The MIT Press, Massachusetts, MA, 2001.

Stuart J. Russell and Peter Norvig. *Artificial intelligence - a modern approach: the intelligent agent book*. Prentice Hall, 1995.

Scott Sanner and Craig Boutilier. Practical solution techniques for first-order MDPs. *Artif. Intell.*, 173(5-6):748–788, 2009.

Scott Sanner and Kristian Kersting. Symbolic dynamic programming for first-order POMDPs. In *Proc. of AAAI*, 2010.

Chenggang Wang and Roni Khardon. Relational partially observable MDPs. In *Proc. of AAAI*, 2010.