Symbolic Dynamic Programming for First-order POMDPs

Scott Sanner
NICTA & ANU
Canberra, Australia
scott.sanner@nicta.com.au

Kristian Kersting
Fraunhofer IAIS
Sankt Augustin, Germany
kristian.kersting@iais.fraunhofer.de

Abstract

Partially-observable Markov decision processes (POMDPs) provide a powerful model for sequential decision-making problems with partially-observed state and are known to have (approximately) optimal dynamic programming solutions. Much work in recent years has focused on improving the efficiency of these dynamic programming algorithms by exploiting symmetries and factored or relational representations. In this work, we show that it is also possible to exploit the full expressive power of first-order quantification to achieve state, action, *and* observation abstraction in a dynamic programming solution to relationally specified POMDPs. Among the advantages of this approach are the ability to maintain compact value function representations, abstract over the space of potentially optimal actions, and automatically derive compact conditional policy trees that minimally partition relational observation spaces according to distinctions that have an impact on policy values. This is the first lifted relational POMDP solution that can optimally accommodate actions with a potentially infinite relational space of observation outcomes.

Introduction

Partially-observable Markov decision processes (POMDPs) are known to be a powerful modeling formalism for sequential decision-making problems with partially observed state. Important recent applications include clinical decision-making, dialog management, and control policies for robots. However, this power does not come without its drawbacks and large POMDPs are also notoriously difficult to solve even approximately optimally.

To address these difficulties, recent work has explored the use of factored propositional or relational representations and algorithms that can exploit these representations (Wang and Schmolze 2005; Wang 2007; Wang and Khardon 2010; Shani et al. 2008). Other recent work has sought to exploit symmetries in the structure of the problem or solution (Doshi and Roy 2008; Kim 2008). However, no current work has fully exploited the power of first-order abstraction in POMDPs with a potentially infinite relational observation space in a way that is independent of ground domain size.

In this paper, we provide an extension of Boutilier *et al.*'s results on first-order MDPs (Boutilier, Reiter, and Price 2001) and Wang *et al.*'s results on relational POMDPs (Wang and Schmolze 2005; Wang 2007; Wang and Khardon 2010) to first-order POMDPs with actions that have a potentially infinite relational space of observation outcomes. Specifically, we provide the first lifted POMDP solution that can automatically derive *relevant quantified observations* leading to a solution whose complexity is independent of the size of the ground state, action, *and* observation spaces.

We proceed as follows: after reviewing POMDPs, we discuss the need for first-order POMDPs, formalize them, and provide a lifted solution via symbolic dynamic programming (SDP). We empirically show the complexity of SDP is invariant to domain size while enumerated state POMDP solvers have complexity exponential in the domain size.

Partially Observable MDPs

We assume familiarity with MDPs (Puterman 1994) and POMDPs (Kaelbling, Littman, and Cassandra 1998) and thus provide only a brief review. A Markov decision process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, h \rangle$ (Puterman 1994). $\mathcal{S} = \{s_1, \ldots, s_n\}$ is a finite set of states. $\mathcal{A} = \{a_1, \ldots, a_m\}$ is a finite set of actions. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a known stationary, Markovian transition function, often written $P(s'|s,a)$ for $s', s \in \mathcal{S}$ and $a \in \mathcal{A}$. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is a fixed known reward function associated with every state and action. γ is a discount factor s.t. $0 \leq \gamma \leq 1$ where rewards t time steps in the future are discounted by γ^t. $h \geq 0$ is a horizon, possibly infinite ($h = \infty$), indicating how many decision stages there are until termination of the process.

A partially observable MDP (POMDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \mathcal{Z}, \gamma, h \rangle$. In addition to the definitions for an MDP, a POMDP introduces an observation set $\mathcal{O} = \{o_1, \ldots, o_p\}$ and a known observation function $\mathcal{Z} : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \rightarrow [0, 1]$ often written as $P(o|a, s')$ for observation $o \in \mathcal{O}$, action $a \in \mathcal{A}$, and next state $s' \in \mathcal{S}$.

In a POMDP, the agent does not directly observe the states and thus must maintain a belief state $b(s) = P(s)$. For a given belief state $\mathbf{b} = b(\cdot)$, a POMDP policy π can be represented by a tree corresponding to a conditional plan β. An h-step conditional plan β^h can be defined recursively in terms of $(h-1)$-step conditional plans as shown in Fig. 1.

Our goal is to find a policy π that maximizes the value function, defined as the sum of expected discounted rewards

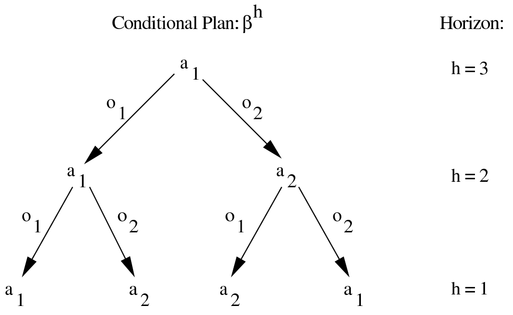

Figure 1: Example conditional plan β^h for POMDP control.

over horizon h starting from initial belief state $\mathbf{b}$:

$$V_\pi^h(\mathbf{b}) = E_\pi\left[\sum\nolimits_{t=0}^{h} \gamma^t \cdot r_t \Big| \mathbf{b}_0 = \mathbf{b}\right] \qquad (1)$$

where r_t is the reward obtained at time t and $\mathbf{b}_0$ is the belief state at $t = 0$. For finite h and belief state $\mathbf{b}$, the optimal policy π takes the form of an h-step conditional plan β^h. For $h = \infty$, the optimal discounted ($\gamma < 1$) value function can be approximated arbitrarily closely by using a sufficiently large, but finite h (Kaelbling, Littman, and Cassandra 1998).

While the number of belief states is infinite, it is well-known that the optimal POMDP value function for finite horizon h is a piecewise linear and convex function of the belief state $\mathbf{b}$ and that V^h can be represented as a maximization over a finite set of "α-vectors" α_i^h:

$$V^h(\mathbf{b}) = \max\nolimits_{\alpha_i^h \in \Gamma^h} \alpha_i^h \cdot \mathbf{b} \qquad (2)$$

Here, $\mathbf{b}$ is a belief state of dimension $|\mathcal{S}|$ (note that $\mathbf{b}$'s last dimension is determined by the other $|\mathcal{S}| - 1$ dimensions) and each $\alpha_i^h \in \Gamma^h$ is likewise of dimension $|\mathcal{S}|$.

The Γ^h in this optimal h-stage-to-go value function can be computed via the dynamic programming algorithm of *value iteration* (VI) (Sondik 1971). We initialize $\alpha_1^0 = \mathbf{0}$ and $\Gamma^0 = \{\alpha_1^0\}$. Then, we can inductively compute Γ^h from Γ^{h-1} in a dynamic programming *backup* operation:[1]

$$\begin{aligned} g_{a,o,j}^h(s) &= \sum_{s'} P(o|s',a)P(s'|s,a)\alpha_j^{h-1}(s'); \forall \alpha_j^{h-1} \in \Gamma^{h-1} \\ \Gamma_a^h &= R(\cdot,a) + \gamma \boxplus_{o \in \mathcal{O}} \left\{ g_{a,o,j}^h(\cdot) \right\}_j \qquad (3) \\ \Gamma^h &= \bigcup_a \Gamma_a^h \end{aligned}$$

From Γ^h, it is easy to determine the optimal h-step conditional plan β^h starting from a belief state $\mathbf{b}$. To do this, we simply find the maximizing α-vector $\alpha^* = \arg\max_{\alpha_i^h \in \Gamma^h} \alpha_i^h \cdot b$. For $h > 0$, α^* must have been generated as the $\boxplus_o$ of $g_{a,o,j}^h(\cdot)$'s for a given action a. Then the optimal action at step h is a, and once we observe o, we know that the optimal $h-1$-step conditional plan β^{h-1} is given by the $\alpha_j^{h-1} \in \Gamma^{h-1}$ that corresponds to $g_{a,o,j}^h(\cdot)$.

First-order (FO) Partially Observable MDPs

The Need for First-order Representations

While dynamic programming provides a provably optimal solution to POMDPs, we note that the number of α-vectors grows exponentially on each backup operation, i.e., $|\Gamma^h| = |\mathcal{A}||\Gamma^{h-1}|^{|\mathcal{O}|}$ (the factor of $|\Gamma^{h-1}|^{|\mathcal{O}|}$ is incurred during the $\boxplus$ operation). Even with pruning algorithms for removing provably suboptimal α-vectors in $|\Gamma^h|$, exact dynamic programming remains intractable even for small h when the observation space $\mathcal{O}$ is large (Spaan and Vlassis 2005).

For this reason, most conventional approaches to solving POMDPs use a carefully defined observation set $\mathcal{O}$ that has been tuned or even artificially restricted to guarantee tractability of the solution. However, as we show in this paper, due to the structure of many natural POMDP formulations, it is not always necessary to limit oneself to small or carefully defined observation spaces in order to ensure a tractable solution. In fact, in this paper, we encourage precisely the opposite: we allow *rich relational observation spaces* that define all possible observations and relations between them. The dynamic programming algorithm should be intelligent enough to *automatically derive only the relevant abstracted observations*.

As a simple illustrative example, we modify the classical TIGER POMDP (Kaelbling, Littman, and Cassandra 1998).

Example 1 (FO-TIGER) *An agent is in the center of a large circular room faced with a choice of doors $d_1, \ldots, d_n$. Behind each door d_i is food, but also possibly a noisy tiger denoted by* state $Tiger_S(d_i)$*. The state of $Tiger_S(d_i)$ does not change and cannot be directly observed. If the agent decides on action $open_S(d_i)$ and $\neg Tiger_S(d_i)$ holds, she receives reward* 10*, otherwise if $Tiger_S(d_i)$ holds, she receives reward -100. No observations are made on an $open_S(d_i)$ action, however, the agent can make observations on a $listen_S$ action (costing -1) where for each d_i, either the* observation $Noise_O(d_i)$ *or $\neg Noise_O(d_i)$ is made; 30% of the time a noise seems to come from all doors. For simplicity, we assume h is finite and thus set discount $\gamma = 1$.*

We note that for a problem with n doors, this problem has n possibilities for the $open_S$ action, but more alarmingly, $|\mathcal{O}| = 2^n$ possible observations for the $listen_S$ action! There is an observation for each door indicating whether or not a noise was heard. Clearly, if value iteration for conventional POMDPs has complexity $|\Gamma^h| = |\mathcal{A}||\Gamma^{h-1}|^{|\mathcal{O}|}$ then this problem cannot be solved in reasonable space and time for even small h using enumerated observations.

Obviously, the key to a succinct solution for FO-TIGER is to avoid working with a fully enumerated state, action, and observation space and to exploit symmetries and homomorphisms — in this paper through the use of logical abstraction — to find lifted abstractions that can be treated identically without loss of value. In our following formalization and solution to this problem, we will demonstrate that despite a very large (or even infinite) observation space, our *first-order* (FO) POMDP representation and solution achieves these goals by deriving a small set of first-order abstracted conditional policies that permit an efficient and compact finite horizon solution to POMDP problems like FO-TIGER.

[1] The $\boxplus$ of sets is defined as $\boxplus_{j\in\{1,\ldots,n\}} S_j = S_1 \boxplus \cdots \boxplus S_n$ where the pairwise cross-sum $P \boxplus Q = \{\mathbf{p} + \mathbf{q} | \mathbf{p} \in P, \mathbf{q} \in Q\}$. The sum of a scalar and a set is defined as the elementwise sum: $\mathbf{v} + \{\mathbf{u}_1, \ldots, \mathbf{u}_k\} = \{\mathbf{v} + \mathbf{u}_1, \ldots, \mathbf{v} + \mathbf{u}_k\}$.

Logical Foundations of FO-POMDPs

We assume basic familiarity with many-sorted first-order logic with equality (Enderton 1972) where for FO-TIGER we need sorts for *Door* and *Situation* (sorts will be obvious from context). The *situation calculus* is a first-order language for axiomatizing dynamic worlds consisting of actions, situations and fluents and specified in a domain theory:

Actions are first-order terms consisting of an action function symbol and its arguments. In FO-POMDPs, there are two types of actions, *transition* actions such as $open_S(d_i)$ and $listen_S$ and *observation* actions such as $listen_O$.

Situations are first-order terms denoting a sequence of actions. These are represented using a binary function symbol do: $do(\alpha, s)$ (and inverse $do^{-1}(\alpha, s)$) denotes the situation after (before) performing the action α starting in state s; e.g., $do(listen_S, do(open_S(d_i), s))$ denotes the situation resulting from opening door d_i then listening. In contrast to states, situations reflect the entire history of action occurrences. However, the FO-POMDP dynamics are Markovian and allow recovery of state properties from situation terms.

Fluents are relations whose truth values vary from state to state and are denoted by predicate symbols whose last argument is a situation term. In FO-POMDPs, there are two types of fluents, **state fluents** such as $Tiger_S(d_i, s)$ and **observation fluents** such as $Noise_O(d_i, s)$.[2]

We assume that all *non-fluent* predicates are *fully observed*, e.g., this includes sorts and equality (=). Similarly, we assume that all terms (including constants) are likewise fully observed. All fully observed predicates may appear in either the state or observation description, thus the observation relations may refer directly to the Herbrand universe of terms used in the state relations.

If *identity uncertainty* (uncertainty over equality of terms t_1 and t_2) is required, it may be formalized via uncertainty over a *Same-as*(t_1, t_2, s) fluent. The binary equality predicate $=$ is not situation dependent and thus fully observed.

Domain theory: A domain theory without action preconditions or an initial database[3] is axiomatized in the situation calculus using two classes of axioms (Reiter 2001):

- **Successor state axioms (SSAs):** There is one such axiom for each fluent $F(\mathbf{x}, s)$ with syntactic form:

  $$F(\mathbf{x}, do(a, s)) \equiv \Phi_F(\mathbf{x}, a, s),$$

  where $\Phi_F(\mathbf{x}, a, s)$ is a formula with free variables among $a, s, \mathbf{x}$. These characterize the truth values of the fluent F in the next situation $do(a, s)$ in terms of the current situation s. While SSAs are typically compiled from *effect axioms* and embody a solution to the frame problem (Reiter 2001), we bypass this compilation step in this paper and directly provide the SSAs for brevity.
- **Unique names axioms:** These state that the action terms of the domain are all pairwise unequal.

Regression and Progression The *regression* of a formula ψ through an action a is a formula ψ' that holds prior to a being performed iff ψ holds after a. Suppose that fluent F's SSA is $F(\mathbf{x}, do(a, s)) \equiv \Phi_F(\mathbf{x}, a, s)$. We inductively define the regression of a formula whose situation arguments all have the form $do(a, s)$ as follows:

$$\begin{aligned}
Regr(F(\mathbf{x}, do(a, s))) &= \Phi_F(\mathbf{x}, a, s) \\
Regr(\neg\psi) &= \neg Regr(\psi) \\
Regr(\psi_1 \wedge \psi_2) &= Regr(\psi_1) \wedge Regr(\psi_2) \\
Regr((\exists x)\psi) &= (\exists x) Regr(\psi)
\end{aligned}$$

The inverse of regression is *progression*, which takes a formula ψ and an action o and represents the formula ψ' that would hold after action o is performed in a situation satisfying ψ. Unlike regression, progression is *only* well-defined for certain language restrictions of effect axioms. There are a variety of such language restrictions, each leading to different expressiveness (Reiter 2001; Vassos, Lakemeyer, and Levesque 2008). In brief, *any* progression algorithm that takes a finite formula ψ as input and generates a progressed finite formula ψ' as output can be used in FO-POMDPs. For a simple example of a finitely progressable theory that suffices for FO-Tiger in this paper, we use a subset of the progressable *context-free* effects model (Reiter 2001). In brief, we assume every pre-action fluent $F_1(\mathbf{x}, do^{-1}(o, s))$ corresponds to a unique post-action fluent $F_{2,o}(\mathbf{x}, s)$. Then with effect axiom pairs of the form

$$F_1(\mathbf{x}, do^{-1}(a, s)) \wedge a = o \supset (\neg)F_{2,o}(\mathbf{x}, s) \qquad (4)$$

$$\neg F_1(\mathbf{x}, do^{-1}(a, s)) \wedge a = o \supset (\neg)F_{2,o}(\mathbf{x}, s) \qquad (5)$$

where $(\neg)$ allows possible negation, Reiter (2001) provides a progression operator $Prog(\psi(do^{-1}(o, s)))=\psi'(s)$ that produces a post-action formula $\psi'(s)$ that must hold in s if the pre-action formula $\psi(do^{-1}(o, s))$ holds prior to the action. The algorithm for *Prog* is too long to reproduce here, but in this paper, progressions of our examples will be self-evident from the effect axioms.

Representation of FO-POMDPs

An FO-POMDP can be viewed as a universal POMDP that abstractly defines the state, action, observations, transition, reward and observation function tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{R}, \mathcal{Z} \rangle$ for all possible domain instantiations, e.g., we desire an FO-POMDP for FO-TIGER that holds for *any* number of doors.

In this section, we draw on the formalization of first-order MDPs (Boutilier, Reiter, and Price 2001) and extend this to first-order POMDPs by defining an observation function.

Case Representation of Reward, Value, & Probability: We use a tabular *case notation* along with its logical definition to allow first-order specifications of the rewards, probabilities, and values required for FO-POMDPs:

$$t = \begin{array}{|c|}\hline \phi_1 : t_1 \\ \hline : \;\; : \;\; : \\ \hline \phi_n : t_n \\ \hline \end{array} \;\equiv\; \left(\bigvee_{i \leq n} \{\phi_i \wedge t = t_i\} \right) \qquad (6)$$

Here the ϕ_i are *state formulae* where fluents in these formulae do not contain the term *do* and the t_i are constant terms. As an example, we present the FO-TIGER reward:

$$R(s, a) = \begin{array}{|ll|}\hline \exists d_j . a = open_S(d_j) \wedge \neg Tiger_S(d_j, s) : & 10 \\ \hline \exists d_j . a = open_S(d_j) \wedge Tiger_S(d_j, s) & :-100 \\ \hline a = listen_S & : \;\; -1 \\ \hline \end{array} \qquad (7)$$

[2] State fluents are subscripted with $_S$ and observation fluents with $_O$. Also note we have augmented the predicates in the FO-TIGER description with situation terms s since they are fluents.

[3] We assume all actions are executable in all situations and we do not require knowledge of the initial state.

This formalizes the previously described FO-Tiger reward obtained when executing an action in a state. The case representation can also be used to specify the value function and transition probabilities, but first we discuss case operations.

Case Operations We begin by describing the following binary $\otimes$, $\oplus$, and $\ominus$ operators on case statements (Boutilier, Reiter, and Price 2001). Letting each ϕ_i and ψ_j denote generic first-order formulae, we can perform the "cross-sum" $\oplus$ of case statements in the following manner:

$$\begin{array}{|c|}\hline \phi_1 : 10 \\ \hline \phi_2 : 20 \\ \hline \end{array} \;\oplus\; \begin{array}{|c|}\hline \psi_1 : 1 \\ \hline \psi_2 : 2 \\ \hline \end{array} \;=\; \begin{array}{|c|}\hline \phi_1 \wedge \psi_1 : 11 \\ \hline \phi_1 \wedge \psi_2 : 12 \\ \hline \phi_2 \wedge \psi_1 : 21 \\ \hline \phi_2 \wedge \psi_2 : 22 \\ \hline \end{array}$$

Likewise, we can perform $\ominus$ and $\otimes$ operations by, respectively, subtracting or multiplying partition values. Note that for a binary operation involving a scalar and a case statement, a scalar value C may be viewed as $\boxed{\top : C}$ where $\top$ is a tautology. We use the $\bigoplus$ and $\bigotimes$ operators to, respectively, denote summations and products of multiple case operands. Some resulting partitions may be inconsistent (e.g., $\phi \wedge \neg\phi$) and pruned, but pruning is only necessary for compactness.

Stochastic Actions There are two types of actions in FO-POMDPs: **transition actions** and **observation actions**. Transition actions such as $\mathit{open}_S(d_i)$ and listen_S are explicitly executed by the "agent".[4] If a transition action generates non-nil observations then it must have an associated observation action with the same parameters (e.g., listen_O is associated with listen_S) that is executed implicitly after the transition action is executed. As usual for POMDPs, transition and observation outcomes may be stochastic.

Stochastic actions (either transition or observation) can be decomposed into a *collection* of deterministic actions, each corresponding to a possible outcome of the stochastic action. We then use a case statement to specify a distribution according to which "Nature" may choose a deterministic action from this set whenever the stochastic action is executed. Formally, letting $A(\mathbf{x})$ be a stochastic action with Nature's choices (i.e., deterministic actions) $n_1(\mathbf{x}), \cdots, n_k(\mathbf{x})$, we represent the probability of $n_i(\mathbf{x})$ given $A(\mathbf{x})$ is executed in s by $P(n_j(\mathbf{x}), A(\mathbf{x}), s)$. Next we provide an example.

Stochastic Observation Actions: As a major contribution of the paper, we propose to model the observation function in an FO-POMDP via a distribution over deterministic observation action outcomes. We crucially note that these observation actions are *not* under direct control of the agent, but rather correspond to possible deterministic ways in which the underlying state "generates" observations.

In FO-TIGER, we note that an observation action listen_O is executed whenever the agent executes transition action listen_S. For stochastic action listen_O, there are two deterministic outcomes: $\mathit{listenSucc}_O$ and $\mathit{listenFail}_O$. Recalling the FO-TIGER description, the agent's hearing fails 30% of the time on a listen_O (she hears noise at all doors on failure):

$$P(\mathit{listenSucc}_O, \mathit{listen}_O, s) = \boxed{\top : 0.7} \qquad (8)$$

$$P(\mathit{listenFail}_O, \mathit{listen}_O, s) = \boxed{\top : 0.3} \qquad (9)$$

Because we will progress the state through deterministic observation actions to obtain relevant observation descriptions, we need to define effect axioms for deterministic observation actions w.r.t. the previously stated requirements for progression. For FO-TIGER, our previously described special case of context-free effects leads us to the following axioms:

$$\mathit{Tiger}_S(d_i, do^{-1}(a,s)) \wedge a = \mathit{listenSucc}_O \supset \mathit{Noise}_O(d_i, s)$$

$$\neg\,\mathit{Tiger}_S(d_i, do^{-1}(a,s)) \wedge a = \mathit{listenSucc}_O \supset \neg\mathit{Noise}_O(d_i, s)$$

$$\mathit{Tiger}_S(d_i, do^{-1}(a,s)) \wedge a = \mathit{listenFail}_O \supset \mathit{Noise}_O(d_i, s)$$

$$\neg\,\mathit{Tiger}_S(d_i, do^{-1}(a,s)) \wedge a = \mathit{listenFail}_O \supset \mathit{Noise}_O(d_i, s)$$

The $\mathit{open}_S(d_i)$ action generates no observations (only nil).

Stochastic Transition Actions: There are two stochastic transition actions in the FO-TIGER POMDP: listen_S and $\mathit{open}_S(d_i)$. According to the FO-TIGER description, these actions are actually deterministic, so there is only one deterministic outcome for each, respectively $\mathit{listenSucc}_S$ and $\mathit{openSucc}_S(d_i)$. Then trivially, we obtain:

$$P(\mathit{listen}_S, \mathit{listenSucc}_S, s) = \boxed{\top : 1.0}$$

$$P(\mathit{open}_S(d_i), \mathit{openSucc}_S(d_i), s) = \boxed{\top : 1.0}$$

The state of fluent $\mathit{Tiger}_S(d_i, s)$ is persistent according to the FO-TIGER description, leading to the simple SSA:

$$\mathit{Tiger}_S(do(a,s)) \equiv \Phi_{\mathit{Tiger}_S}(s) \equiv \mathit{Tiger}_S(s) \qquad (10)$$

This completes the FO-TIGER FO-POMDP description. We necessarily keep the FO-TIGER domain simple to ensure its solution is self-contained in this paper. More complex examples of stochastic transtion actions compatible with our FO-POMDP definition are in Sanner and Boutilier (2009).

Here we do not consider the *factored* extension of FO-POMDPs as done for *factored FO-MDPs* (Sanner and Boutilier 2007; Sanner 2008) to permit the specification of stochastic actions with an indefinite number of independent outcomes. Such factored extensions are highly non-trivial and beyond the scope of this initial FO-POMDP work.

Symbolic Dynamic Programming

In this section, we define a symbolic dynamic programming (SDP) algorithm for computing solutions to FO-POMDPs in the form of lifted α-vectors. This SDP algorithm performs all derivations at a symbolic level that is independent of any particular set of domain objects and can accommodate actions with infinite relational observation outcome spaces. In this way, the algorithm may derive exact compact lifted representations of the value function in cases where doing so would be generally impossible for ground methods.

In our symbolic dynamic programming solution to FO-POMDPs, lifted α-vectors will be represented as parameterized case statements, e.g. $\alpha\mathit{Case}(\mathbf{x}, s)$. The parameters in these case statements will refer to the parameters $\mathbf{x}$ of *all* actions in the conditional policy represented by $\alpha\mathit{Case}(\mathbf{x}, s)$.

Our lifted POMDP value function will be represented as a maximization over a set $\Gamma^h = \{\alpha\mathit{Case}(\mathbf{x}, s)\}$:

$$V^h(s) = \max_\theta \max_{\alpha\mathit{Case} \in \Gamma^h} \mathbf{b} \cdot \alpha\mathit{Case}(\mathbf{x}, s)\theta \qquad (11)$$

Technically, belief state $\mathbf{b}$ assigns a probability to each ground state of an FO-POMDP (for a fixed instantiation

[4] For actions, $_S$ indicates transition and $_O$ indicates observation.

of domain objects encountered at execution-time). If $\mathbf{b}$ is sparse, one efficient way to calculate $\mathbf{b} \cdot \alpha Case(\mathbf{x}, s)\theta$ is to sum over the non-zero probability belief states in $\mathbf{b}$, multiplying each belief state probability by the weight of the partition of $\alpha Case(\mathbf{x}, s)\theta$ satisfied by that belief state. Alternately, if $\mathbf{b}$ is logically structured, Wang and Schmolz (2005; 2007) provide insights on the compact maintenance of belief states $\mathbf{b}$ and the efficient computation of $\mathbf{b} \cdot \alpha Case(\mathbf{x}, s)\theta$.

Now we need only specify how to derive Γ^h via symbolic dynamic programming, but as a crucial intermediate step, we discuss the automatic derivation of relevant observations $\mathcal{O}^h_A$ and distribution $P^h(\mathcal{O}^h_A|s, A)$ for action A at horizon h.

Automatically Deriving the Observation Function Unlike standard enumerated or factored POMDPs where there are a finite number of observation outcomes for an action, FO-POMDPs permit a potentially infinite relational observation space. However, given a policy where we execute transition action a then follow the conditional policy given by $\alpha Case(\mathbf{x}, s)$, we need only make the *relevant* observations that disambiguate future attainable values.

For example, in FO-TIGER, assume that we will execute $open_S(d_i)$ (as the only step of a 1-step conditional policy associated with $\alpha Case(\mathbf{x}, s)$) where there is a tiger potentially lurking behind d_i. Then a relevant observation for initial action $a = listen_S$ is $Noise_O(d_i, \cdot)$, but not $Noise_O(d_j, \cdot)$ for some $d_j \neq d_i$ (since knowing the latter would not help us determine whether $open_S(d_i)$ is a good action). Fortunately, given that an action a generates observations from states via observation actions *and* our observation action theory was required to be finitely progressable, we have a convenient way to automatically derive a *finite, minimal* partitioning of the relational observation space (and observation probabilities) relevant to distinguishing the value of action a followed by the conditional policy given by $\alpha Case(\mathbf{x}, s)$.

Let us clarify this important issue with an example, where we assume that at horizon $h = 1$, the 1-step conditional policy associated with $\alpha Case^1_{1,open_S}(d_i, s)$ is $open_S(d_i)$:

$$\alpha Case^1_{1,open_S}(d_i, s) = \begin{array}{|l|}\hline \neg Tiger_S(d_i, s) : \quad 10 \\ \hline Tiger_S(d_i, s) \quad : -100 \\ \hline \end{array} \qquad (12)$$

Clearly, the only relevant state partitions in this case are $\{\neg Tiger_S(d_i, s), Tiger_S(d_i, s)\}$. Next we assume that we will execute $listen_S$ at horizon $h = 2$. Then the observations and probabilities relevant to $\alpha Case^1_{1,open_S}(d_i, s)$ that can be observed on a $listen_S$ action are given by the progressions of the pre-observation state partitions of $\alpha Case^1_{1,open_S}(d_i, s)$ through the deterministic outcomes of $listen_O$:

$$Prog(Tiger_S(d_i, do^{-1}(listenSucc_O, s))) = Noise_O(d_i, s)[.7]$$
$$Prog(\neg Tiger_S(d_i, do^{-1}(listenSucc_O, s))) = \neg Noise_O(d_i, s)[.7]$$
$$Prog(Tiger_S(d_i, do^{-1}(listenFail_O, s))) = Noise_O(d_i, s)[.3]$$
$$Prog(\neg Tiger_S(d_i, do^{-1}(listenFail_O, s))) = Noise_O(d_i, s)[.3]$$

Here in brackets, we also indicate the probabilities of each deterministic observation outcome from (8) and (9). Progressing the relevant state from $\alpha Case^1_1(d_i, s)$ to obtain relevant observations and taking the cross-product of these observations, we obtain relevant observation space $\mathcal{O}^2_{listen_S}$:

$$\mathcal{O}^2_{listen_S} = \{Noise_O(d_i, s), \neg Noise_O(d_i, s)\} \qquad (13)$$

From this we now explicitly write $P(\mathcal{O}^2_{listen_S}|s, listen_S)$ by summing probabilities in $[\cdot]$ above for the same observation outcomes generated from the same underlying state (for readability, we suppress the post-observation situation):

$$P(Noise_O(d_i)|s, listen_S) = \begin{array}{|l|}\hline \neg Tiger_S(d_i, s) : 0.3 \\ \hline Tiger_S(d_i, s) \quad : 1.0 \\ \hline \end{array} \qquad (14)$$

$$P(\neg Noise_O(d_i)|s, listen_S) = \begin{array}{|l|}\hline \neg Tiger_S(d_i, s) : 0.7 \\ \hline Tiger_S(d_i, s) \quad : 0.0 \\ \hline \end{array} \qquad (15)$$

For correctness, note the sum over observations in each state

$$P(Noise_O(d_i)|s, listen_S) \oplus P(\neg Noise_O(d_i)|s, listen_S) = 1.0$$

We did not discuss parameterized observation actions since they were not needed for FO-TIGER, however, we note their treatment is no different than above. Hypothetically, if we allowed for a door-specific $listen_S(d_i)$ transition action then we might have an observation action $listen_O(d_i)$ with deterministic outcomes $listenSucc_O(d_i)$ and $listenFail_O(d_i)$. Then two context-free SSAs might be

$$Tiger_S(d_i, do^{-1}(a, s)) \wedge a = listenSucc_O(d_i) \supset Noise_O(d_i, s)$$
$$Tiger_S(d_j, do^{-1}(a, s)) \wedge a = listenSucc_O(d_i) \wedge d_i \neq d_j$$
$$\supset \neg Noise_O(d_i, s)$$

This fragment states we would only hear noise from a tiger behind door d_i if d_i *is* the door being listened to. In this way, we see that observation action outcomes can be restricted by transition action parameters, if needed. Progression of such context-free SSA theories according to Reiter (2001) to obtain relevant observations proceeds as above.

This completes our demonstration of how to derive relevant observations from an α-vector given transition action A_S. Generally, given a set of $\alpha Case$s, $\Gamma^h = \{\alpha Case^h_1, \ldots, \alpha Case^1_k\}$ at horizon h, one can derive *all* relevant observations $\mathcal{O}^h_{A_S}$ from the progressions of partitions of $\bigotimes_{i=1}^k \alpha Case^h_i$ though observation outcomes of A_O and for each $\phi \in \mathcal{O}^h_{A_S}$, derive corresponding case statement probabilities $P^h(\phi|s, A_S)$ as above.

Lifted Dynamic Programming with SDP As for dynamic programming in the ground case, we will obtain an iterative algorithm that starts with the $\mathbf{0}$ α-vector at zero decision stages to go and computes the optimal receding horizon control policy for successively larger horizons up to H. At every horizon of $h \leq H$ decisions to go, the optimal value function $V^h(s)$ will be of the form in (11) where Γ^h represents the set of α-vector cases ($\alpha Case$s) that comprise this optimal value function. This algorithm can be formally stated as follows (and will be followed by an example):

1. Set $\Gamma^0 = \{\boxed{\top : 0.0}\}$, set $h = 1$.
2. For Γ^h and each stochastic transition action $A(\mathbf{x})$, compute the relevant observation set $\mathcal{O}^h_A$ and for each $o \in \mathcal{O}^h_A$, derive $P^h(o|s, A)$ as previously described.

3. For each stochastic transition action $A(\mathbf{x})$ and each $o \in \mathcal{O}^h_A$, compute a $gCase^h_{A(\mathbf{x}),o,j}(s, \mathbf{y}_j)$ for each $\alpha Case^{h-1}_j \in \Gamma^{h-1}$ by summing over all Nature's choice state-changing action $n_k(\mathbf{x})$ outcomes for $A(\mathbf{x})$:

$$g^h_{A(\mathbf{x}),o,j}(s, \mathbf{y}_j) = \bigoplus_{n_k(\mathbf{x})} Regr(P^h(o|do(n_k(\mathbf{x}), s), A)) \otimes$$
$$P(n_k(\mathbf{x}), s, A(\mathbf{x})) \otimes Regr(\alpha Case^{h-1}_j(do(n_k(\mathbf{x}), s), \mathbf{y}_j))$$

   Here we note that $g^h_{A(\mathbf{x}),o,j}(s, \mathbf{y}_j)$ is parameterized not only by the action parameters of $A(\mathbf{x})$, but also by any (standardized-apart) action parameters $\mathbf{y}_j$ of the conditional policy for $\alpha Case^{h-1}_j$.

4. Compute the set $\Gamma^h_A = \{\alpha Case^h_{i,A}(\mathbf{y}, s)\}$ of all possible conditional policies for given initial action $A(\mathbf{x})$:[5]

$$\Gamma^h_A = R(s, A(\mathbf{x})) \oplus \gamma \boxplus_{o \in \mathcal{O}^h} \left\{ g^h_{A(\mathbf{x}),o,j}(s, \mathbf{y}_j) \right\}_j$$

   Here we let $\mathbf{y}$ be the concatenation of all action parameters $\mathbf{y}_j$ appearing in each of the $g^h_{A(\mathbf{x}),o,j}(s, \mathbf{y}_j)$.

5. Union the sets Γ^h_A for all stochastic transition actions A to represent the optimal form of $V^h(s)$ in (11):

$$\Gamma^h = \bigcup\nolimits_A \Gamma^h_A$$

6. If horizon $h < H$, set $h = h + 1$ and go to step 2.

SDP Example For $h = 0$, Γ^0 is given in step 1. Space limitations only permit us to present one conditional policy up to horizon $h = 2$. Clearly, one good $h = 2$ policy is to first $listen_S$ followed by $open_S(d_i)$ on a $\neg Noise_O(d_i)$ observation, otherwise $listen_S$ again on a $Noise_O(d_i)$. Hence we focus on generating this particular $h = 2$ conditional policy in the following example.

For $h = 1$, we derive Γ^1 starting with $open_S(d_i)$. There is no observation on $open_S(d_i)$ so $\mathcal{O}^1_{open_S} = nil$ in step 2. Because Γ^0 consists only of $\boxed{\top : 0.0}$, we trivially compute $g^h_{open_S(d_i)}(s) = \boxed{\top : 0.0}$ in step 3. Step 4 adds in the reward $R(s, a)$ given in (7) and after simplification w.r.t. $a = open_S(d_i)$ and the unique names axiom, we obtain the result $\alpha Case^1_{1,open_S}(d_i, s)$ already shown in (12). For $listen_S$, $\alpha Case^1_{1,listen_S}(s) = \boxed{\top : -1.0}$; intuitively, since the action has no effects or usable observations, we simply obtain the reward of $R(s, listen_S) = -1$ for all states. Thus $\Gamma^1 = \{\alpha Case^1_{1,open_S}(d_i, s), \alpha Case^1_{1,listen_S}(s)\}$.

For $h = 2$, since $\alpha Case^1_{1,open_S}(d_i, s)$ is the only α-vector in Γ^1 with partitions other than $\top$, we can derive $\mathcal{O}^2_{listen_S}$ precisely as outlined from (12) to (13). Doing steps 3 and 4:

$$\alpha Case^2_{1,listen_S}(d_i, s) = \boxed{\begin{array}{l} \neg T(d_i) : 5.7 \\ \hline T(d_i) \;\; : -1 \end{array}} = -1 \oplus \tag{16}$$
$$\underbrace{\boxed{\begin{array}{l} \neg T(d_i) : .7 \\ \hline T(d_i) \;\; : \;0 \end{array}} \otimes \boxed{\begin{array}{l} \neg T(d_i) : \;\; 10 \\ \hline T(d_i) \;\; : -100 \end{array}}}_{\neg Noise_O(d_i) \rightarrow open_S(d_i)} \oplus \underbrace{\boxed{\begin{array}{l} \neg T(d_i) : .3 \\ \hline T(d_i) \;\; : \;1 \end{array}} \otimes -1}_{Noise_O(d_i) \rightarrow listen_S}$$

For compactness we abbreviate $Tiger_S(d_i, s)$ as $T(d_i)$. In the calculation following the result, the first term $R(s, listen_S) = -1$, the second term is the observation probability $\neg Noise_O(d_i)$ in (15) times $\alpha Case^1_{1,open_S}(d_i, s)$, and the third term is observation probability $Noise_O(d_i)$ given in (14) times $\alpha Case^1_{1,listen_S}(s)$. We note that step 3 of SDP requires that the observation probabilities and $\alpha Case^1_1$ statements are regressed using the $Regr$ operator, but in the special case of FO-TIGER with transition action SSA given by 10, the state is persistent and $Regr$ is an identity operator.

Lifted Conditional Policies and Dominance In general, we note that since the optimal policy in a POMDP is history-dependent, we cannot directly quantify out the action variables $\mathbf{x}$ at each state as we would do for a state-dependent policy in first-order MDPs as observed by Wang *et al.* (2007; 2010). However, it is still possible to identify logical conditions on $\mathbf{x}$ that eliminate dominated instantiations of $\alpha Case(\mathbf{x})$ (recall that each $\alpha Case(\mathbf{x})$ is a lifted template for ground α-vectors and conditional policies obtained by instantiating $\mathbf{x}$). While we cannot provide a full exposition of dominance testing here, we do provide an example illustrating the power of lifted pointwise dominance testing.

Let us revisit (16); because the first action in this conditional policy is $listen_S$ and requires no action parameters, let us delay dominance pruning for action parameter d_i until *after* we have executed $listen_S$ and made the observation of $(\neg)Noise_O(d_i)$. Thus, we have two conditional policies:

$$\neg Noise_O(d_i) \rightarrow \alpha Case^1_{1,open_S}(d_i, s) = \boxed{\neg T(d_i) : 10}$$
$$Noise_O(d_i) \rightarrow \alpha Case^1_{1,listen_S}(s) = \boxed{\top : -1}$$

We note that $\alpha Case^1_{1,open_S}(d_i, s)$ does not have a partition for $T(d_i)$ since there is zero probability of observing $\neg Noise_O(d_i)$ in state $T(d_i)$.

To demonstrate lifted dominance pruning, we first define a new case binary comparison operator: $\geq$ is like the other binary operators $\oplus$ and $\otimes$ except that the terms of the resulting case statement are the boolean values $\top$ and $\bot$ representing the inequality evaluation.

Given this $\geq$ operator, we define a *pointwise dominance* test for $\alpha Case_1(\mathbf{x})$ over $\alpha Case_2(\mathbf{y})$ (non-situation variables in each $\alpha Case$ must be standardized apart), where $\mathbf{b}$ is a belief state probability vector ($\forall i\, \mathbf{b}_i \in [0, 1]$, $\sum_i \mathbf{b}_i = 1$):

$$\begin{aligned} &\forall \mathbf{b} \;\; [\mathbf{b} \cdot \alpha Case_1(\mathbf{x}) \geq \mathbf{b} \cdot \alpha Case_2(\mathbf{y})] = \top \\ \equiv &\forall \mathbf{b} \;\; [\mathbf{b} \cdot (\alpha Case_1(\mathbf{x}) \ominus \alpha Case_2(\mathbf{y})) \geq 0] = \top \\ \equiv &\; [\alpha Case_1(\mathbf{x}) \ominus \alpha Case_2(\mathbf{y}) \geq 0] = \top \\ \equiv &\; [\alpha Case_1(\mathbf{x}) \geq \alpha Case_2(\mathbf{y})] = \top \end{aligned} \tag{17}$$

We removed the dependence on $\mathbf{b}$ since potentially if each $\mathbf{b}_i > 0$, this is equivalent to checking that value dominance

[5] Here, the cross-sum of sets of case statements is defined as $\boxplus_j S_j = S_1 \boxplus \cdots \boxplus S_j$ where $P \boxplus Q = \{\mathbf{p} \oplus \mathbf{q} | \mathbf{p} \in P, \mathbf{q} \in Q\}$. The $\oplus$ of a case statement and a set is defined as the elementwise sum: $\mathbf{v} \oplus \{\mathbf{u}_1, \ldots, \mathbf{u}_k\} = \{\mathbf{v} \oplus \mathbf{u}_1, \ldots, \mathbf{v} \oplus \mathbf{u}_k\}$.

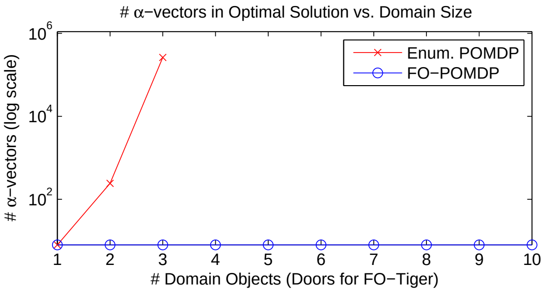

Figure 2: Comparison of SDP and enumerated DP on FO-Tiger.

holds in all mutually satisfiable partitions. Thus, pointwise dominance only holds for an instantiation of $\mathbf{x}$ and $\mathbf{y}$ if *all* consistent partitions of $\alpha Case_1(\mathbf{x}) \geq \alpha Case_2(\mathbf{y})$ are $\top$.

Applying this dominance test to our example, we obtain:

$$\alpha Case^1_{1,open_S}(d_i, s) \geq \alpha Case^1_{1,listen_S}(s) = \boxed{\neg T(d_i) \wedge \top : \top}$$

Since the only partition in the resulting case statement is $\top$, we see that after executing $listen_S$ in the conditional policy for $\alpha Case^2_{1,listen_S}(d_i, s)$ in (16), any instantiation of d_i s.t. $\exists d_i\, \neg Noise_O(d_i)$ strictly dominates all alternate instantiations of d_i where $Noise_O(d_i)$. Hence, lifted pointwise dominance analysis can prune suboptimal policies through the use of existential abstraction in the observation space.

Proof-of-concept Empirical Evaluation We solved FO-TIGER using both a simple implementation of our FO-POMDP SDP algorithm and Sondik's enumerated DP algorithm (Sondik 1971) for $h = 2$ (Sondik's algorithm could not solve $h = 3$ for more than 1 door). On a 2Ghz Intel Linux machine, SDP required less than 30s and produced 8 $\alpha Case$s; we limited Sondik's algorithm to 10 hours (we had to restrict to pointwise dominance for it to complete for 3 doors). In Figure 2, we compare the number of α-vectors generated in the optimal solution vs. the number of $\alpha Case$s (domain-size independent) generated in SDP. Even for small problems, SDP outperforms enumerated DP due to the large $2^{\#\text{Doors}}$ observation space size. For correctness, we note that each α-vector generated by DP corresponds to a ground instantiation of one of the $\alpha Case$s generated by SDP.

Correctness of SDP Wang *et al.* (2007; 2010) provide a proof of correctness of a related SDP algorithm for relational FO-POMDPs; since their FO-POMDP formalism restricts observations to predicates over the action parameters only, it remains to show that correctness is preserved in our more general observation framework. We note that a sketch of correctness is quite simple: if we inductively assume that the state partitions in the $\alpha Case$ statements distinguish all relevant values (this holds for the base case of Γ^0), then the observations need only provide a distribution over the $\alpha Case$ state partitions. This is precisely what is achieved via the use of progression to generate *relevant* observations. If we were to subdivide observation partitions, this would only give us more accurate information on states *within* the *same* $\alpha Case$ state partition; since these states would have the same value, no additional value would be distinguished in expectation.

Related Work and Concluding Remarks

Wang *et al.* (2007; 2010) have presented the only competing *relational* POMDP solution, however we note that their approach *cannot encode* the $listen_S$ action in FO-TIGER that generated an observation for *every* door d_i. We argue that our treatment of the observation space in FO-POMDPs is the first to fully exploit the full power of first-order techniques: we can now *automatically derive compact relevant first-order abstractions in large or potentially infinite observation spaces*. While future work needs to focus on practical enhancements of the SDP algorithm (e.g., approximation, point-based approaches), this work represents a crucial advance in scalable relational POMDP solutions.

Acknowledgements

We thank Roni Khardon for his clarifying observations on earlier versions of this work. NICTA is funded by the Australian Government's Backing Australia's Ability initiative, and the Australian Research Council's ICT Centre of Excellence program. This work was supported by the Fraunhofer ATTRACT fellowship STREAM.

References

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for first-order MDPs. In *IJCAI*, 690–697.

Doshi, F., and Roy, N. 2008. The permutable POMDP: fast solutions to POMDPs for preference elicitation. In *AAMAS*, 493–500.

Enderton, H. A. 1972. *A Mathematical Introduction to Logic*. New York: Academic Press.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*. (1-2): 99-134.

Kim, K.-E. 2008. Exploiting symmetries in POMDPs for point-based algorithms. In *AAAI*, 1043–1048.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: Wiley.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press.

Sanner, S., and Boutilier, C. 2007. Approximate solution techniques for factored first-order MDPs. In *(ICAPS-07)*, 288 – 295.

Sanner, S., and Boutilier, C. 2009. Practical solution techniques for first-order MDPs. *Artificial Intelligence*. 173:748–488.

Sanner, S. 2008. *First-order Decision-theoretic Planning in Structured Relational Environments*. Ph.D. Dissertation, University of Toronto, Toronto, ON, Canada.

Shani, G.; Brafman, R. I.; Shimony, S. E.; and Poupart, P. 2008. Efficient add operations for point-based algorithms. In *ICAPS*.

Sondik, E. J. 1971. *The Optimal Control of Partially Observable Markov Processes*. Ph.D. Dissertation, Stanford University.

Spaan, M. T. J., and Vlassis, N. 2005. Perseus: Randomized point-based value iteration for POMDPs. *JAIR* 24:195–220.

Vassos, S.; Lakemeyer, G.; and Levesque, H. 2008. First-order strong progression for local-effect basic action theories. In *KR*.

Wang, C., and Khardon, R. 2010. Relational partially observable MDPs. In *AAAI*.

Wang, C., and Schmolze, J. 2005. Planning with POMDPs using a compact, logic-based representation. In *IEEE ICTAI*, 523–530.

Wang, C. 2007. *First Order Markov Decision Processes*. Ph.D. Dissertation, Tufts University, USA.