Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)

Relational Partially Observable MDPs

Chenggang Wang and **Roni Khardon**
Department of Computer Science
Tufts University
Medford, MA, USA
{cwan|roni}@cs.tufts.edu

Abstract

Relational Markov Decision Processes (MDP) are a useful abstraction for stochastic planning problems since one can develop abstract solutions for them that are independent of domain size or instantiation. While there has been an increased interest in developing relational fully observable MDPs, there has been very little work on relational partially observable MDPs (POMDP), which deal with uncertainty in problem states in addition to stochastic action effects. This paper provides a concrete formalization of relational POMDPs making several technical contributions toward their solution. First, we show that to maintain correctness one must distinguish between quantification over states and quantification over belief states; this implies that solutions based on value iteration are inherently limited to the finite horizon case. Second, we provide a symbolic dynamic programing algorithm for finite horizon relational POMDPs, solving them at an abstract level, by lifting the propositional incremental pruning algorithm. Third, we show that this algorithm can be implemented using first order decision diagrams, a compact representation for functions over relational structures, that has been recently used to solve relational MDPs.

Introduction

Decision problems with objects and relations can in principle be solved by by grounding a given problem with concrete objects and using a propositional solver. However, the solver gets slower with increasing problem size, and the solutions it provides are problem specific and they cannot be reused. Recently there has been an increased interest in developing relational fully observable MDPs (RMDP) and abstract solutions for them that hold for any domain instance. Some examples include exact solutions (Boutilier, Reiter, and Price 2001; Kersting, Van Otterlo, and De Raedt 2004; Hölldobler and Skvortsova 2004; Wang, Joshi, and Khardon 2008), and several approximate or heuristic methods (e.g., (Fern, Yoon, and Givan 2006; Sanner and Boutilier 2009)).

Partially Observable Markov Decision Processes (POMDP) generalize MDPs by allowing for incomplete information about the underlying state of the process, which is a natural situation in many applications. Each POMDP action has two effects: changing the world and gathering information about the world. In recent years, researchers have sought to combine the benefits of logic with the power of POMDPs. However, there has been very little research on relational POMDPs, and work has mostly focused either on efficient algorithms for propositional or grounded representations (Boutilier and Poole 1996; Hansen and Feng 2000; Geffner and Bonet 1998; Shani et al. 2008), or on representation (and not algorithmic) issues in the relational case (Poole 1997; Bacchus, Halpern, and Levesque 1999). The first approach to handle both aspects (Wang and Schmolze 2005) developed a simple model for relational POMDPs and a compact representation for grounded belief states, and used heuristic search with an algorithm for updating the belief state.

To our knowledge this paper is the first to give a complete treatment for relational POMDPs capturing non-ground optimal value functions and dynamic programming algorithms for them. We use the ground observation model (Wang and Schmolze 2005) that provides the simplest possible context for such a study. Our first contribution is a simple but important fact about relational POMDPs. We show that the value function of relational POMDPs must distinguish between quantification over states (that may specify conditions under which some action may be desirable) from quantification over belief states (that specifies the choice of action arguments). This implies that solutions based on value functions are inherently limited to the finite horizon case. On the other hand we show that the finite horizon case can be handled at the abstract level. Concretely we show that the incremental pruning algorithm (Cassandra, Littman, and Zhang 1997; Hansen and Feng 2000) calculating value iteration can be lifted to handle relational POMDPs. This approach provides abstract solutions that are independent of domain size and instantiation. Finally we show that the lifted algorithm can be realized using first order decision diagrams (FODD), a representation developed recently for RMDPs (Wang, Joshi, and Khardon 2008). This is particularly encouraging because the FODD framework for MDPs has been extended to more expressive settings (Joshi, Kersting, and Khardon 2009) and has been implemented and demonstrated to work well on stochastic planning problems (Joshi and Khardon 2008; Joshi, Kersting, and Khardon 2010). Very recently Sanner and Kersting (2010) explore extensions of our work to handle more complex observation functions.

Fully and Partially Observable MDPs

MDPs provide a mathematical model for sequential decision making. A MDP can be characterized by a state space S, an action space A, a state transition function $Pr(s_j|s_i, a)$ denoting the probability of transition to state s_j given state s_i and action a, and an immediate reward function $r(s)$, specifying the immediate utility of being in state s. A solution to a MDP is an optimal policy that maximizes expected discounted total reward as defined by the Bellman equation. The value iteration algorithm (VI) uses the Bellman equation to iteratively refine an estimate of the value function:

$$V_{n+1}(s) = max_{a \in A}[r(s) + \gamma \sum_{s' \in S} Pr(s'|s, a)V_n(s')] \quad (1)$$

where $V_n(s)$ represents our current estimate of the value function and $V_{n+1}(s)$ is the next estimate.

In POMDPs the states of the underlying MDP are not directly observable. We add an observation space O and an observation function $Pr(o|s, a)$, denoting the probability of observing o when action a is executed and the *resulting* state is s. Since the agent does not know the state, a belief state — a probability distribution over all states — is commonly used. It is well known that a POMDP can be converted to a MDP over belief space. Since the state space for this MDP is a $|S|$-dimensional continuous space, it is much more complex than MDPs with discrete state space. However, the value function is piecewise linear and convex and can be represented using a set of state-value functions $\{v^1, \cdots, v^m\}$ for some m so that for belief state b, Value$(b) = max_i \sum_s b(s)v^i(s)$ and this fact is useful in designing efficient algorithms. A significant amount of work has been devoted to finding efficient algorithms for POMDPs, including algorithms for propositionally factored POMDPs, e.g., (Hauskrecht 1997; Cassandra, Littman, and Zhang 1997; Hansen 1998; Hansen and Feng 2000; Shani et al. 2008).

FODDs and Value Iteration for RMDPs

In this section we briefly review the first order decision diagram representation and its use in solving RMDPs (Wang, Joshi, and Khardon 2008). This work follows up on the successful application of Algebraic Decision Diagrams (ADD) (Bahar et al. 1993) in solving propositionally factored MDPs (Hoey et al. 1999).

A First Order Decision Diagram (FODD) is a labeled directed acyclic graph, where each non-leaf node has exactly two children. The outgoing edges are marked with values `true` and `false`. Each non-leaf node is labeled with an atom $P(t_1, \ldots, t_n)$ or an equality $t_1 = t_2$ where t_i is a variable or a constant. Leaves are labeled with *non-negative* numerical values. An example FODD is shown in Figure 1(a).

Similar to first order logical formulas, the semantics of FODDs is given relative to interpretations. Informally, an interpretation corresponds to a state in a relational planning problem. An interpretation has a domain of elements, a mapping of constants to domain elements, and for each predicate a relation over the domain elements which specifies when the predicate is true. The semantics for FODDs is defined first relative to a variable valuation ζ. Given a FODD B over variables $\vec{x}$ and an interpretation I, a valuation ζ maps each variable in $\vec{x}$ to a domain element in I. Once this is done, each node predicate evaluates either to `true` or `false` and we can traverse a single path to a leaf. The value of this leaf is denoted by MAP$_B(I, \zeta)$. The value of the diagram on the interpretation is defined by aggregating over all reachable leaves, and here we use maximum aggregation, that is MAP$_B(I) = max_\zeta\{$MAP$_B(I, \zeta)\}$. Consider evaluating the diagram R in Figure 1(a) on interpretation I with domain $\{1, 2\}$ and relations $\{p_1(1), p_1(2), p_2(2)\}$. The valuation $\{x/1\}$ leads to a leaf with value 0; the valuation $\{x/2\}$ leads to a leaf with value 10. The final result is MAP$_R(I) = \max\{0, 10\} = 10$. Thus a FODD captures a real-valued function over interpretations. The maximum aggregation corresponds to existential quantification when leaf values are in $\{0, 1\}$, and gives useful maximization for value functions in the general case.

Given two functions f_1, f_2 captured by FODDs we can add the functions (denoted by $f_1 \oplus f_2$), multiply them ($f_1 \otimes f_2$), take the maximum between their values ($\max(f_1, f_2)$), or use any other binary operation. Taking addition as an example, this means that if $f = f_1 \oplus f_2$ then for any interpretation I, the value of f on I is the sum of the values returned by f_1 and f_2 when evaluated on I. A representation for f can be computed using a dynamic programming algorithm similar to the one for ADDs. Like ADDs, FODDs use a total order over node labels in order to simplify the computations. On the other hand simplification of FODDs is more complex; in previous work we developed a set of reduction operators to minimize representation size. The details of these FODD operations are not important for the constructions in the paper and are omitted due to space constraints.

RMDPs are MDPs where the states, transitions, and reward functions can be captured compactly by referring to objects and relations among them. Following Boutilier, Reiter, and Price (2001) RMDPs are specified by describing stochastic actions as a randomized choice among a small number of deterministic alternatives. In the FODD formulation of RMDPs, the domain dynamics of deterministic action alternatives are defined by *truth value diagrams* (TVDs). For every action schema $A(\vec{a})$ and each predicate schema $p(\vec{x})$ the TVD $T(A(\vec{a}), p(\vec{x}))$ is a FODD with $\{0, 1\}$ leaves. The TVD gives the truth value of $p(\vec{x})$ in the next state when $A(\vec{a})$ has been performed in the current state. We call $\vec{a}$ action parameters, and $\vec{x}$ predicate parameters. No other variables are allowed in the TVD. Probabilities, rewards, and value functions can be represented directly using FODDs, with the restriction that probability FODDs can only refer to constants or action parameters.

We use the following simple domain to illustrate the representation and later extend it to show how relational POMDP domains are formalized. The domain includes two predicates $p_1(x)$ and $p_2(x)$, and two actions A_1 and A_2. $A_1(x)$ deterministically makes $p_1(x)$ true. When attempting $A_2(x)$, a successful version $A_2S(x)$ is executed with probability 0.7 and it makes $p_2(x)$ true, and an unsuccessful version A_2F (effectively a *no-op*) is executed with probability

0.3. The reward function, capturing a planning goal, awards a reward of 10 if the formula $\exists x, p_1(x) \wedge p_2(x)$ is true. We assume the discount factor is 0.9 and that there is an absorbing state $\exists x, p_1(x) \wedge p_2(x)$, i.e., no extra value will be gained once the goal is satisfied. Figure 1(a) gives the reward function for this domain and TVDs are given in Figure 1(b)(c). All the TVDs omitted in the figure are trivial in the sense that the predicate is not affected by the action. The probabilistic choice of actions in this example can be captured with a FODD with a single numerical leaf; the framework allows the choice of actions to be state dependent.

The general first order value iteration algorithm for RMDPs works as follows (Boutilier, Reiter, and Price 2001). Given the reward function R and action model as input, set $V_0 = R, n = 0$ and repeat Procedure 1 until termination:

Procedure 1 (Performing one Iteration of Relational Value Iteration for MDPs)
1. For each action type $A(\vec{x})$, compute:
$Q^{A(\vec{x})}_{V_n} = R \oplus [\gamma \otimes \oplus_j (prob(A_j(\vec{x})) \otimes Regr(V_n, A_j(\vec{x})))]$
2. $Q^{A}_{V_n} = obj\text{-}max(Q^{A(\vec{x})}_{V_n})$.
3. $V_{n+1} = \max_A Q^{A}_{V_n}$.

In the first step we calculate the Q-function of a stochastic action $A(\vec{x})$ parameterized with action parameters $\vec{x}$. We can regress V_n through a deterministic action choice $A_j(\vec{x})$ by *block replacement*, i.e., replacing each node in V_n with the corresponding TVD, with outgoing edges connected to the 0,1 leaves of the TVD. Figure 1(d) illustrates regression by block replacement. A slightly more complex algorithm is needed in general to avoid violations of node ordering (Wang, Joshi, and Khardon 2008). To handle absorbing states in goal based domains where R has only one non-zero leaf (as in our example), we can replace step 1 with $Q^{A(\vec{x})}_{V_n} = max(R, \gamma \otimes \oplus_j (prob(A_j(\vec{x})) \otimes Regr(V_n, A_j(\vec{x})))$. Figure 1(e) and (f) show parameterized Q-functions $Q^{A_1(x^*)}_{V_0}$ and $Q^{A_2(x^*)}_{V_0}$ calculated by the first step.

In the second step we maximize over the action parameters of each Q-function to get the maximum value that can be achieved by using an instance of the action. To implement this the FODD based algorithm simply renames the action parameters using new variable names. This captures the correct meaning due to the maximum aggregation semantics of the FODDs. The third step maximizes over the Q-functions. This algorithm correctly calculates regression and hence VI for RMDPs (Wang, Joshi, and Khardon 2008).

Specifying Relational POMDPs

In POMDPs, observations replace knowledge of the world state. We follow standard POMDP convention where the observation probability $Pr(o|s, a)$ in the POMDP refers to the probability of observing o when action a is executed and the *resulting* state is s.

In general one could consider complex observations that might even include quantification. For example, in the example domain we may have a sensing action which gives information on $\exists x, p_2(x)$. In this paper, like (Wang and Schmolze 2005), we restrict ourselves to atomic observations and assume that each observation is associated with a subset of the action parameters, i.e., the agent can only have relational ground observations about the objects it operates on. We allow a modular representation of observations that can be combined. We use predicates to specify different aspects of an observation and a complete observation is the cross product of different aspects. If A has n observation aspects then $|\mathcal{O}_A|$, the number of possible observations associated with A, is 2^n. Note that if A provides no feedback, the null observation is obtained with certainty, and $|\mathcal{O}_A| = 1$,

For this paper we assume that different observation aspects are statistically independent. The general case can be handled in a similar way by capturing the conditional dependencies; we defer the details to the full version of the paper. An alternative approach (Poole 1997) associates each observation with something similar to a deterministic action alternative, and can be captured in our model as well. Each of the two approaches can be used to capture some situations more compactly than the other.

We next extend the example domain to illustrate the formalization. Suppose p_2 is not observable and we have a pure sensing action, $A_3(x)$, that does not change the state of the world but provides imperfect information about $p_2(x)$ through $q_2(x)$. If $p_2(x)$ is true, then $q_2(x)$ is true with probability 0.9. If $p_2(x)$ is false, then $q_2(x)$ is true with probability 0.2. Altogether this is a very simple domain. Each action has only one effect: it either changes the world or gathers information. Notice how the restriction to atomic observations is enforced. For object x^* if we execute $A_3(x^*)$, then the observation we get will be either $q_2(x^*)$ or $\neg q_2(x^*)$. In this domain A_1 and A_2 have no observations and A_3 has a single aspect thus $|\mathcal{O}_{A_1}|$=$|\mathcal{O}_{A_2}|$=1 and $|\mathcal{O}_{A_3}|$=2.

We use FODDs to specify observation probabilities $prob(o(\vec{y})|A(\vec{x}))$ where $\vec{y} \subseteq \vec{x}$. To guarantee correctness of the FODD procedure given below we require that the corresponding FODDs not contain variables; only action parameters and constants can appear as arguments in the observation probability FODD. Figure 1(h) gives the observation probability $prob(q_2(x^*)|A_3(x^*))$ where the condition $p_2()$ refers to the truth value of $p_2()$ after action execution. Due to independence, if we have more than one observation aspect, we can use $\otimes$ to multiply the FODD for each aspect and get the probability of a complete observation.

The Expected Value of a Belief State

Action selection requires that we evaluate the value function in the context of a belief state. When we have 0 steps to go, the expected value of a belief state given a reward function R can be computed using $\sum_s b(s)\text{MAP}_R(s) = \sum_s b(s) max_\zeta \text{MAP}_R(s, \zeta)$. However, as we show next given a Q-function, e.g. $Q^{A(\vec{x})}$ capturing value when we have one step to go, we cannot use the same equation. In relational domains two actions are the same iff they have the same action name *and* parameters. If we calculate as above then it is possible that the maximizing valuations for two different states do not agree on action parameters. If this happens then the calculated value wrongly assumes we can execute two different actions in the same belief state.

We illustrate this with an example. Suppose we have the following belief state in a world with objects $\{1,2\}$ and where we know that both $p_1(1)$ and $p_1(2)$ are false: $([\neg p_1(1)\wedge p_2(1)\wedge\neg p_1(2)\wedge p_2(2)] : 0.63; [\neg p_1(1)\wedge\neg p_2(1)\wedge \neg p_1(2)\wedge p_2(2)] : 0.27; [\neg p_1(1)\wedge p_2(1)\wedge\neg p_1(2)\wedge\neg p_2(2)] : 0.07; [\neg p_1(1)\wedge\neg p_2(1)\wedge\neg p_1(2)\wedge\neg p_2(2)] : 0.03)$. The Q-function we use is depicted in Figure 1(e) where x_1^* is the action parameter. For the first and the fourth states it does not matter what value x_1^* takes. Both $x_1^* = 1$ and $x_1^* = 2$ give the first state value of 9 and the fourth state value of 0. For the second and the third states, $x_1^* = 2$ and $x_1^* = 1$ give value 9 respectively. Therefore the expected value calculated using a state based formula is wrong because it is based on the best action for each state in a belief state.

To correctly capture the intended semantics of the value function, we define the expected value of a belief state given a Q-function as $Val(Q^{A(\vec{x})}, b) = max_{\zeta_{\vec{x}}} \sum_s b(s) max_{\zeta} \text{MAP}_Q(s, \zeta_{\vec{x}}\zeta)$ where $\zeta_{\vec{x}}$ is valuation to action parameters and ζ is valuation to all the other variables in the function. Note that for the belief state discussed above, the expected value given the Q-function in Figure 1(e) is 8.73 when $x_1^* = 2$.

As we show later in the value iteration algorithm, a set of parameterized value functions that make up a value function includes all the action parameters accumulated over the decision stages. Given a parameterized value function v^i as FODD, the expected value of a belief state is defined as

$$Val(v^i, b) = max_{\zeta_{\vec{x}_i}} \sum_s b(s) max_{\zeta_i} \text{MAP}_{v^i}(s, \zeta_{\vec{x}_i}\zeta_i) \quad (2)$$

where $\zeta_{\vec{x}_i}$ is valuation to all the action parameters in v^i and ζ_i is valuation to all the other variables in v^i. The expected value given a set of parameterized value functions $\{v^1, \cdots, v^n\}$ is defined as

$$Val(b) = max_{i\leq n}\{Val(v^i, b)\}. \quad (3)$$

Value Iteration for Relational POMDPs

This section presents a generalization of the incremental pruning algorithm (Cassandra, Littman, and Zhang 1997; Hansen and Feng 2000) for relational POMDPs. The algorithm works as follows: given the reward function R and the action model, we set $\mathcal{V}_0 = R$, n=0, and repeat Procedure 2 until termination. Steps 1,2,3,5 in the algorithm follow the structure in (Cassandra, Littman, and Zhang 1997; Hansen and Feng 2000) lifting the operations and adapting them to use appropriate operations with FODDs. Step 4 is new to the relational case and is required for correctness.

Procedure 2 (Performing one Iteration of Relational Value Iteration for POMDPs)

1. *For each action type $A(\vec{x})$, each observation $O_k^{A(\vec{x})}$ associated with $A(\vec{x})$, and each $v^i \in \mathcal{V}_n$ compute:*
   $\mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})},i} = \frac{R}{|\mathcal{O}^A|} \oplus \quad \gamma[\sum_j (prob(A_j(\vec{x})) \otimes$
   $Regr(prob(O_k^{A(\vec{x})}), A_j(\vec{x})) \otimes Regr(v^i, A_j(\vec{x})))]$.
2. $\mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})}} = \cup_i \mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})},i}$.
3. $\mathcal{Q}^{A(\vec{x})} = \bigoplus_k \mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})}}$.
4. $\mathcal{Q}^A$ *= rename action parameters $\vec{x}$ as special constants in $\mathcal{Q}^{A(\vec{x})}$.*
5. $\mathcal{V}_{n+1} = \cup_A \mathcal{Q}^A$.

The result in steps 2-5 of the algorithm is a set of Q or value functions. At each stage we can prune dominated members of this set. For example in step 2, if $\mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})},i}$ dominates $\mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})},j}$ for some $j \neq i$ then the latter can be removed. This type of pairwise dominance can be done with operations over FODDs. However, the test where one v^j is dominated by a set of other v^i's is more complex and requires further research. The model checking reductions of (Joshi, Kersting, and Khardon 2009) may provide a potential implementation of this step. Further details and discussion are given in (Wang 2007).

When we have an absorbing state (as in our example), the equation in the first step becomes:

$$\mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})},i} = \gamma[\sum_j (prob(A_j(\vec{x})) \otimes \quad (4)$$
$$Regr(prob(O_k^{A(\vec{x})}), A_j(\vec{x})) \otimes Regr(v^i, A_j(\vec{x})))].$$

The equation in the third step becomes

$$\mathcal{Q}^{A(\vec{x})} = max(R, \bigoplus_k \mathcal{Q}^{A(\vec{x}),O_k^{A(\vec{x})}}). \quad (5)$$

The first step: fixes an action and an observation, and associates the pair with some vector v^i in $\mathcal{V}_n$. The result captures the expected future value of executing action $A(\vec{x})$, and on observing $O_k^{A(\vec{x})}$ following the policy encoded in v^i. We have to regress over conditions of observation probabilities because they are specified in terms of the future state.

If the action has no observation, i.e., $|\mathcal{O}_A| = 1$, the first step can be rewritten as $\mathcal{V}^{A(\vec{x}),i} = R \oplus \gamma[\sum_j (prob(A_j(\vec{x})) \otimes Regr(v^i, A_j(\vec{x})))]$ which is the same as the calculation for RMDPs. Finally, note that in the first iteration, we do not need to take observations into account since they do not affect the result when no future actions are taken (since there are "0 steps to go").

Figure 1(e)(f)(g) give a set of FODDs $\{v^1, v^2, v^3\}$ as the result of the first iteration. We omit details of calculating these but give details of the second iteration since it is more informative. Figure 1(i)(j) show $\mathcal{Q}^{A_3(x^*),q_2(x^*),v^1}$ and $\mathcal{Q}^{A_3(x^*),\neg q_2(x^*),v^2}$ respectively, calculated by Equation 4. Figure 1(i) corresponds to the expected future value of executing action $A_3(x^*)$, and on observing $q_2(x^*)$ following the policy encoded in v^1. Figure 1(j) captures the case with observation $\neg q_2(x^*)$ and the policy encoded in v^2. Note that $A_3()$ is a pure sensing action and it does not change the world; therefore there is only one deterministic alternative, which is *no-op*. As a result the calculation is simplified since $prob(A_j) = 1$ and FODDs before and after regression over this action are the same.

The second step: groups the resulting FODDs in the first step by the action and the observation for all possible next step values v^i. To argue correctness of the algorithm, note

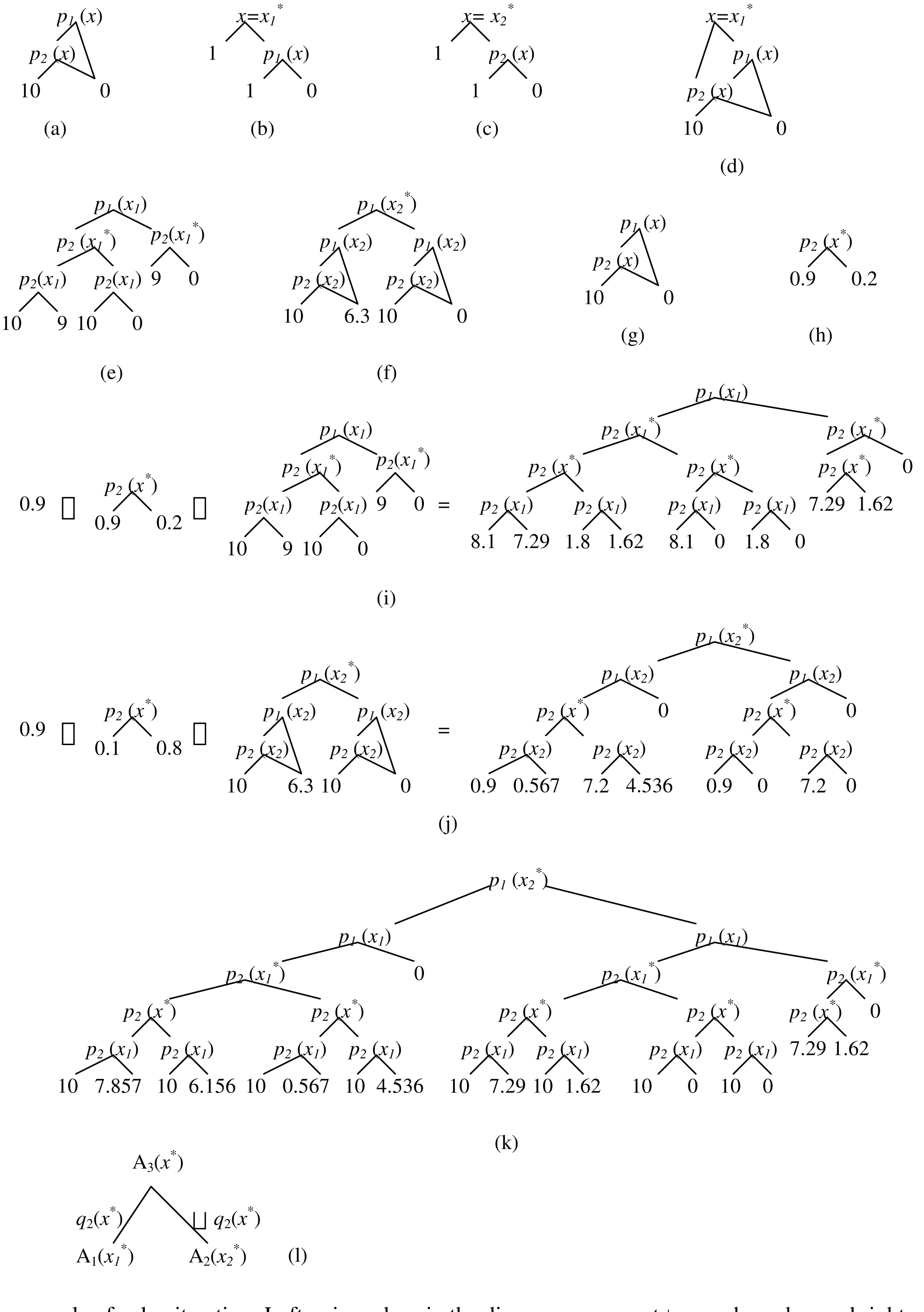

Figure 1: An example of value iteration. Left going edges in the diagrams represent `true` branches and right edges represent `false` branches. Variables decorated with a $*$ (as in x_1^*) represent action parameters; other variables (such as x_1) represent regular variables of the FODD. (a) The reward function R. (b) The TVD for $p_1(x)$ under action $A_1(x_1^*)$. (c) The TVD for $p_2(x)$ under action alternative $A_2S(x_2^*)$. (d) Calculating $Regr(R, A_1(x_1^*))$ by *block replacement*. (e) $Q_R^{A_1(x_1^*)}(i.e., v^1)$. (f) $Q_R^{A_2(x_2^*)}(i.e., v^2)$. (g) $Q_R^{A_3(x_3^*)}(i.e., v^3)$. (h) Observation probability $prob(q_2(x^*)|A_3(x^*))$. (i) $\mathcal{Q}^{A_3(x^*),q_2(x^*),v^1}$. (j) $\mathcal{Q}^{A_3(x^*),\neg q_2(x^*),v^2}$. (k) One parameterized value function in $\mathcal{Q}^{A_3(x^*)}$. (l) The parameterized policy tree corresponding to the value function in (k).

that if we use Equation 3 to maximize over the set of resulting FODDs we get the best value that can be obtained when taking $A(\vec{x})$ and observing $O_k^{A(\vec{x})}$.

The third step: sums together all the value contributions from different observations by calculating the cross sum $\bigoplus$ on sets of FODDs for the same parameterized action, where $A \bigoplus B = \{\alpha \oplus \beta | \alpha \in A, \beta \in B\}$ given two sets of FODDs A and B. Note that we use $\bigoplus$ to denote *cross sum*, while use $\oplus$ to denote the addition of two FODDs. Figure 1(k) gives the result of $max(R, \mathcal{Q}^{A_3(x^*), q_2(x^*), v^1} \oplus \mathcal{Q}^{A_3(x^*), \neg q_2(x^*), v^2})$, calculated as in Equation 5. This gives one of the functions for the action type $A_3(x^*)$. It also encodes a non-stationary 2-step parameterized policy tree as shown in Figure 1(l). To argue correctness, note that for any set of concrete action parameters $\vec{x}$, if we use Equation 3 to maximize over the set of resulting FODDs we get the best value that can be obtained by taking $A(\vec{x})$.

The fourth step: Recall that so far we have calculated Q relative to an action parameterized with arguments $\vec{x}$. The fourth step turns the action parameters into special constants for each FODD in the set. The corresponding step in value iteration for RMDPs performs object maximization at this stage to get the maximal value an instance of an action can achieve. However, as discussed in the previous section, we cannot use existential quantification here because the formulas will be evaluated in each state separately leading to wrong results. We therefore conclude:

Observation 1 *The value function calculated in value iteration in relational POMDPs must distinguish standard existentially quantified variables (quantified per state) from those which are quantified once over the belief state. This is a property of relational POMDPs and is true across different representation frameworks.*

Consider again our running example. In the first iteration when we reach the fourth step, we have Q-functions for $A_1(x_1^*)$, $A_2(x_2^*)$, and $A_3(x_3^*)$ as shown in Figure 1(e)(f)(g) and we denote them as $\{v^1, v^2, v^3\}$. Here x_1^* and x_2^* are action parameters for the corresponding Q-functions. Notice that x_3^* was dropped in reducing $Q_R^{A_3(x_3^*)}$. This is simply because $A_3()$ is a pure sensing action and will not have any effect when there is only one step to go. In the second iteration, the value function FODD for $A_3(x^*)$ shown in Figure 1(k) contains three action parameters: its own action parameter x^* and two action parameters, x_1^* and x_2^*, "inherited" from its future plan. The action parameters are treated as constants for the purpose of FODD reductions. In evaluation they are treated differently than other variables as prescribed in Equation 2.

The fifth step: puts together the Q-functions for all actions (each of which is made up of a set of value function FODDs) and the combined set forms the updated value function. As sketched in the discussion above we have:

Theorem 1 *If the input to Procedure 2, $\mathcal{V}_n$, is a collection of Q-functions correctly capturing the value of belief states when there are n steps to go, then the output $\mathcal{V}_{n+1}$ is a collection of Q-functions correctly capturing the value function when there are $n + 1$ steps to go.*

Acknowledgments

This work was partly supported by NSF grant IIS 0936687.

References

Bacchus, F.; Halpern, J. Y.; and Levesque, H. J. 1999. Reasoning about noisy sensors and effectors in the situation calculus. *Artificial Intelligence* 111:171–208.

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1993. Algebraic decision diagrams and their applications. In *Proceedings of the International Conference on Computer-Aided Design*, 188–191.

Boutilier, C., and Poole, D. 1996. Computing optimal policies for partially observable decision processes using compact representations. In *AAAI*, 1168–1175.

Boutilier, C.; Reiter, R.; and Price, B. 2001. Symbolic dynamic programming for first-order MDPs. In *IJCAI*, 690–700.

Cassandra, A.; Littman, M.; and Zhang, N. 1997. Incremental pruning: A simple, fast, exact method for partially observable Markov Decision Processes. In *UAI*, 54–61.

Fern, A.; Yoon, S.; and Givan, R. 2006. Approximate policy iteration with a policy language bias: Solving relational markov decision processes. *JAIR* 25:75–118.

Geffner, H., and Bonet, B. 1998. High-level planning and control with incomplete information using POMDPs. In *Proceedings of Fall AAAI Symposium on Cognitive Robotics*.

Hansen, E. A., and Feng, Z. 2000. Dynamic programming for POMDPs using a factored state representation. In *AIPS*, 130–139.

Hansen, E. A. 1998. Solving POMDPs by search in policy space. In *UAI*, 211–219.

Hauskrecht, M. 1997. A heuristic variable-grid solution method for POMDPs. In *AAAI*, 727–733.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic planning using decision diagrams. In *UAI*, 279–288.

Hölldobler, S., and Skvortsova, O. 2004. A logic-based approach to dynamic programming. In *AAAI-04 workshop on learning and planning in Markov Processes – advances and challenges*.

Joshi, S., and Khardon, R. 2008. Stochastic planning with first order decision diagrams. In *ICAPS*, 156–163.

Joshi, S.; Kersting, K.; and Khardon, R. 2009. Generalized first order decision diagrams for first order markov decision processes. In *IJCAI*, 1916–1921.

Joshi, S.; Kersting, K.; and Khardon, R. 2010. Self-taught decision theoretic planning with first order decision diagrams. In *ICAPS*.

Kersting, K.; Van Otterlo, M.; and De Raedt, L. 2004. Bellman goes relational. In *ICML*, 465–472.

Poole, D. 1997. The independent choice logic for modeling multiple agents under uncertainty. *Artificial Intelligence* 94:7–56.

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

Sanner, S., and Kersting, K. 2010. Symbolic dynamic programming for first-order POMDPs. In *AAAI*.

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

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

Wang, C.; Joshi, S.; and Khardon, R. 2008. First order decision diagrams for relational MDPs. *JAIR* 31:431–472.

Wang, C. 2007. *First order Markov Decision Processes*. Ph.D. Dissertation, Tufts University. Technical Report TR-2007-4.