First Order Decision Diagrams for Relational MDPs

Saket Joshi SJOSHI01@CS.TUFTS.EDU
Roni Khardon RONI@CS.TUFTS.EDU
Chenggang Wang CWAN@CS.TUFTS.EDU
Department of Computer Science, Tufts University, 161 College Avenue, Medford, MA 02155, USA

Abstract

Dynamic programming algorithms provide a basic tool identifying optimal solutions in Markov Decision Processes (MDP). The paper develops a representation for decision diagrams suitable for describing value functions, transition probabilities, and domain dynamics of First Order or Relational MDPs (FOMDP). By developing appropriate operations for such diagrams the paper shows how value iteration can be performed compactly for such problems. This improves on previous approaches since the representation combines compact form with efficient operations and manipulation. The work also raises interesting issues on suitability of different representations to different FOMDPs tasks.

1. Introduction

In the past years there has been an increased interest in developing relational or first-order MDPs. Some examples include symbolic dynamic programming (SDP) (Boutilier et al., 2001), the relational Bellman algorithm (ReBel) (Kersting et al., 2004), approximate linear programming for RMDPs and FOMDPs (Guestrin et al., 2003; Sanner & Boutilier, 2005), approximate policy iteration (Fern et al., 2003), and inductive policy selection using first-order regression (Gretton & Thiebaux, 2004).

Among these, only SDP and ReBel are exact solution methods. To our knowledge there is no working implementation of SDP because it is hard to keep the state formulas consistent and of manageable size in the context of situation calculus. Compared with SDP, ReBel provides a more practical solution. ReBel uses simpler language (a probabilistic STRIPS-like language) to represent RMDPs, so that reasoning over formulas is easier to perform.

Inspired by the successful application of Algebraic Decision Diagrams (ADD) (Bryant, 1986; McMillan, 1993; Bahar et al., 1993) in solving propositionally factored MDPs (Hoey et al., 1999; St-Aubin et al., 2000) we lift propositional ADDs to handle relational structure and use them in the solution of FOMDPs. The intuition behind this idea is that ADD representation allows information sharing, e.g., sharing between state partitions. If there is a sufficient regularity in the model, ADDs can be very compact, allowing problems to be represented and solved efficiently.

First order decision trees and even decision diagrams have already been considered in the literature (Blockeel & De Raedt, 1998; Groote & Tveretina, 2003) and several semantics for such diagrams are possible. In particular Groote and Tveretina (2003) provide a notation for first order BDDs that can capture formulas in Skolemized conjunctive normal form and then provide a theorem proving algorithm based on this representation. In this paper we adapt and extend their approach to handle first order MDPs. In particular, we extend the definitions to handle existential quantification and numerical leaves through the use of an aggregation function. This allows us to capture value functions using algebraic diagrams in a natural way. We also provide additional reduction transformations for algebraic diagrams that help keep their size small, and allow the use of background knowledge in reductions. We then develop appropriate representation and algorithms showing how value iteration can be performed using the decision diagrams.

It is useful to compare our solutions to the propositional ones. The main difficulty in lifting the ideas from the propositional case (Hoey et al., 1999; St-Aubin et al., 2000) is that in relational domains the transition function specifies a set of schemas for conditional probabilities. The propositional solution uses

Presented at the ICML Workshop on Open Problems in Statistical Relational Learning, Pittsburgh, PA, 2006.

the concrete conditional probability to calculate the regression function. But this is not possible with schemas. One way around this problem is to first ground the domain and problem at hand and only then perform the reasoning (see for example (Sanghai et al., 2005)). However this does not allow for solutions abstracting over domains and problems. Like SDP and ReBel our constructions do perform general reasoning and they do so by using decision diagrams.

2. Markov decision processes

We assume familiarity with standard notions of MDPs and value iteration (see for example (Bellman, 1957; Puterman, 1994)). In the following we introduce some of the notions and our notation.

Markov Decision Processes (MDPs) are mathematical models of sequential optimization problems with stochastic actions. 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 an MDP is an optimal policy that maximizes expected discounted total reward as defined by the Bellman equation. The value iteration algorithm uses the Bellman equation to iteratively refine an estimate of the value function:

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

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

The main observation used by Hoey et al. (1999) is that if we can represent each of $r(s)$, $Pr(s'|s, a)$, and $V_k(s)$ compactly using an algebraic decision diagram then value iteration can be done directly using these representations, avoiding the need to enumerate the state space which is implicit in the equation above.

Taking the next step, the SDP approach (Boutilier et al., 2001) was developed in the context of the situation calculus. One of the useful restrictions introduced in this work is that stochastic actions must be specified as a non-deterministic choice among deterministic alternatives. In this way one can separate the regression over action effects, which is now deterministic, from the probabilistic choice of action. On each regression step during value iteration, the value of a stochastic action $A(\vec{x})$ parameterized with free variables $\vec{x}$ is determined in the following manner:

$$Q^V(A(\vec{x}), s) = rCase(s) \oplus \gamma[\oplus_j pCase(n_j(\vec{x}), s) \otimes regr(vCase(do(n_j(\vec{x}), s)))]$$

where $rCase(s)$ and $vCase(s)$ denote reward and value functions in the compact "case notation" of Boutilier et al. (2001), $n_j(\vec{x})$ denotes the possible outcomes of the action $A(\vec{x})$, and $pCase(n_j(\vec{x}), s)$ the choice probabilities for $n_j(\vec{x})$.

After the regression, we need to maximize over the action parameters of each Q-function to get the maximum value that could be achieved by using an instance of this action. In SDP, this is done by adding the negation of higher value partitions into the description of lower value partitions, leading to complex formulas and reasoning. Finally, to get the next value function we maximize over the choice of action schema.

The solution of ReBel (Kersting et al., 2004) follows the same outline but uses a simpler logical language, a probabilistic STRIPS-like language, for representing FOMDPs. More importantly the paper uses a decision list (Rivest, 1987) style representation for value functions and policies. The decision list gives us an implicit maximization operator since rules higher on the list are evaluated first. As a result the object maximization step is very simple in ReBel. Each state partition is represented implicitly by the negation of all rules above it, and explicitly by the conjunction in the rule. On the other hand regression in ReBel requires that one enumerate all possible matches between a subset of a conjunctive goal (or state partition) and action effects and reason about each of these separately.

3. First-order Decision Diagrams

An Algebraic Decision Diagram is a labeled directed acyclic graph where non-leaf nodes are labeled with propositional variables, each non-leaf node has exactly two children corresponding to `true` and `false` branches, and leaves are labeled with numerical values. Ordered decision diagrams specify a fixed order on propositions and require that node labels respect this order on every path in the diagram. In this case every function has a unique canonical representation and diagrams have efficient manipulation algorithms, leading to successful applications (Bryant, 1986; McMillan, 1993; Bahar et al., 1993).

There are various ways to generalize ADDs to capture relational structure. One could use closed or open formulas in the nodes, and in the latter case we must interpret the quantification over the variables. We focus on the following syntactic definition which does not have any explicit quantifiers.

Definition of First Order Decision Diagrams
(1) We assume a signature with a fixed set of predicates and constant symbols, and an enumerable set of

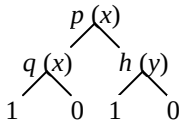

Figure 1. A simple FODD.

variables. We also allow to use an equality between any pair of terms (constants or variables).
(2) 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`.
(3) Each non-leaf node is labeled with: an atom $P(t_1, \ldots, t_n)$ or an equality $t_1 = t_2$ where each t_i is a variable or a constant.
(4) Leaves are labeled with numerical values.

Figure 1 shows a FODD with binary leaves. Left going edges represent `true` branches. To simplify diagrams in the paper we draw multiple copies of the leaves 0 and 1 but they represent the same node in the FODD.

The semantics of first order formulas are given relative to interpretations. 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. There is more than one way to define the meaning of FODD B and on interpretation I. In the following we discuss two possibilities.

Semantics based on a single path: A semantics for decision trees is given by Blockeel and De Raedt (1998) that can be adapted to FODDs. The semantics define a unique path that is followed when traversing B relative to I. All variables are existential and a node is evaluated relative to the path leading to it. For example, if we evaluate the diagram in Figure 1 on the interpretation with domain $\{1, 2, 3\}$ and relations $\{p(1), q(2), h(3)\}$ then we follow the `true` branch at the root since $\exists x, p(x)$ is satisfied, but we follow the `false` branch at $q(x)$ since $\exists x, p(x) \wedge q(x)$ is not satisfied. Since the leaf is labeled with 0 we say that B does not satisfies I. This is an attractive approach, since it builds mutually exclusive partitions over states, and various FODD operations can be developed for it. However, for reasons we discuss later this semantics is not so well suited to value iteration, and it is therefore not used in the paper.

Semantics based on a multiple paths: Following Groote and Tveretina (2003) we define the semantics 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 $\text{MAP}_B(I, \zeta)$.

We next define $\text{MAP}_B(I) = \text{aggregate}_\zeta\{\text{MAP}_B(I, \zeta)\}$ for some aggregation function. That is, we consider all possible valuations ζ, for each we calculate $\text{MAP}_B(I, \zeta)$ and then we aggregate over all these values. In (Groote & Tveretina, 2003) leaf labels are in $\{0, 1\}$ and variables are universally quantified; this is easily captured by using minimum as the aggregation function. In this paper we use maximum as the aggregation function. This corresponds to existential quantification in the binary case, and gives useful maximization for value functions in the general case. We therefore define:

$$\text{MAP}_B(I) = \max_\zeta\{\text{MAP}_B(I, \zeta)\}$$

Consider evaluating the diagram in Figure 1 on the interpretation with domain $\{1, 2, 3\}$ and relations $\{p(1), q(2), h(3)\}$. The valuation $\{x/2, y/3\}$ leads to a leaf with value 1 so the maximum is 1 and we say that I satisfies B.

We define node formulas (NF) and edge formulas (EF) recursively as follows. For a node n labeled $l(n)$ with incoming edges $e_1, \ldots, e_k$, the node formula $\text{NF}(n) = (\vee_i \text{EF}(e_i))$. Denote the `true` branch of a node n by $n_{\downarrow t}$ and the `false` branch by $n_{\downarrow f}$. The edge formula for the `true` outgoing edge of n is $\text{EF}(n_{\downarrow t}) = \text{NF}(n) \wedge l(n)$. The edge formula for the `false` outgoing edge of n is $\text{EF}(n_{\downarrow f}) = \text{NF}(n) \wedge \neg l(n)$. These formulas, where all variables are existentially quantified, capture the conditions under which a node or edge are reached.

Basic Reduction of FODDs: Groote and Tveretina (2003) define several operators that reduce a digram into "normal form". A total order over open predicates (node labels) is assumed. We describe these operators briefly and give their main properties.

(R1) Neglect operator: if both children of a node p in the FODD lead to the same node q then we remove p and link all parents of p to q directly. **(R2)** Join operator: if two nodes p, q have the same label and point to the same 2 children then we can join p and q (remove q and link q's parents to p). **(R3)** Merge operator: if a node and its child have the same label then the parent can point directly to the grandchild. **(R4)** Sort operator: If a node p is a parent of q but the label ordering is violated ($l(p) > l(q)$) then we can reorder the nodes locally using 2 copies of p and q such that labels of the nodes do not violate the ordering.

Define a FODD to be reduced if none of the 4 operators

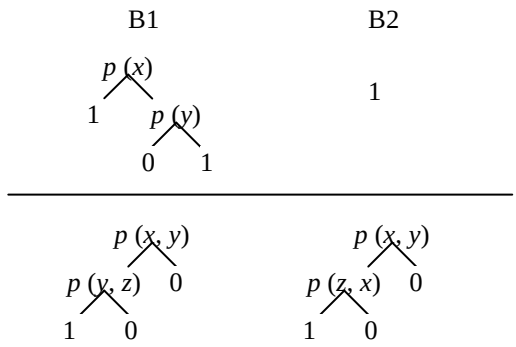

Figure 2. Examples illustrating weakness of normal form.

can be applied. We have the following:

Theorem 1 (Groote & Tveretina, 2003) *(1) Let $O \in \{Neglect, Join, Merge, Sort\}$ be an operator and $O(B)$ the result of applying O to FODD B, then for any ζ, $MAP_B(I,\zeta) = MAP_{O(B)}(I,\zeta)$*
(2) if B_1, B_2 are reduced and satisfy $\forall\zeta$, $MAP_{B_1}(I,\zeta) = MAP_{B_2}(I,\zeta)$ then they are identical.

Property (1) gives soundness, and property (2) shows that reducing a FODD gives a normal form. However, this only holds if the maps are identical for every ζ and this condition is stronger than normal equivalence. Figure 2 shows two pairs of reduced FODDs such that $\text{MAP}_{B_1}(I) = \text{MAP}_{B_2}(I)$ but $\exists\zeta, \text{MAP}_{B_1}(I,\zeta) \neq \text{MAP}_{B_2}(I,\zeta)$. In this case although the maps are the same the FODDs are not reduced to the same form. This weak normal form suffices for Groote and Tveretina (2003) who use it to provide a theorem prover for first order logic.

Combining FODDs: Given two algebraic diagrams we may need to add the corresponding functions, take the maximum or use any other binary operation `op` over the values represented by the functions. Here we adopt the solution from the propositional case (Bryant, 1986) in the form of the procedure **Apply(**p,q,`op`**)** where p and q are the roots of two diagrams. This procedure chooses a new root label (the lower among labels of p, q) and recursively combines the corresponding sub-diagrams, according to the relation between the two labels ($<$, $=$, or $>$). In order to make sure the result is reduced in the propositional sense one can use dynamic programming to avoid generating nodes for which either neglect or join operators ((R1) and (R2) above) would be applicable.

Additional Reduction Operators: In our context, especially for algebraic FODDs we may want to reduce the diagrams further. We distinguish *strong reduction* that preserves $\text{MAP}_B(I,\zeta)$ for all ζ and *weak reduction* that only preserves $\text{MAP}_B(I)$. In the following let $\mathcal{B}$ represent any background knowledge we have about the domain. For example in the Blocks World we may know that $\forall x, y, [on(x,y) \rightarrow \neg clear(y)]$.

(R5) Strong Reduction for Implied Branches: Consider any node n with label $l(n)$. Let $\vec{x}$ be the variables in $\text{EF}(n_{\downarrow t})$. If $\mathcal{B} \models \forall\vec{x}, [\text{NF}(n) \rightarrow l(n)]$ then whenever node n is reached then the `true` branch is followed. In this case we can remove n and connect its parent directly to the `true` branch. It is clear that the map is preserved for any valuation. A similar reduction can be formulated for the `false` branch.

Implied branches may be a result of background knowledge or simply a result of equalities along a path. For example $(x = y) \wedge p(x) \rightarrow p(y)$ so we may prune $p(y)$ if $(x = y)$ and $p(x)$ are known to be true.

(R6) Weak Reduction Removing Dominated Branches: Consider any node n such that if we can reach node n using some valuation then we can reach $n_{\downarrow t}$ using a possibly different valuation. If $n_{\downarrow t}$ always gives better values than $n_{\downarrow f}$ then we should be able to remove $n_{\downarrow f}$ from the diagram. We first present a simple version of this condition and then follow with a more general case.

Let $\vec{x}$ be the variables that appear in $\text{NF}(n)$, and $\vec{y}$ the variables in $l(n)$ and not in $\text{NF}(n)$. Consider the condition **(I1):** $\mathcal{B} \models \forall\vec{x}, [\text{NF}(n) \rightarrow \exists\vec{y}, l(n)]$ which requires that every valuation reaching n can be extended into a valuation reaching $n_{\downarrow t}$.

Let $\min(n_{\downarrow t})$ be the minimum leaf value in $n_{\downarrow t}$, and $\max(n_{\downarrow f})$ be the maximum leaf value in $n_{\downarrow f}$. Consider next the additional condition **(V1):** $\min(n_{\downarrow t}) \geq \max(n_{\downarrow f})$. In this case regardless of the valuation we know that it is better to follow $n_{\downarrow t}$ and not $n_{\downarrow f}$. If both I1 and V1 hold then according to maximum aggregation the value of $\text{MAP}_B(I)$ will never be determined by the `false` branch. Therefore we can safely replace $n_{\downarrow f}$ with any constant value between 0 and $\min(n_{\downarrow t})$ without changing the map.

In some cases we can also drop the node n completely and connect its parents directly to $n_{\downarrow t}$. This can be done if **(I1A):** for every valuation ζ_1 that reaches $n_{\downarrow f}$ there is a valuation ζ_2 that reaches $n_{\downarrow t}$ and such that ζ_1 and ζ_2 agree on all variables in (the sub-diagram of) $n_{\downarrow t}$. To see that this is true consider any valuation ζ_1 for the FODD. If ζ_1 leads to $n_{\downarrow t}$ it will continue reaching $n_{\downarrow t}$ and the value will not change. If ζ_1 leads to $n_{\downarrow f}$ it will now lead to $n_{\downarrow t}$ reaching some leaf of the diagram (and giving its value instead of the constant value assigned above). By the condition, ζ_2 will reach the same leaf and assign the same value. So under maximum aggregation the map is not changed. It

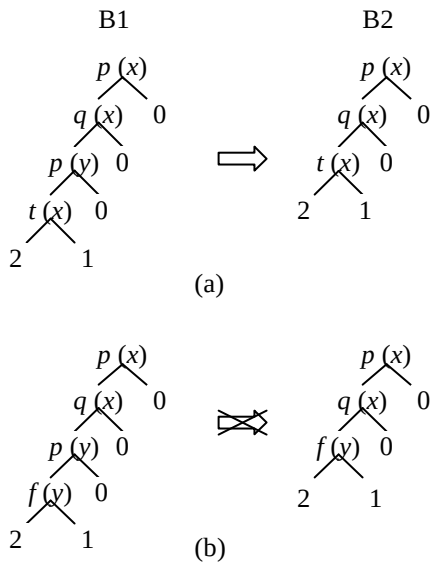

Figure 3. Examples illustrating Remove Reduction R6.

is easy to see that I1A follows from I1 if **(I1B):** no variable in $\vec{y}$ appears in the sub-diagram of $n_{\downarrow t}$. More generally we can state the following condition. Let $\vec{u}$ be the variables that appear in $n_{\downarrow t}$, $\vec{v}$ be the variables that appear in $\mathrm{NF}(n)$ but not in $n_{\downarrow t}$, and $\vec{w}$ the variables in $l(n)$ and not in $\vec{u}$ or $\vec{w}$. The condition I1A is satisfied if **(I1C):** $\mathcal{B} \models \forall \vec{u}, [[\exists \vec{v}, \mathrm{NF}(n)] \rightarrow [\exists \vec{v}, \vec{w}, l(n)]$.

Figure 3 illustrates two cases in R6. In part (a) conditions I1, I1B and V1 hold and we can drop $p(y)$ completely. We cannot drop $p(y)$ in part (b). Intuitively removing $p(y)$ removes a constraint on y. Consider an interpretation I with domain $\{1, 2\}$ and relations $\{p(1), q(1), f(2)\}$. Before reduction, $\mathrm{MAP}_{B1}(I) = 1$. But if $p(y)$ is removed $\mathrm{MAP}_{B2}(I) = 2$.

A symmetric operation can be applied replacing $n_{\downarrow t}$ and $n_{\downarrow f}$. This resolves the first example from Figure 2 but not the second one which has to do with sorting.

Some care is needed when applying weak reductions, e.g. when replacing a sub-diagram with 0 as above. While this preserves the map it does not preserve the map for every valuation. If we apply non-monotonic operations that depend on all valuations (e.g. subtract two diagrams that share variables) the result may be incorrect. The reductions are always safe for monotonic operations and when we only operate in this way if different diagrams do not share variables.

An important special case of R6 occurs when $l(n)$ is an equality $t_1 = y$ where y is a variable that does not occur on the FODD above node n. In this case, the condition I1 holds since we can choose the value of y. Therefore if V1 holds we can remove the node n connecting its parents to $n_{\downarrow t}$ and substituting t_1 for y in the diagram $n_{\downarrow t}$. As we see below this case is typical when using FODDs for MDPs.

(R6) general case: We first show that the condition on values can be relaxed. Consider the diagram $D = n_{\downarrow t} - n_{\downarrow f}$ which we can calculate using Apply. If **(V2):** all leaves in D have non-negative values then it is clear that for any fixed valuation it is better to follow $n_{\downarrow t}$ instead of $n_{\downarrow f}$. Consider again condition I1A from above requiring that for every valuation ζ_1 there is a useful ζ_2. Exactly the same argument as above shows that in this case we can drop the node n and connect its parents to $n_{\downarrow t}$.

Finally, we argue that the condition for reachability of $n_{\downarrow t}$ can also be relaxed. In particular **(I2):** $\mathcal{B} \models [\exists \vec{x}, \mathrm{NF}(n)] \rightarrow [\exists \vec{x}, \vec{y}, \mathrm{EF}(n_{\downarrow t})]$ requires that if n is reachable then $n_{\downarrow t}$ is reachable but does not put any restriction on the valuations (in contrast with the requirement in I1 to extend the valuation reaching n). If both I2 and V1 hold then as above we can replace $n_{\downarrow f}$ with a constant between 0 and $\min(n_{\downarrow t})$ without changing the map. In order to drop the node n completely we need to guarantee that I1A holds. Using the notation for $\vec{u}, \vec{v}, \vec{w}$ from above this is satisfied if **(I2A):** $\mathcal{B} \models \forall \vec{u}, [\exists \vec{v}, \mathrm{NF}(n)] \rightarrow [\exists \vec{v}, \vec{w}, \mathrm{EF}(n_{\downarrow t})]$.

To summarize we can replace $n_{\downarrow f}$ with a constant if both I1 and V1 hold or both I2 and V1 hold. Recall that (I1∧I1B)∨I1C→I1A. We can drop n completely if both I1A and V1 hold or both I1A and V2 hold or both I2A and V2 hold.

(R7) Weak Reduction Removing Dominated Nodes: Consider a FODD with two nodes p, q where q is in the sub-FODD of $p_{\downarrow f}$ and their formulas satisfy that if we can follow $q_{\downarrow t}$ then we can also follow $p_{\downarrow t}$. Formally, let $\vec{x}$ be the variables in $\mathrm{EF}(p_{\downarrow t})$ and $\vec{y}$ the variables in $\mathrm{EF}(q_{\downarrow t})$. We require that **(I3):** $\mathcal{B} \models [\exists \vec{x}, \mathrm{EF}(q_{\downarrow t})] \rightarrow [\exists \vec{y}, \mathrm{EF}(p_{\downarrow t})]$.

If I3 holds and if **(V3):** $\min(p_{\downarrow t}) \geq \max(q_{\downarrow t})$ then $\mathrm{MAP}_B(I)$ will never be determined by $q_{\downarrow t}$ so we can replace $q_{\downarrow t}$ with a constant between 0 and $\min(p_{\downarrow t})$. This is true since if a valuation reaches $q_{\downarrow t}$ then there is another valuation reaching $p_{\downarrow t}$ and by V3 it gives a higher value so that the map is not changed.

If in addition to I3 and V3 we have **(V4):** $\min(p_{\downarrow t}) \geq \max(q_{\downarrow f})$ then it is also safe to remove q completely. To see this consider any valuation reaching $q_{\downarrow t}$. As above its true value is dominated by another valuation reaching $p_{\downarrow t}$. When we remove q the valuation will reach $q_{\downarrow f}$ and by V4 the value produced is smaller than the value from $p_{\downarrow t}$. So again the map is preserved.

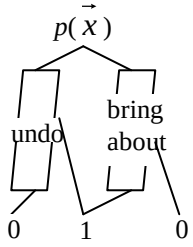

Figure 4. A template for the TVD

Here too we can relax the condition for applying the reduction. Let T be a diagram with q as root but with $q_{\downarrow f}$ replaced with 0. Consider the diagram $D = p_{\downarrow t} - T$ which we can calculate using Apply. If **(V5):** all leaves in D have non-negative values, then it is clear that for any fixed valuation it is better to follow $p_{\downarrow t}$ instead of going through q to $q_{\downarrow t}$.

Now consider the condition **(I3A):** for every valuation ζ_1 that reaches $q_{\downarrow t}$ there is a valuation ζ_2 that reaches $p_{\downarrow t}$ and such that ζ_1 and ζ_2 agree on all variables in $p_{\downarrow t}$, q and $q_{\downarrow t}$. Let $\vec{u}$ be the variables that appear in $p_{\downarrow t}$ or $q_{\downarrow t}$ or in $l(q)$, $\vec{v}$ be the variables that appear in $\mathrm{EF}(p_{\downarrow t})$ but are not in $\vec{u}$, and let $\vec{w}$ be the variables that appear in $\mathrm{EF}(q_{\downarrow t})$ but are not in $\vec{u}$. Condition I3A holds if **(I3B):** $\mathcal{B} \models \forall \vec{u}, [[\exists \vec{w}, \mathrm{EF}(q_{\downarrow t})] \rightarrow [\exists \vec{v}, \mathrm{EF}(p_{\downarrow t})]]$.

If I3B and V5 hold then we can replace $q_{\downarrow t}$ with a constant as above. To see that this is true consider any valuation ζ_1 that reaches $q_{\downarrow t}$. By I3A ζ_2 reaches $p_{\downarrow t}$ and by V5 it achieves a higher value. Note that here we need the fact that ζ_1, ζ_2 agree on all variables in $\vec{u}$ so the value from D is correct. As above if we also have V4 then we can drop q completely.

To summarize if both I3 and V3 hold or both I3B and V5 hold then we can replace $q_{\downarrow t}$ with a constant. If V4 holds as well then we can drop q completely.

(R8) Weak Reduction by Unification: Consider a FODD B and two sets of variables $\vec{x}$ and $\vec{y}$ of the same cardinality. By $B\{\vec{x}/\vec{y}\}$ we denote the FODD resulting from replacing variables in $\vec{x}$ by the corresponding variables in $\vec{y}$. Now consider the FODD $B\{\vec{x}/\vec{y}\} - B$ which we can calculate using Apply. If all leaves in this diagram are non negative then we can safely replace B by $B\{\vec{x}/\vec{y}\}$.

4. Decision Diagrams for MDPs

We follow Boutilier et al. (2001) and specify stochastic actions as a non-deterministic choice among deterministic alternatives. We therefore need to use FODDs to represent the deterministic domain dynamics of actions, the probabilistic choice among actions, and value functions.

Example Domain: We use the following variant of the logistics problem (Boutilier et al., 2001) to illustrate our constructions for MDPs. The domain includes boxes, trucks and cities, and predicates are $Bin(Box, City)$, $Tin(Truck, City)$, and $On(Box, Truck)$ with their obvious meaning. The reward function, capturing a planning goal, awards a reward of 10 if the formula $\exists b, Bin(b, Paris)$ is true, that is if there is any box in Paris. Thus the reward is allowed to include constants but need not be completely ground.

The domain includes 3 actions *load*, *unload*, and *drive*. Actions have no effect if their preconditions are not met. Actions can also fail with some probability. When attempting *load*, a successful version *loadS* is executed with probability 0.99, and an unsuccessful version *loadF* (effectively a no-operation) with probability 0.01. The drive action is executed deterministically. When attempting *unload*, the probabilities depend on whether it is raining or not. If it is not raining then a successful version *unloadS* is executed with probability 0.9, and *unloadF* with probability 0.1. If it is raining *unloadS* is executed with probability 0.7, and *unloadF* with probability 0.3.

The domain dynamics: 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. The truth value is valid when we fix a valuation of the parameters.

Notice that the TVD simultaneously captures the truth values of all instances of $p(\vec{x})$ in the next state. Notice also that TVDs for different predicates are separate and independent. This can be safely done even if an action has coordinated effects (not conditionally independent) since the actions are deterministic.

For any domain, a TVD for predicate $p(\vec{x})$ can be defined generically as in Figure 4. The idea is that the predicate is true if it was true before and is not "undone" by the action or was false before and is "brought about" by the action. TVDs for the logistics domain in our running example are given in Figure 5. All the TVDs omitted in the figure are trivial in the sense that the predicate is not affected by the action. In order to simplify the presentation we give the TVDs in their generic form and did not sort the diagrams. Notice that the TVDs capture the implicit assumption that if the preconditions of the action are not satisfied then the action has no effect.

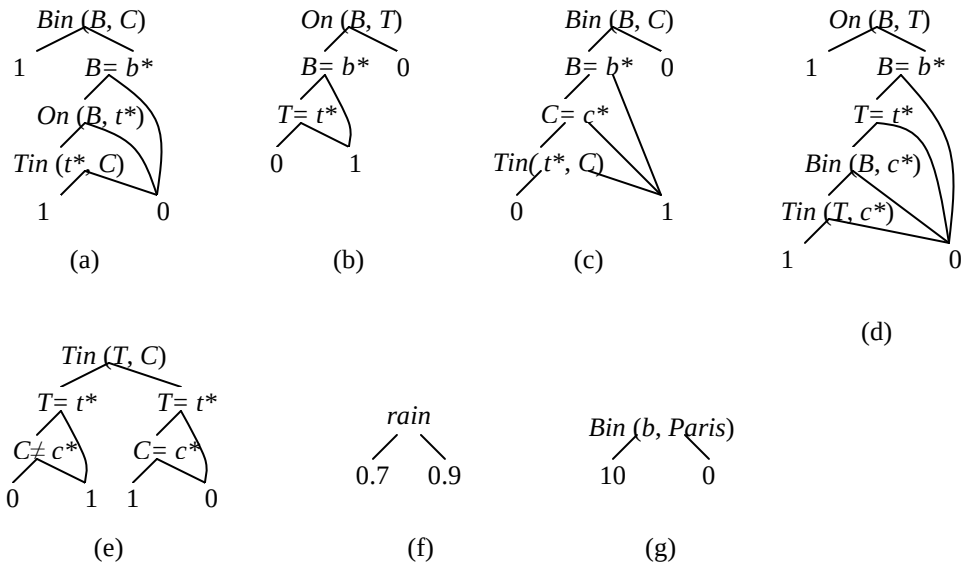

Figure 5. FODDs for logistics domain: TVDs, action choice, and reward function. (a)(b) The TVDs for $Bin(B, C)$ and $On(B, T)$ under action choice $unloadS(b^*, t^*)$. (c)(d) The TVDs for $Bin(B, C)$ and $On(B, T)$ under action choice $loadS(b^*, t^*, c^*)$. Note that c^* must be an action parameter so that (d) is a valid TVD. (e) The TVD for $Tin(T, C)$ under action choice $driveS(t^*, c^*)$. (f) The probability FODD for the action choice $unloadS(b^*, t^*)$. (g) The reward function.

Notice how we utilize the multiple path semantics with maximum aggregation. A predicate is true if it is true according to one of the paths specified so we get a disjunction over the conditions for free. If we use the single path semantics the corresponding notion of TVD is significantly more complicated since a single path must capture all possibilities for a predicate to become true. To capture that we must test sequentially for different conditions and then take a union of the substitutions from different tests and in turn this requires additional annotation on FODDs with appropriate semantics.

Probabilistic Action Choice: One can consider modeling arbitrary conditions described by formulas over the state to control nature's probabilistic choice of action. Here the multiple path semantics makes it hard to specify mutually exclusive conditions using existentially quantified variables and in this way specify a distribution. We therefore restrict the conditions to be either propositional or depend directly on the action parameters. Notice that under this condition any interpretation follows exactly one path (since there are no variables and thus only the empty valuation) so the aggregation function does not interact with the probabilities assigned. A diagram showing action choice for *unloadS* in our logistics example is given in Figure 5.

Reward and value functions: can be represented directly using algebraic FODDs. The reward function for our logistics domain example is given in Figure 5.

5. Value Iteration with FODDs

The general first-order value iteration algorithm works as follows: given as input the reward function R and the action model, we set $V_0 = R, n = 0$ and perform the following steps until termination:
(1) For each action type $A(\vec{x})$, compute:
$T_{n+1}^{A(\vec{x})}(V_n) = \gamma[\oplus_j(prob(A_j(\vec{x}))\otimes Regr(V_n, A_j(\vec{x})))]$.
(2) $Q_{n+1}^A = R \oplus \text{obj-max}(T_{n+1}^{A(\vec{x})}(V_n))$
(3) V_{n+1} is obtained by maximizing over Q_{n+1}: $V_{n+1} = \max_A Q_{n+1}^A$.

Regression by Block Replacement: We first describe the calculation of $Regr(V_n, A_j(\vec{x}))$ using a simple idea we call block replacement. We then proceed to discuss how to obtain the result efficiently.

Consider V_n and the nodes in its FODD. For each such node take a copy of the corresponding TVD, where predicate parameters are renamed so that they correspond to the node's arguments and action parameters are unmodified. BR-regress(V_n) is the FODD resulting from replacing each node in V_n with the corresponding TVD, with outgoing edges connected to the 0, 1 leaves of the TVD.

To see that block replacement is correct, consider an action from state $\hat{s}$ to state s. Notice that V_n and BR-regress(V_n) have exactly the same variables. Consider any valuation to these variables and the paths $P, \hat{P}$ followed under this valuation in the two diagrams.

By the definition of TVDs, the sub-paths of $\hat{P}$ applied to $\hat{s}$ guarantee that the corresponding nodes in P take the same truth values in s. So $P, \hat{P}$ reach the same leaf and the same value is obtained.

However, naive implementation of block replacement may not be efficient. If we use block replacement for regression then the resulting FODD is not necessarily reduced and moreover, since the different blocks are sorted to start with the result is not even sorted. Reducing and sorting the results may be an expensive operation. Instead we calculate the result as follows. For any FODD V_n we traverse BR-regress(V_n) using postorder traversal in term of blocks and combine the blocks. At any step we have to combine up to 3 FODDs such that the parent block has not yet been processed (so it is a TVD with binary leaves) and the two children have been processed (so they are general FODDs). If we call the parent B, the `true` branch child B_t and the `false` branch child B_f then we can represent their combination as $[B \times B_t] + [(1-B) \times B_f]$. It is easy to see that for any valuation the formula gives the same value as the original FODD. This is true since by fixing the valuation we effectively ground the FODD and all paths are mutually exclusive. We can therefore replace the combination of the 3 FODDs with the result of using several calls to Apply to combine them using the formula. Notice that different blocks share variables so we cannot perform weak reductions during this process. However, we can perform strong reduction in intermediate steps since it does not change the map for any valuation. After the process is completed we can perform any combination of weak and strong reductions since this does not change the map of the regressed value function.

Object Maximization: As mentioned above we get maximization over action parameters for free. We simply rename the action parameters using new variable names (to avoid repetition between iterations) and consider them as variables. The aggregation semantics provides the maximization. Since constants are turned into variables additional reduction is typically possible at this stage. Any combination of weak and strong reductions can be used.

Adding and Maximizing Over Actions: These can be done directly using the Apply procedure. Recall that $prob(A_j(\vec{x}))$ is restricted to include only action parameters and cannot include variables. We can therefore calculate $prob(A_j(\vec{x})) \otimes Regr(V_n, A_j(\vec{x}))$ in step (1) directly. However, the different regression results are independent functions so that in the sum $\oplus_j (prob(A_j(\vec{x})) \otimes Regr(V_n, A_j(\vec{x})))$ we must standardize apart the different regression results before adding

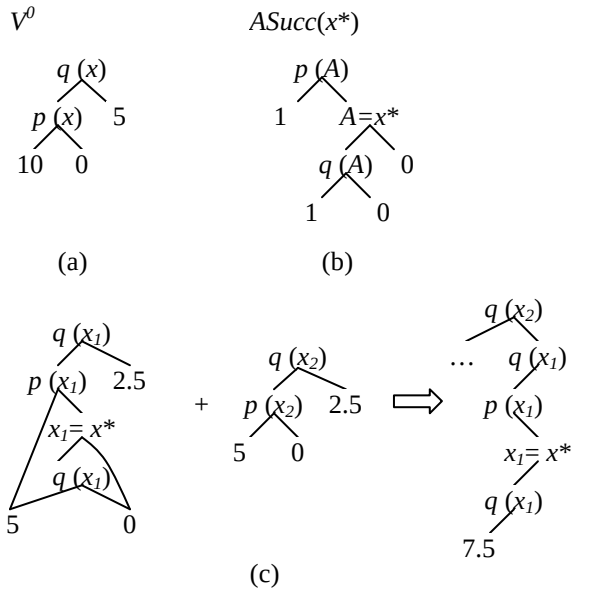

Figure 6. Example illustrating the need to standardize apart.

the functions (note that action parameters are still considered constants at this stage). Similarly the maximization $V_{n+1} = \max_A Q^A_{n+1}$ in step (3) must first standardize apart the different diagrams. The need to standardize apart complicates the diagrams and often introduces structure that can be reduced. In each of these cases we first use the propositional Apply procedure and then follow with weak and strong reductions.

Figure 6 illustrates why we need to standardize apart different action outcomes. Action A can succeed (denoted as $ASucc$) or fail (denoted as $AFail$, effectively a no-operation), and each is chosen with probability 0.5. Part (a) gives the value function V^0. Part (b) gives the TVD for $P(A)$ under the action choice $ASucc(x^*)$. All other TVDs are trivial. Part (c) shows part of the result of adding the two outcomes for A after standardizing apart (to simplify the presentation the diagrams are not sorted). Consider an interpretation with domain $\{1, 2\}$ and relations $\{q(1), p(2)\}$. As can be seen from (c), by choosing $x^* = 1$ i.e. action $A(1)$, the valuation $x_1 = 1, x_2 = 2$ gives a value of 7.5 after the action (without considering the discount factor). Obviously if we do not standardize apart (i.e $x_1 = x_2$), there is no leaf with value 7.5 and we get a wrong value. Intuitively the contribution of $ASucc$ to the value comes from the "bring about" portion of the diagram and $AFail$'s contribution uses bindings from the "not undo" portion. Standardizing apart allows us to capture both simultaneously.

Figure 7 traces several steps in the application of value

iteration to the logistics domain. In order to simplify the presentation the diagrams are not completely sorted allowing equalities in arbitrary locations. Block replacement is illustrated in the transitions (a) to (b) and (h) to (i). Comparing (e), (f) and (g) notice the use of Apply and Remove reductions.

6. Discussion

ADDs have been used successfully to solve propositional factored MDPs. Our work gives one proposal of lifting these ideas to FOMDPs. While the general steps are similar the technical details are significantly more involved than the propositional case. Our decision diagram representation combines the strong points of the SDP and ReBel approaches to FOMDP. On the one hand we get simple regression algorithms directly manipulating the diagrams. On the other hand we get object maximization for free as in ReBel. We also get space saving since different state partitions can share structure in the diagrams.

In terms of expressiveness, our approach can easily capture probabilistic STRIPS style formulations as in ReBel, allowing for more flexibility since we can use FODDs to capture rewards and transitions. However, it is more limited than SDP since we cannot use arbitrary formulas for rewards, transitions, and probabilistic choice. For example we cannot express universal quantification using maximum aggregation.

An implementation and empirical evaluation are obvious next steps. Any implementation can easily incorporate the idea of approximation by combining leaves with similar values (St-Aubin et al., 2000) to control the size of FODDs. The precise choice of reduction operators and their application will be crucial to obtain an effective system.

There are many open issues concerning the current representation and algorithms. It would be interesting to investigate conditions that guarantee a normal form for a useful set of reduction operators. Also, the representation can be improved to allow further compression. For example we are currently investigating the effect of allowing edges to rename variables when they are traversed so as to compress isomorphic sub-FODDs. It is also important to investigate the implications and limitations of separating the dynamics to probabilistic nature's choice and deterministic actions.

Acknowledgments

This work has been partly supported by NSF Grant IIS-0099446, and by a Research Semester Fellowship Award from Tufts University (Khardon).

References

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

Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press.

Blockeel, H., & De Raedt, L. (1998). Top down induction of first order logical decision trees. *Artificial Intelligence*, *101*, 285–297.

Boutilier, C., Reiter, R., & Price, B. (2001). Symbolic dynamic programming for first-order mdps. *Proceedings of the International Joint Conference of Artificial Intelligence*.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, *C-35*, 677–691.

Fern, A., Yoon, S., & Givan, R. (2003). Approximate policy iteration with a policy language bias. *International Conference on Neural Information Processing Systems*.

Gretton, C., & Thiebaux, S. (2004). Exploiting first-order regression in inductive policy selection. *Proceedings of the International Conference on Machine Learning*.

Groote, J. F., & Tveretina, O. (2003). Binary decision diagrams for first-order predicate logic. *The Journal of Logic and Algebraic Programming*, *57*, 1–22.

Guestrin, C., Koller, D., Gearhart, C., & Kanodia, N. (2003). Generalizing plans to new environments in relational mdps. *Proceedings of the International Joint Conference of Artificial Intelligence*.

Hoey, J., St-Aubin, R., Hu, A., & Boutilier, C. (1999). Spudd: Stochastic planning using decision diagrams. *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*.

Kersting, K., Otterlo, M. V., & Raedt, L. D. (2004). Bellman goes relational. *Proceedings of the International Conference on Machine Learning*.

McMillan, K. L. (1993). *Symbolic model checking*. Kluwer Academic Publishers.

Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning*, *2*, 229–246.

Sanghai, S., Domingos, P., & Weld, D. (2005). Relational dynamic bayesian networks. *Journal of Artificial Intelligence Research*, *24*, 759–797.

Sanner, S., & Boutilier, C. (2005). Approximate linear programming for first-order mdps. *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*.

St-Aubin, R., Hoey, J., & Boutilier, C. (2000). Apricodd: Approximate policy construction using decision diagrams. *International Conference on Neural Information Processing Systems*.

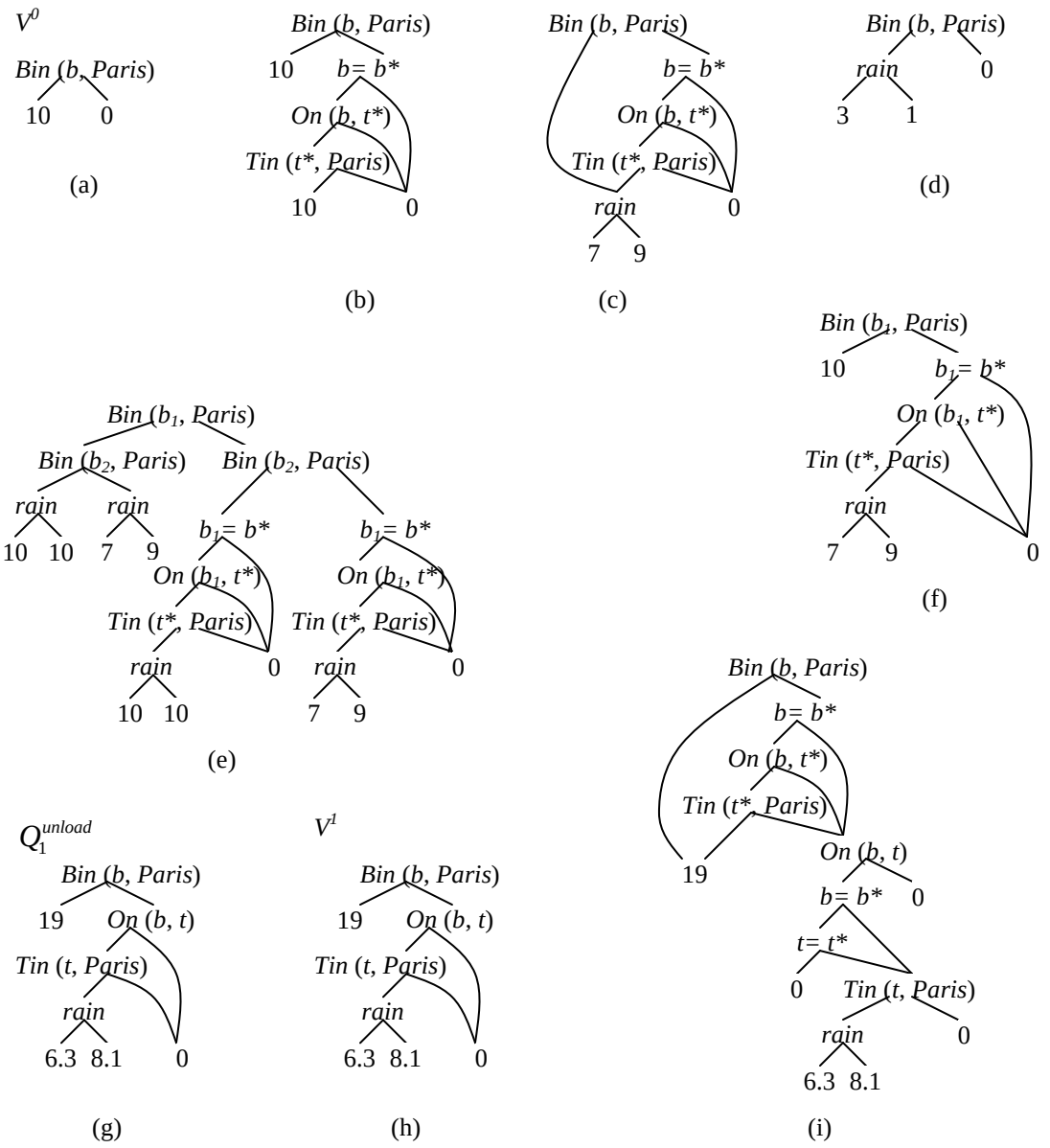

Figure 7. Regression in the Logistics Domain: (a) The value function V^0. (b) Regression of V^0 through $unloadS(b^*, t^*)$. (c) Multiply (b) with the choice probability FODD. (d) Regression of V^0 over $unloadF(b^*, t^*)$ multiplied with the probability. (e) The unreduced result of adding two outcomes for $unload(b^*, t^*)$. (f) The reduced result after addition. Notice that the left branch reduces to 10 by using both the recursive part of Apply and R6. The middle part is dropped by R7. (g) Multiply by $\gamma = 0.9$, perform object maximization, and add the reward to get Q_1^{unload}. Notice that in object maximization we have also dropped the equality on the right branch by the special case of R6. (h) V^1, the value function after first iteration. In this case the diagram for *unload* dominates the other actions (not shown). (i) Block replacement in computing V^2 through action $unloadS(b^*, t^*)$.