Say "No" to Grounding: An Inference Algorithm for First-Order MDPs

Olga Skvortsova
International Center for Computational Logic
Technische Universität Dresden
skvortsova@iccl.tu-dresden.de

Abstract

We propose an algorithm, referred to as ALLTHETA, for performing efficient domain-independent symbolic reasoning in a planning system FLUCAP that solves first-order MDPs. The computation is done avoiding vicious grounding.

Introduction

Markov Decision Processes (MDPs) are de facto standard representational and computational model for decision-theoretic planning problems. Recently, several compact representations for propositionally-factored MDPs have been proposed, including dynamic Bayesian networks (Boutilier, Dean, & Hanks 1999) and algebraic decision diagrams (Hoey *et al.* 1999). For instance, the SPUDD algorithm (Hoey *et al.* 1999) has been used to solve MDPs with hundreds of millions of states optimally, producing logical descriptions of value functions that involve only hundreds of distinct values.

Meanwhile, many realistic planning domains are best specified in first-order terms. However, most existing implemented solutions for first-order MDPs (FOMDPs) rely on grounding, i.e., eliminate all variables at the outset of a solution attempt by instantiating terms with all possible combinations of domain objects, e.g., (2002). This technique is very impractical because the number of propositions grows considerably with the number of domain objects and relations. This has a dramatic impact on the complexity of the algorithms that depends directly on the number of propositions. Moreover, as soon as the universe of objects is infinite, these algorithms cannot be made to work. Finally, systems for solving FOMDPs that rely on state grounding also perform action grounding which is problematic in first-order domains, because the number of ground actions also grows drastically with domain size.

To address these difficulties, we have recently proposed a first-order generalization of LAO* algorithm (Karabaev & Skvortsova 2005), referred to as FOLAO*, in which our contribution was to show how to perform heuristic search for FOMDPs, circumventing their grounding. In order to ensure first-order reasoning without descending to the propositional level, a planning system should be equipped with highly-optimized domain-independent inference algorithms that compute sets of successor and predecessor states of a given state wrt. a given action. Such inference algorithms rely on non-trivial symbolic computations as, e.g., unification or subsumption problem under some equational theory between two states specified as first-order terms.

In this paper, we develop an algorithm, referred to as ALLTHETA, that solves the subsumption problem under AC1[1] equational theory and delivers all possible substitutions. The computation is done avoiding aggressive grounding. ALLTHETA has been recently integrated into the planning system FLUCAP (Hölldobler, Karabaev, & Skvortsova 2006).

First-order Representation of MDPs

First, we propose a concise representation of FOMDPs within Probabilistic Fluent Calculus ($\mathcal{PFC}$). $\mathcal{PFC}$ is a logical approach to modelling dynamically changing and uncertain environments based on first-order logic (Hölldobler, Karabaev, & Skvortsova 2006).

MDPs An MDP is a tuple $(\mathcal{Z}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{C})$, where $\mathcal{Z}$ and $\mathcal{A}$ are finite sets of states and actions, resp.; $\mathcal{P} : \mathcal{Z} \times \mathcal{Z} \times \mathcal{A} \to [0, 1]$, written $\mathcal{P}(z'|z, a)$, specifies transition probabilities of reaching a state z' by executing a in z. $\mathcal{R} : \mathcal{Z} \to \Re$ is a real-valued reward function associating with each state z its immediate utility $\mathcal{R}(z)$. $\mathcal{C} : \mathcal{A} \to \Re$ is a real-valued cost function associating a cost $\mathcal{C}(a)$ to each action a. A solution of an MDP is a policy $\pi : \mathcal{Z} \to \mathcal{A}$ that maximizes the total expected discounted reward received when executing the policy π over an infinite horizon. The value of a state z with respect to a policy π is defined recursively as:

$$V_\pi(z) = \mathcal{R}(z) + \mathcal{C}(\pi(z)) + \gamma \sum_{z' \in \mathcal{Z}} \mathcal{P}(z'|z, \pi(z)) V_\pi(z'),$$

where $0 \leq \gamma < 1$ is a discount factor.

Probabilistic Fluent Calculus: Formally, let Σ denote a set of function symbols. We distinguish two function symbols in Σ, namely $\circ/2$ which is associative (A), commutative (C), and admits the unit element, and a constant 1. Let $\Sigma_{-} = \Sigma \setminus \{\circ, 1\}$. Non-variable Σ_{-}-terms are called fluents.

[1] A - associative, C - commutative, 1 - unit element.

Let $\mathcal{F}$ denote the set of fluents. Fluent terms are defined inductively as follows: 1 is a fluent term; each fluent is a fluent term; $F \circ G$ is a fluent term, if F and G are fluent terms.

A *state* is a fluent term. We assume that each fluent may occur at most once in a state, i.e., states of the form $euro \circ euro$ are disallowed. For example, a state $Z = on(X', Y') \circ on(Y', t) \circ cl(X') \circ e$ denotes that some clear block X' is on the block Y', which is on the table, the gripper is empty and something else might be also true. We note that the negation can be effortlessly included in the language (Hölldobler, Karabaev, & Skvortsova 2006). The interpretation over $\mathcal{F}$, denoted as $\mathcal{I}$, is the pair $(\Delta, \cdot^{\mathcal{I}})$, where the domain Δ is a set of all finite sets of ground fluents from $\mathcal{F}$; and an interpretation function $\cdot^{\mathcal{I}}$ which assigns to each state Z a set $Z^{\mathcal{I}} = \{d \in \Delta | \exists\theta.(Z \circ U)\theta =_{AC1} d\}$, where θ is a substitution and U is a new AC1-variable. Thus, states in $\mathcal{PFC}$ represent clusters of individual states. In this way, they embody a form of state space abstraction, referred to as first-order state abstraction, and, hence, can be treated as abstract states. E.g, the state $z_1 = on(b, c) \circ on(c, t) \circ cl(b) \circ e \circ cl(f)$, where t stands for table and b, c and f are blocks, is represented by the abstract state Z above; whereas $z_2 = on(b, c)$ is not, since other three 'mandatory' fluents of Z are missing in z_2. In essence, abstract states are defined under incomplete semantics, viz., other fluents that are not explicitly present in the state description might also hold, as e.g., $cl(f)$ appears in the state $z_1 \in Z^{\mathcal{I}}$.

Actions are first-order terms leading with an action function symbol. For example, the action of picking up some block X from another block Y might be denoted as *pickup* (X, Y). Stochastic actions are described via decomposition into deterministic primitives under nature's control, referred to as nature's choices. E.g., action *pickup* (X, Y) can be defined by means of successful *pickupS* (X, Y) and failure *pickupF* (X, Y) nature's choices. Preconditions and effects of an action a, denoted as *Pre* (a) and *Eff* (a), respectively, are abstract states. E.g., for preconditions and effects of the action *pickupS* (X, Y), we have: *Pre* $(\textit{pickupS}(X, Y)) := on(X, Y) \circ cl(X) \circ e$ and *Eff* $(\textit{pickupS}(X, Y)) := h(X)$, where $h(X)$ stands for the fact of holding a block X. Probabilities of each nature's choice, rewards and action costs can be defined in an obvious way.

An Inference Algorithm for FOMDPs

Systems for solving FOMDPs that rely on state grounding also perform action grounding which is problematic in first-order domains, because the number of ground actions grows drastically with domain size. Herein, we show how to perform inferences, i.e., compute successors and predecessors of a given abstract state, with action schemata directly, avoiding unnecessary grounding.

For this, an inference problem of finding all a-successors (all a-predecessors) of an abstract state Z is represented in terms of the AC1-unification problem[2], referred to as AC1-UNIFY(Z_1, Z_2), where Z_1 represents the preconditions (effects) of a and $Z_2 = Z$. AC1-UNIFY(Z_1, Z_2) is defined by: $\exists\theta.\ (Z_1 \circ U)\theta =_{AC1} (Z_2 \circ W)\theta$, where U and W are new AC1-variables.

Intuitively, an action a is applicable to an abstract state Z iff it is applicable to *all* individual states that constitute $Z^{\mathcal{I}}$. In order to determine all fragments of Z, an action a is applicable to, we compute all solutions for the following AC1-unification problem: $(\textit{Pre}\,(a) \circ U)\theta =_{AC1} (Z \circ W)\theta$. In this way, the bindings for W define the fragments $Z^i = (Z \circ W)\theta$ of Z, an action a is applicable to. Moreover, the bindings for U allow us to construct the successors of Z^i, i.e., $Z^i_{succ} := (\textit{Eff}\,(a) \circ U)\theta$. In essence, in order to compute the set of all a-successors of all fragments of Z, a is applicable to, it is enough to find all solutions θ for the above AC1-unification problem.

In this work, we present a restricted case of AC1-unification, i.e., AC1-subsumption, referred to as AC1-SUBSUME(Z_1, Z_2), where $(Z_2 \circ W)\theta = Z_2$:

$$\exists\theta.\ (Z_1 \circ U)\theta =_{AC1} Z_2\ .$$

There are at least two applications of AC1-SUBSUME(Z_1, Z_2) in the FOLAO* algorithm. First, for detecting a more specific abstract state between Z_1 and Z_2, that can be removed from the state space thereafter. Second, for computing a set of *all* states that are reachable from an initial state wrt. all actions.

In the following, we exploit the fact that the AC1-subsumption problem is a specialization of the θ-subsumption problem on general clauses, since abstract states are Horn clauses with empty head (Scheffer, Herbrich, & Wysotzki 1996). The θ-subsumption problem for clauses C and D is a problem of whether there exists a substitution θ such that $C\theta \subseteq D$ (or, in our terms, $(C \circ U)\theta =_{AC1} D$).

In general, θ-subsumption is NP-complete (Scheffer, Herbrich, & Wysotzki 1996). It is known that deterministic subsumption, i.e., when there exists an ordering of fluents, such that in each step there is a fluent which has exactly one match that is consistent with the previously matched fluents, can be solved in polynomial time. Unfortunately, in general, there are only few, or none at all, fluents in a state that can be matched deterministically.

Following (Scheffer, Herbrich, & Wysotzki 1996), we have developed two approaches to reduce the complexity of non-deterministic θ-subsumption, and hence, AC1-subsumption. Both approaches have been reconciled in an algorithm, referred to as ALLTHETA, that returns all solutions for the AC1-subsumption problem.

Phase one: context-based subsumption. One approach is context-based matching candidate elimination. In general, a fluent f in an abstract state Z_1 can be matched with several fluents in an abstract state Z_2, that are referred to as matching candidates of f. The approach is based on the idea that fluents in Z_1 can be only matched to those fluents in Z_2, the context of which include the context of the fluents in Z_1. The context is given by occurrences of identical variables or chains of such occurrences and is defined up to some fixed depth. In effect, matching candidates that do not meet the above context condition can be effortlessly pruned. In most cases, such pruning results in deterministic

[2] AC1-unification problem is a unification problem under the equational theory AC1.

subsumption, thereby considerably extending the tractable class of abstract states. Deterministic subsumption that exploits the context information is referred to as context-based deterministic subsumption.

For example, two abstract states $Z_1 = on(X,Y) \circ on(Y,t)$ and $Z_2 = on(a,b) \circ on(b,c) \circ on(c,t) \circ on(d,t)$ cannot be subsumed deterministically because each fluent in Z_1 has more than one matching candidate in Z_2. However, exploiting the context information already at depth 1 enables us to conclude that Z_1 subsumes Z_2. At depth 1, the context of $on(X,Y)$ contains the path $on \cdot 2 \rightarrow 1 \cdot on$, i.e., a variable Y appears at position 2 in $on(X,Y)$ and at position 1 in $on(Y,t)$. The context of $on(Y,t)$ contains the path $on \cdot 1 \rightarrow 2 \cdot on$, i.e., the variable Y appears at position 2 in $on(X,Y)$ and at position 1 in $on(Y,t)$. The contexts of the fluents in Z_2 are $\{on \cdot 2 \rightarrow 1 \cdot on\}$, $\{on \cdot 1 \rightarrow 2 \cdot on, on \cdot 2 \rightarrow 1 \cdot on\}$, $\{on \cdot 1 \rightarrow 2 \cdot on, on \cdot 2 \rightarrow 2 \cdot on\}$ and $\{on \cdot 2 \rightarrow 2 \cdot on\}$, resp. The fluent $on(Y,t)$ has two matching candidates, viz., $on(c,t)$ and $on(d,t)$. Since the context of $on(Y,t)$ can only be embedded in the context of $on(c,t)$, the matching candidate $on(d,t)$ is excluded and $on(Y,t)$ can be matched deterministically. Then, the matching substitution $\mu_1 = \{Y \mapsto c\}$ is applied to Z_1. As a result, the fluent $on(X,Y)\mu_1 = on(X,c)$ can be matched deterministically to $on(b,c)$ with $\mu_2 = \{X \mapsto b\}$. Hence, both fluents can be matched deterministically and the substitution $\theta = \mu_1\mu_2$ was found without backtracking.

There is a well-known tradeoff. The deeper inside the abstract state we look, thus devoting the considerable effort for computing the context itself, the higher the pruning rate is. Alternatively, if the depth value is underestimated, we save time and space for constructing the context but end up with a larger search space. Very often, the optimal depth has the value of 2.

Phase two: ALL-CLIQUES**.** In some cases, however, after performing the context-based deterministic subsumption, there still remain some fluents that cannot be matched deterministically. Thus, a remaining space of matching candidates has to be searched for a substitution. For this, a second approach that reduces the complexity of non-deterministic AC1-subsumption, referred to as ALL-CLIQUES, has been developed. ALL-CLIQUES is a modified version of its ancestor CLIQUE (Scheffer, Herbrich, & Wysotzki 1996), where all cliques are computed and additional pruning techniques have been developed in order to alleviate the search for substitutions.

ALL-CLIQUES exploits a well-known correspondance between the AC1-subsumption problem and the clique problem, i.e., a problem of finding a clique[3] of the fixed size in a graph. More precisely, an abstract state Z_1 subsumes an abstract state Z_2 iff there is a clique of size $|Z_1|$ in the space of matching candidates for fluents in Z_1. By the size $|Z|$, we mean the number of fluents comprising Z. The candidates that do not form a clique can be effortlessly excluded from the search space.

We start with constructing a substitution graph (V,E) for abstract states Z_1 and Z_2 with nodes $v = (\mu, i) \in V$, where

[3]A clique in a graph is a set of pairwise adjacent nodes.

Function `findPath(`V, E, *Paths*, v, *currPath*, i`)`

```
1  if valid(v) then
2      currPath:=currPath∪{v}
3      if i = |Z1| then
4          Paths := Paths ∪ {currPath}
5      else
6          foreach u = (μ', i+1) ∈ V with (v,u) ∈ E do
7              if clique(u, currPath) then
8                  findPath(V, E, Paths, u, currPath,
                   i+1)
9  else V := V \ {v}
10 return Paths
```

μ matches some fluent at position i in Z_1 to some fluent in Z_2 and $i \geq 1$ is referred to as a layer of v. Two nodes (μ_1, i_1) and (μ_2, i_2) are connected with an edge iff $\mu_1\mu_2 = \mu_2\mu_1$ and $i_1 \neq i_2$.

ALL-CLIQUES returns all paths *Paths* in the graph (V,E) that start at the first layer and form a clique of size $|Z_1|$. Its core is the function FINDPATH. If `valid(`v`)` is true, i.e., v has at least one edge to each layer, v is added to the current path *currPath*. If v is located at the last layer then *Paths* is updated with the *currPath*. Otherwise, if a next-layer neighbour u of v forms a clique with the nodes in *currPath*, i.e., `clique(`u, *currPath*`)` holds in line 7, then `findPath` is called recursively for u. The removal of invalid nodes in line 9 is a distinct feature of ALL-CLIQUES, which was not introduced before. Another important pruning technique, employed in ALL-CLIQUES, relies on the idea of a layered substitution graph. In contrast to (Scheffer, Herbrich, & Wysotzki 1996), we organize a substitution graph in layers, i.e., each node $v = (\mu, i) \in V$ belongs to a layer i. The layers should be visited in the order of their appearance. The layered architecture of the substitution graph is a natural way to avoid duplicate occurrences of the same clique in the set of all cliques. In effect, context-based determinacy and ALL-CLIQUES are combined into an algorithm, referred to as ALLTHETA, that delivers all substitutions for the AC1-SUBSUME(Z_1, Z_2) problem.

Experimental Evaluation

We demonstrate the advantages of using the context information for efficient domain-independent symbolic reasoning in FOMDPs on a system, referred to as ALLTHETA. ALLTHETA has been recently integrated as a module into the FLUCAP 1.1 planning system, that is a successor of FLUCAP 1.0 (Hölldobler, Karabaev, & Skvortsova 2006) that has entered the probabilistic track of the International Planning Competition IPC'2004. The experimental results were all obtained using a Linux RedHat machine running at 2.4 GHz Intel Celeron with 1 Gb of RAM.

Table 1 presents the comparison results of ALLTHETA with the system FASTTHETA (Ferilli *et al.* 2003) on the CBW dataset. CBW stems from the colored Blocksworld scenario that was first introduced during the IPC'2004. CBW is, currently, one of a few probabilistic scenarios that are represented in first-order terms and, hence, enable to make use

B	C	Total time, sec.						
			AllTheta					
		FTheta	d=0	d=1	d=2	d=3	d=4	d=5
5	3	0.5	2.9	0.4	0.3	0.3	0.4	1.0
	4	0.4	2.0	0.3	0.2	0.2	0.3	0.6
	5	0.4	1.7	1.3	0.2	0.2	0.2	0.5
10	3	1.5	44.7	1.1	0.5	0.5	1.0	4.3
	4	1.1	22.4	1.1	0.4	0.4	0.5	1.4
	5	0.9	13.5	1.0	0.5	0.5	0.8	3.1
15	3	3.9	n/a	2.3	0.9	0.9	1.7	7.7
	4	3.5	243.3	2.4	0.8	0.9	2.0	10.6
	5	2.8	84.7	2.0	0.7	0.7	1.2	4.9
20	3	8.7	n/a	10.1	4.6	3.1	4.2	15.7
	4	9.2	n/a	3.3	1.1	1.0	1.8	8.5
	5	7.3	n/a	3.0	1.0	1.1	2.1	11.6
25	3	16.5	n/a	7.2	2.0	1.8	4.1	28.3
	4	17.1	n/a	7.8	1.8	1.7	4.2	30.7
	5	15.7	n/a	7.3	1.7	1.8	4.2	34.0
50	3	164.9	n/a	n/a	38.8	29.5	28.6	52.2
	4	201.1	n/a	186.8	33.0	26.0	27.9	42.7
	5	175.1	n/a	140.4	30.8	26.3	29.1	57.7
75	5	702.5	n/a	240.8	58.0	47.2	52.3	121.8
100	5	n/a	n/a	452.6	96.7	78.1	74.0	155.0

Table 1: Performance comparison of ALLTHETA (denoted as AllTheta) with FASTTHETA (denoted as FTheta) on the CBW dataset.

of symbolic reasoning. CBW differs from the classical case in that, along with the unique identifier, each block is assigned a specific color. A goal formula, specified in first-order terms, provides an arrangement of colors instead of an arrangement of blocks. FASTTHETA, that is motivated by the field of Inductive Logic Programming (ILP), can be applied to compute all solutions of the AC1-subsumption problem.

In the following, we motivate the importance of the context depth parameter. Altogether, there are 100 abstract states that lead to 10000 subsumption tests. The column labelled Total time presents the time needed to solve all of 10000 subsumption tests. A 30-mins slot is allocated for each problem. The cells marked with 'n/a' mean that the limit was exceeded. Each CBW problem is defined by a number B of blocks and a number C of colors.

In CBW case, on small problems of size up to 25 blocks, the depth parameter d posesses the optimal value of 2. Whereas, on larger problems, this value grows. This reflects the necessity to store an additional context information about the fluents in an abstract state. The special case of d=0 means that no context information is considered. In comparison to ALLTHETA, the runtime of FASTTHETA grows considerably faster in the size of a problem. For example, at depth of 2, for the five-colored 15, 25 and 75 problems, FASTTHETA is by factor of 4, 8 and 15 slower. As a result, it could scale to problems up to the size of 75 blocks only. Whereas, the limit of ALLTHETA comprises 360 blocks.

Neither FASTTHETA nor ALLTHETA are sensitive to the number of colors in a problem. In contrast, grounding-based reasoners are severely affected by this parameter. The timing results for a special case of d=0 demonstrate the dramatic loss in runtime in comparison even with the case of d=1, where the context information about the direct neighbours of a fluent is counted.

Most importantly, present results indicate that the domain-independent inference algorithm ALLTHETA performs symbolic reasoning for first-order MDPs in about the same time as the domain-specific subsumption solver that was integrated in FLUCAP 1.0. We note that the latter reduces the AC1-subsumption problem to a quadratic variant of the subset problem. Whereas, the former solves the general case, which is NP-complete. For example, for a single subsumption test at depth of 2 in the problem of 15 blocks and 3 colors, ALLTHETA requires of about 92 microseconds. Whereas, for its domain-specific counterpart, the runtime comprises 85 microseconds. Finally, FASTTHETA has outperformed ALLTHETA by a factor of four, on the Mutagenesis dataset that is a classical ILP testbed.

Conclusions, Related and Future Work

We have proposed an algorithm, referred to as ALLTHETA, for performing automated domain-independent symbolic reasoning in FOMDPs. The construction is done avoiding grounding. In comparison to FASTTHETA, our approach scales better on larger FOMDPs. Some related approaches are known. For example, Django (Maloberti & Sebag 2004) is, nowadays, the fastest θ-subsumption checker that is based on the constraint satisfaction. Yet, it returns a binary answer 'yes/no' only and provides no solutions, even in the positive case. In (Kersting, van Otterlo, & de Raedt 2004), authors employ a generalized AC1-subsumption framework in the ReBel algorithm. ReBel treats abstract states as sets of fluents. Whereas, ALLTHETA can potentially work with multisets. We plan to incorporate these and disequalities into our setting. It is also important to extend our results towards the case of the AC1-unification problem.

Acknowledgements

We thank reviewers for their comments. Many thanks to Eldar Karabaev and Georg Rammé for fruitful discussions. This work is supported by the grant GRK 334 under auspices of DFG.

References

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural Assumptions and Computational Leverage. *JAIR* 11.

Feng, Z., and Hansen, E. 2002. Symbolic heuristic search for factored markov decision processes. In *AAAI*.

Ferilli, S.; Di Mauro, N.; Basile, T.; and Esposito, F. 2003. A complete subsumption algorithm. In *AI*IA*.

Hoey, J.; St-Aubin, R.; Hu, A.; and Boutilier, C. 1999. SPUDD: Stochastic Planning using Decision Diagrams. In *UAI*.

Hölldobler, S.; Karabaev, E.; and Skvortsova, O. 2006. FLUCAP: A heuristic search planner for first-order MDPs. *JAIR*. To appear.

Karabaev, E., and Skvortsova, O. 2005. A Heuristic Search Algorithm for Solving First-Order MDPs. In *UAI*.

Kersting, K.; van Otterlo, M.; and de Raedt, L. 2004. Bellman goes relational. In *ICML*.

Maloberti, J., and Sebag, M. 2004. Fast theta-subsumption with constraint satisfaction algorithms. *ML* 55(2).

Scheffer, T.; Herbrich, R.; and Wysotzki, F. 1996. Efficient θ-subsumption based on graph algorithms. In *ILP Workshop*.