

# Automated Conversion and Simplification of Plan Representations

Martin Allen and Shlomo Zilberstein

Department of Computer Science, University of Massachusetts, Amherst, MA 01003  
{mwallen, shlomo}@cs.umass.edu

## Abstract

*As planning agents grow more sophisticated, plan representation issues arise. Planners work over increasingly large and difficult problems and output is often complex or unwieldy. Further, where planners must interact with human users—either for plan verification and analysis, or in mixed-initiative settings—plans must be represented so that the intended course of action is readily available. We propose automated techniques for simplification of, and conversion between, plan representations.*

## 1. Introduction

Planning research tries to discover and describe correct or optimal courses of action, tending to neglect other aspects of resulting plans, such as compactness, storage- or transmission-cost, comprehensibility, or the ability to verify the plan or to recover from failure. As planning algorithms improve in their ability to solve large, realistic problems, so grow the complexity and size of the resulting plans. Many existing techniques produce plans that are optimal, but rank poorly in terms of other qualities. For many purposes, however, plan representation is of the essence. NASA’s Mars rover project is an example. Existing rovers have limited memory and processing power, and constrained bandwidth for remote communication. Combined with the need for in-depth checking of plans prior to execution, this requires that plans for rover operation be represented simply and compactly [1]. Similarly, *mixed-initiative* planning [2], in which planners work in concert with human beings, requires that interim plans be represented so that they are comprehensible to these users. In such contexts, optimal probabilistic planning methods have been considered infeasible, due to the tendency for their output to be large and unwieldy.

## 2. Generating Plan Representations

Our work focusses on conversion between optimal MDP solutions and graph-based contingency plans. To solve large

problems efficiently, MDPs are represented in factored form using Algebraic decision diagrams. Since the cost of operations on ADDs is proportional to their size, MDPs with large state spaces can be solved efficiently if encoded using relatively compact diagrams. We investigate simple transformations that effectively reduce the size of output policies in this context, and convert them into compact graphical plans, better-suited for such purposes as plan analysis.

We solve MDPs using Symbolic-LAO\* (SLAO\*), which combines the use of ADDs with reachability analysis and heuristic search to limit value-updates to only those states strictly necessary for calculating the optimal policy [3]. SLAO\* produces an MDP policy in the form of an ADD mapping sets of states to actions. Leaf-nodes of this ADD are labelled with actions  $\{a_u, a_0, \dots, a_n\}$ . These actions comprise an optimal policy, with the exception of  $a_u$ , a special place-holder; since some values of the variables necessary to compute the policy are never actually visited by the policy, these states are safely assigned an “unknown” action, without concern for sacrificing optimality. Figure 1 shows an example for a relatively simple instance of a simulated Mars rover problem with  $2^{12}$  states, given by variables for current time, rover position, and action duration. We write ADDs as binary trees with outputs at the leaves, using a solid edges for true values, and a dashed edge where false. Here, for instance, we “collect” in any state corresponding to the left-most path in the ADD. On the other hand, as shown by the right-most path, the optimal policy never leads to any state in which  $d1$  and  $d5$  are both true, and all such states are assigned the “unknown” action.

While the ADD version of this policy is reasonably compact, solutions for problem-domains with a larger number of variables and many possible actions will generally be far more complicated, featuring many hundreds of nodes and complicated paths. Indeed, SLAO\*, like most algorithms, is ultimately indifferent to the representational features of its output, beyond those necessary to represent the solution correctly. Furthermore, even this simple policy tells us nothing about such things as the *order* in which the rover will perform its actions. Policies of this sort are therefore of limited usefulness for straightforward plan analysis and verification; human users are provided with little information that

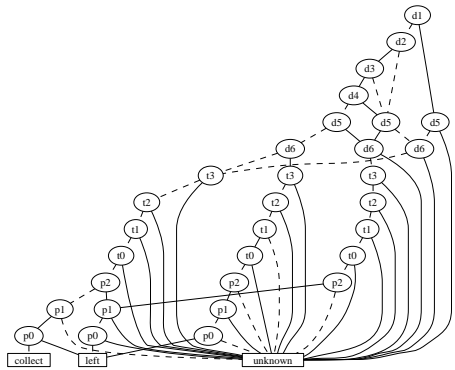


Figure 1. A sample policy output by SLAO\*.

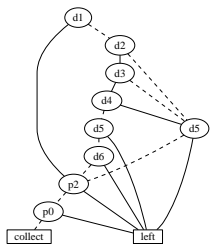


Figure 2. Pruned policy from Figure 1.

might allow them, for instance, to identify errors in the initial specification of the planning domain.

### 3. Converting Plan Representations

As explained, SLAO\* assigns unreachable states a null “unknown” action. The resulting plan is still optimal, precisely because such states are unreachable. In fact, optimality is preserved *no matter which* actions are assigned such states. We thus implement a simple recursive pruning algorithm, which preserves optimality, while trimming “unknown” branches by assigning them to one of the other genuinely available actions. Figure 2 shows the result of applying it to the ADD from Figure 1. Unreachable value-combinations have dropped out; it is straightforward to verify that old and new policies dictate the same action for any reachable state. Computing a minimal decision diagram without “unknown” actions is NP-complete [4]. Thus our one-pass pruning method can only be heuristic; however, experimental results show it to be effective in reducing output policy size by 30–90% over a range of cases.

While the pruning method just given reduces the size of the policy output by our MDP solver, it does nothing to answer questions about the relative ordering of actions and outcomes in the plan. For such purposes, MDP policies are not natural candidates; we therefore investigate methods

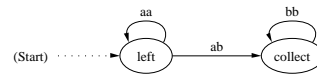


Figure 3. Plan created from policy of Figure 2.

for converting such policies into useful *graph-based contingency plans*. Each such plan consists of a directed graph  $(V, E)$  with starting vertex  $v_0$ . Each vertex  $v \in V$  is labelled with an *action*. Edges  $e \in E$  are labelled with a *set of states* (here in the form a characteristic function, computed by an ADD). To follow such a plan, one starts in vertex  $v_0$ , takes associated action  $a_0$ , and observes state  $s'$  that results; one then follows the edge labelled with the set containing  $s'$ , and repeats the process. Such plans are generated using a combination of set-theoretic and reachability operators, all readily available in efficient form for ADDs.

Figure 3 shows the result of applying this method to the policy considered in Figure 2. Evidently, the plan of action is a sequence of leftward movements, followed by a period of sample collection before time expires. So, even this simple example reveals an important difference between graph-based contingency plans and the original MDP policies. Since the original policy in Figure 1 does not carry with it any explicit information about reachability, the actual behavior to be expected from the rover cannot be read off the policy in the way that it is made visible by the corresponding plan. Graph-based contingency plans can thus serve the purposes of plan analysis and verification more directly.

We continue to investigate techniques and operators for the manipulation of plan representations. After initial generation, such simple contingency plans can be re-structured in accord with a variety of relevant measures of plan complexity. In addition, our work opens up the possibility of algorithms that make trade-offs between plan optimality and representational clarity.

### References

- [1] J. Bresina, R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proc. 18th Conf. on Uncertainty in AI*, Edmonton, Alberta, 2002.
- [2] Mark Burstein and Drew McDermott. Issues in the development of human-computer mixed-initiative planning. In B. Gorayska and J. L. Mey, editors, *Cognitive Technology*, pages 285–303. Elsevier, 1996.
- [3] Zhengzhu Feng and Eric A. Hansen. Symbolic heuristic search for factored Markov decision processes. In *Proc. 18th Natl. Conf. on Artificial Intelligence*, pages 455–460, 2002.
- [4] Martin Sauerhoff and Ingo Wegener. On the complexity of minimizing the OBDD size for incompletely specified functions. *IEEE Trans. CAD*, 15(11):1435–1437, Nov. 1996.