Completable Planning: A Curative Learning Approach to the Imperfect Theory Problem

Melinda T. Gervasio and Gerald F. DeJong

University of Illinois at Urbana–Champaign
Beckman Institute for Advanced Science and Technology
405 North Mathews Avenue, Urbana, Illinois 61801
gervasio@cs.uiuc.edu dejong@cs.uiuc.edu

Abstract

The complexity of the real world makes a perfect characterization impossible. Purely deliberative approaches such as classical planning are thus susceptible to unexpected failures. Preventive learning approaches address the imperfect theory problem through the diagnosis of such failures and the determination of fixes to avoid similar failures in the future. Curative learning approaches such as completable planning instead treat failures as alternative outcomes and learn alternative plans to the recover from the failures. Through learning, a completable planner learns to plan only for the more likely outcomes in the particular problem distribution it faces. It thus significantly reduces the cost of disjunctive planning.. As a curative learning approach, it is also better suited to domains where outright failures are unacceptable or failure diagnosis is expensive, making preventive learning infeasible. Completable planning is a general incremental learning approach whose success in real world domains may be increased through the integration of other learning techniques and the consideration of different planning perspectives.

Introduction

Real world planners must live with imperfect information. Consider a robot navigation system given the simple task of moving to a specific location down the hallway. A traditional classical planning representation of this problem involves information regarding the robot's initial location and its goal location It also includes some characterization of a MOVE operator, with parameters such as speed, distance, start time, and stop time. This representation is imperfect in many ways. If the robot is unstable in its initial position, then it may inadvertently change locations between the start of planning and the start of execution. For example, if one of its wheels is teetering on the edge of a raised platform, it may fall off even before attempting the move. If the robot calculated a specific distance to move based on its given initial location, then the success of the move action will be affected as well. Furthermore, movement by dead reckoning will only succeed if the calculated movement parameters account for all possible factors– the speed of the motors, wheel slippage during starting and stopping, wheel alignment, air resistance, and so on, not to mention potential interference from other active agents in the vicinity.

Problems with Imperfect Theories

The *frame problem* [McCarthy69] guarantees that it is impossible to completely consider all the factors possibly affecting the effects an action. For STRIPS-type operators, this incompleteness may be manifested in the form of an incomplete set of preconditions. This results in two types of execution failure. First, the action may fail to achieve its intended effects because a necessary but unspecified precondition did not hold immediately prior to executing the action. Second, the action may result in additional unintended effects that clobber previously achieved preconditions for subsequent actions. With a domain theory consisting of actions with incomplete preconditions, a planner may construct plans which the domain theory supports as being correct but which fail to achieve their goals.

Preventive Learning

While a perfect symbolic representation of a domain is unattainable, learning may enable real world planners with imperfect domain theories to construct plans with high probabilities of success. Previous learning approaches to the problem of imperfect domain theories have been *preventive* (e.g. [Bennett90, Chien94, DeJong93, Gil93, Mostow87]). These approaches rely failure diagnosis to explain the cause of any failures encountered during execution. Similar future failures can then be *prevented* by fixing the cause of failure–e.g. retracting a faulty assumption, adding a missing precondition, or changing acceptable parameter ranges.

Learning is appropriate for planning domains where problems fall into problem classes, and the plans for the problems of a problem class have much in common. General plan segments or planning techniques may then be extracted from specific plans and used for other problems in the problem class. A *plan schema* is a general plan which may be instantiated into specific plans for a problem class. A plan schema may be associated with a *success rate*. This is the probability that when the schema is applied to a problem, the specific instantiation will achieve the problem goals.

A preventive plan schema learner may be characterized as increasing the success rate of a plan schema by preventing its application plans to problems where it will not succeed. In the limit, the believed applicability of a plan schema closes in on the problems for which instantiating the schema results in a

plan which will achieve the problem goals. An alternative approach is to increase the success rate of a plan schema by augmenting it with additional or other actions to execute in the case when actions have unexpected effects. In this approach, unexpected effects are not treated as failures but rather as situations for which alternative plans are to be constructed. In the limit, the plan schema will succeed for the range of problems within its applicability set. Because this approach provides treatment to recover from failures, we call it the *curative* approach.

Completable Planning

Completable planning is a curative learning approach to the problem of imperfect domain theories. Completable planning begins with the construction of a *base plan*, which is a completely ordered sequence of operators. The base plan may be constructed using any standard classical planner, including partial-order planners (e.g. [Chapman87, McAllester91, Penberthy92]). However, a permanent complete ordering must then be imposed. This is because different orderings may result in different sets of failures. Allowing arbitrary orderings would compromise the usefulness of any plans constructed in response to failures.

Every step in the base plan is associated with a particular set of expectations. These are the conditions that, according to the domain theory of the planner, will be true and must be true for the achievement of the problem goals. Minimally, the expectations associated with a plan step are its preconditions, since these are the conditions believed to be necessary for the action to achieve its effects. Maximally, the expectations associated with a plan step is the regression of the goal over the tail of the plan beginning with the action. If these conditions hold immediately prior to executing the action, then according to the domain theory, the preconditions of every remaining action will be satisfied. Thus, the goal ultimately will be achieved.

During execution, an attempt is made to verify the expectations associated with each step immediately prior to the step's execution. If the expectations are met, execution continues. If the expectations are not met, *contingent planning* is invoked. The planning component is called to construct an alternative plan for achieving the goal from the current state. The contingent plan may branch back into the base plan at any point to reuse some of the actions in the plan. If a contingent plan is successfully constructed, the new segment is generalized and added into the base plan as a new alternative branch. Execution proceeds until either the goal is reached or contingent planning fails. [Gervasio94] elaborates on the information–gathering issues of completable planning.

Completable planning employs a probably approximately correct (PAC) learning algorithm. A solution is defined with respect to θ (minimum goal success rate), X (maximum execution length), and Y (maximum per episode planning cost). A completable plan (i.e. a base plan with augmentations) is a solution if every augmentation is constructed with cost $\leq Y$, every successful unfolding achieves the goal in $\leq X$ actions, and the probability of plan success is $\geq \theta$. Given a base plan and PAC parameters ϵ (error), δ (confidence), and γ (solvability), if the the probability of generating a solution is $\geq \gamma$, then with probability $\geq 1-\delta$ the algorithm outputs a plan with success $\geq \theta - \epsilon$. The number of examples required to generate a solution is polynomial in $\frac{1}{\delta}$, $\frac{1}{\epsilon}$, and $\frac{1}{\gamma}$. The completable planning approach is thus an attractive, tractable learning approach to the problem of imperfect domain theories.

Feasible Disjunctive Planning

Disjunctive planning [Draper94, Peot92, Warren76] deals with the problem of incompletely specified preconditions by allowing actions to be characterized as having multiple possible outcomes. Goal achievement is then guaranteed by determining a contingency plan for each possible outcome of every action in a plan. However, with this branching potentially occurring for every action in a plan as well as every action in every contingency plan and so on, disjunctive planning quickly becomes unwieldy. With fewer preconditions to achieve, planning in the backward direction is decreased. However, with multiple possible outcomes, planning in the forward direction is increased. Planning approaches to making disjunctive planning more feasible have included associating probabilities with the different outcomes and planning only for the most probable ones, as in [Drummond90], and using control rules to avoiding useless search, as in [Genesereth93]. These approaches buy feasibility at the cost of an increased domain engineering burden.

Completable planning increases the feasibility of the disjunctive planning approach in three ways. First, the requirement of specifying all the possible outcomes of every action is eliminated. The burden on the knowledge engineer is thus reduced. Second, contingent planning cost is reduced because the completable planner plans only in response to failures. Thus, planning resources are expended only on those outcomes likely to occur in the particular problem distribution the system is facing. Third, total planning cost is reduced because of learning. This allows the cost associated with constructing a particular schema or plan segment to be amortized over many applications. The increase in feasibility comes at the cost of learning.

Curative Learning vs. Preventive Learning

Crucial to the preventive learning approach is a good failure diagnosis component. Without one, the determined cause of failure may be incorrect. Consequently, the attempted fixes may be insufficient for preventing future similar failures. Unfortunately, failure diagnosis is a difficult problem, not unlike the planning problem. The preventive learning approach is thus faced with a similar quality/tractability tradeoff.

In contrast, the curative learning approach does not require additional knowledge for failure diagnosis and subsequent domain theory repair. In problem scenarios where pinpointing the cause of failure is difficult, completable planning may be the better choice. For example, consider a mobile system

given the task of exploring a partially-charted planet. Because the information about the domain is incomplete, the system cannot anticipate all the situations which might occur. Thus, its initial plan may not work. Upon encountering a hostile life form, a completable planning system could decide to employ defensive or evasive maneuvers to handle the situation. It can do this without having to understand why it crossed paths with the other agent. It only has to realize the anticipated encounter and figure out how to deal with it. The preventive alternative is not only more expensive, but potentially dangerous as well, as it may leave the system vulnerable while it attempts to explain the unexpected situation.

A curative learning approach such as completable planning can immediately use the results of its learning. Individual failures thus do not automatically result in a failure to achieve the problem goals. This makes completable planning particularly attractive for problem domains where achieving the goal is paramount. In a factory, for example, machines may break down, schedules may be delayed, and particular raw materials may be used up. A preventive solution would involve stopping production whenever such unexpected events occurred, determining the cause of failure, installing preventive measures, and starting over. On the other hand, a completable planner could learn ways to handle the failures and continue production. Some cost would be incurred by the delay, but production need not come to a standstill and incur even greater costs.

The savings achieved by eliminating failure diagnosis do come at the cost of absorbing unrecoverable failures. While a preventive approach may result in a handful of robots falling off a cliff, a curative approach is susceptible to sending whole armies into the ravine, as it never learns why the robots are getting too close to the edge. Because it does not alter the flawed domain theory, the completable planning approach is much more sensitive to the quality of the given domain theory. However, unless the given domain theory is pathologically incorrect, the completable planning approach will result in greater plan success.

Beyond Well-Behaved Environments

Completable planning was developed for *well-behaved* environments for which often-correct though imperfect domain theories can be constructed. For example, an office environment is well-behaved. Each room is expected to contain objects such as desks, chairs, and computers. However, these may range in size and type as well as position and location. Floors may be known to either be carpeted or tiled. But the particular carpet fiber or tile material may be unknown. The domain is well-behaved because ignorance of the precise flooring material, for example, does not drastically affect the effects of moving across the floor. An office courier attempting to move four feet may result in moving four feet and two inches on freshly-waxed tile or three feet and six inches on plush carpet. But it is unlikely to result in the courier turning two corners, entering another room, and crashing through the window.

The more well-behaved a domain, the better the performance of completable planning. Experiments with a variety of simulated domains, including robot navigation, a toolbox world [Krebsbach92], and parts assembly confirm that average execution length becomes longer as the probability of expected outcomes decreases. A greater range of possible outcomes also increases the planning cost, associated with a plan schema. The learning cost is consequently increased as well. The upper bound on the success rate of a completable planner is also affected by the probability of unrecoverable failures.

There are two ways in which completable planning may be more successfully adapted to less well-behaved domains. First, some aspects of preventive learning may be adopted for the purpose of improving the initial domain theory. In the exploration and manufacturing examples presented earlier, curative learning techniques could be used to immediately handle unexpected situations. Meanwhile, a deeper analysis could be performed offline for the purpose of determining more long-term solutions. These could then be installed later at a more convenient time. The complementary strengths of preventive and curative learning approaches make a hybrid approach attractive.

A second way to make completable planning more applicable to less well-behaved domains is to alter the planning component. Completable planning was designed around a classical, deliberative planner. By using planners more suited to dynamic, unpredictable environments, the completable planning approach may be made more applicable to less well-behaved domains. For example, the system could employ an approach interleaving planning and execution. Instead of learning monolithic plan schemas, a completable planner could learn smaller general plan segments. This would be particularly useful in domains where different plans often contain similar segments but are rarely exactly alike. For example, the daily plan for a pickup and delivery service will always involve transporting packages between destinations, but the destinations as well as the packages will change. Through completable planning, an interleaving system could learn general segments such as transporting a package from the east side to the west side. These segments could then be combined as appropriate to the other pickup and delivery tasks of the day. Developing the completable planning approach through the investigation of different learning techniques and planning paradigms raises many interesting issues for future research.

Acknowledgments. This research was supported by the Office of Naval Research under grant N-00014-94-I-0684.

References

[Bennett90] S. Bennett, "Reducing Real–world Failures of Approximate Explanation–based Rules," *Proceedings of the Seventh International Conference on Machine Learning*, Austin, TX, 1990, pp. 226–234.

[Chapman87] D. Chapman, "Planning for Conjunctive Goals," *Artificial Intelligence 32*, 3 (1987), pp. 333–378.

[Chien94] S. Chien and G. DeJong, "Constructing Simplified Plans via Truth Criteria Approximation," *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, Chicago, IL, June 1994, pp. 19–24.

[DeJong93] G. F. DeJong, "Learning to Plan in Continuous Domains," *Artificial Intelligence 64*, 2 (1993),.

[Draper94] D. Draper, S. Hanks and D. Weld, "Probabilistic Planning with Information Gathering and Contingent Execution," *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, Chicago, IL, June 1994, pp. 31–36.

[Drummond90] M. Drummond and J. Bresina, "Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction," *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, August 1990, pp. 138–144.

[Genesereth93] M. Genesereth and I. Nourbakhsh, "Time–Saving Tips for Problem Solving with Incomplete Information," *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, July 1993, pp. 724–730.

[Gervasio94] M. T. Gervasio and G. F. DeJong, "An Incremental Learning Approach for Completable Planning," *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ, July 1994, pp. 78–86.

[Gil93] Y. Gil, "Efficient Domain–Independent Experimentation," *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MD, June 1993, pp. 128–134.

[Krebsbach92] K. Krebsbach, D. Olawsky and M. Gini, "An Empirical Study of Sensing and Defaulting in Planning," *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, College Park, MD, June 1992, pp. 136–144.

[McAllester91] D. McAllester and D. Rosenblitt, "Systematic Nonlinear Planning," *Proceedings of the Ninth National Conference on Artificial Intelligence*, Anaheim, CA, July 1991, pp. 634–639.

[McCarthy69] J. McCarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence,"in *Machine Intelligence 4*, B. Meltzer and D. Michie (ed.), Edinburgh University Press, Edinburgh, Scotland, 1969.

[Mostow87] J. Mostow and N. Bhatnagar, "Failsafe – A Floor Planner that Uses EBG to Learn from its Failures." *Proceedings of the Tenth International Conference on Artificial Intelligence*, Milan, Italy, August 1987.

[Penberthy92] J. S. Penberthy and D. S. Weld, "UCPOP: A Sound, Complete, Partial–Order Planner for ADL," *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, October 1992, pp. 103–114.

[Peot92] M. A. Peot and D. E. Smith, "Conditional Nonlinear Planning," *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, College Park, MD, June 1992, pp. 189–197.

[Warren76] D. H. Warren, "Generating Conditional Plans and Programs," *Proceedings of AISB76*, Edinburgh, 1976, pp. 344–354.