PERMISSIVE PLANNING: A MACHINE LEARNING APPROACH TO LINKING INTERNAL AND EXTERNAL WORLDS

Gerald DeJong dejong@cs.uiuc.edu
Computer Science / Beckman Institute
University of Illinois at Urbana/Champaign
405 N. Mathews, Urbana IL 61801

Scott Bennett bennett@sra.com
Systems Research and Applications Corporation
2000 15th St. North
Arlington VA 22201

Abstract

Because complex real–world domains defy perfect formalization, real–world planners must be able to cope with incorrect domain knowledge. This paper offers a theoretical framework for *permissive planning*, a machine learning method for improving the real–world behavior of planners. Permissive planning aims to acquire techniques that tolerate the inevitable mismatch between the planner's internal beliefs and the external world. Unlike the reactive approach to this mismatch, permissive planning embraces projection. The method is both problem–independent and domain–independent. Unlike classical planning, permissive planning does not exclude real–world performance from the formal definition of planning.

Introduction

An important facet of AI planning is *projection*, the process by which a system anticipates attributes of a future world state from knowledge of an initial state and the intervening actions. A planner's projection ability is often flawed. A classical planner can prove goal achievement only to be thwarted by reality. The reactive approach, which has received much attention, avoids these problems by reducing reliance on projection or disallowing it altogether. For all its stimulating effect on the field, however, reactivity is only one path around projection problems. It is important to continue searching for and researching alternatives. In this paper we advance one such alternative termed *permissive planning*. In some ways it is the dual of the reactive approach, relying heavily on a goal projection ability enhanced by machine learning. From a broader perspective, permissive planning embodies an approach to inference which integrates empirical observations into a traditional *a priori* domain axiomatization.

The research reported in this paper was carried out at the University of Illinois and was supported by the Office of Naval Research under grant N0001491–J–1563. The authors also wish to thank Renee Baillargeon, Pat Hayes, Jon Gratch and the anonymous reviewers for helpful comments.

Permissive Planning

The real world is captured within an inference system by some description in a formal language such as the predicate calculus. The system's internal description may only approximate the external real world, giving rise to discrepancies between inferred and observed world behavior. In classical planning, the difficulties with projection can be traced to such a discrepancy, in this case, a discrepancy between an action's *internal* definition (the one represented and reasoned about) and its *external* definition (the mapping enforced by the real world). To concentrate on the discrepancies of action definitions, we will assume in this paper that no difficulty is introduced by the system's sensing or state representation abilities. Then, for simplicity in our figures, we can employ a single universe of states to denote both internal and external sets of states. Figure 1 illustrates a difference between a plan's projected and actual mappings from an initial state. The dot labeled *initial state* represents both a particular configuration of the world (which we call the external state) and the system's formal description of it (the internal state). According to the system's internal model, the plan's action sequence transforms the initial state into a goal state. In the real world, however, the actual final state falls well outside the goal region.

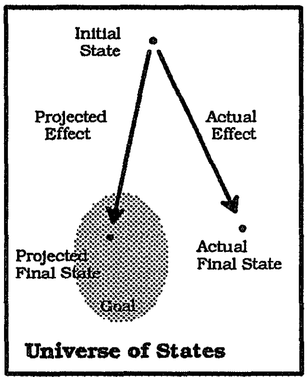

Figure 1: An Action Sequence's Projected and Actual Mappings

One might employ machine learning to improve the system's operator definitions. The result would be a more faithful representation of their real–world effects. This would yield a more accurate projection ability as illustrated in Figure 2. Un-

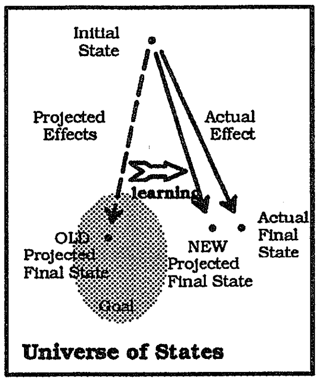

Figure 2: Conventional Learning Adjusts Projected Mapping towards Actual Mapping

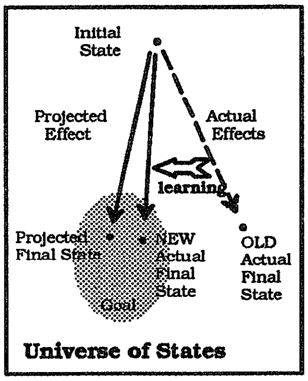

Figure 3: Permissive Planning Adjusts Actual Mapping towards Projected Mapping

fortunately, a trend toward increasingly correct operator definitions is necessarily also a trend towards more complex operator definitions. Increased complexity results in more work for the planner which must rule out concomitantly more negative interactions. Indeed, operator complexity can grow unboundedly as dictated by the qualification problem [McCarthy80]. With sufficient experience, this use of machine learning could paralyze any planner.

There is an alternative machine learning approach. When an apparently sound plan fails to achieve its goal in the real world, it may be possible to find a small alteration of the plan which tends not to influence the *projected* final state but moves the *actual* final state closer to the projected one. Instead of altering the system's representations to better fit the observed world (Figure 2), this approach alters the actions selected so the real–world effects better match the projected effects. This is illustrated in Figure 3. If such a plan can be found, the planner itself might be altered so that when presented with similar future problems, the planner will tend to produce a solution similar to the improved plan rather than repeating the mistakes of the the original plan. Our approach is to alter systematically the planner in response to execution failures so the planner prefers not to employ its domain knowledge in ways that seem to lead to failure. We call the approach "permissive" because it allows the planner to construct plans that work in the real world in spite of flaws in its domain knowledge.

Permissive Planning Principle: Blame the plan and adjust the planner in response to execution failures even though the implementor–supplied domain theory is known to be at fault.

Planner Bias and Permissive Planning Adjustment

In response to the need for breakfast a planner may be able to formulate several acceptable action sequences. One results in pancakes, another in cold raisin bran cereal, still another in hot oatmeal, etc. Of course, most planners will not produce all possible plans. Planning activity typically ceases when the first acceptable solution is found. After constructing an acceptable plan for a hot oatmeal breakfast, the system should not waste effort in working out the details for cooking pancakes.

We call the set of all possible plans that a particular classical planner could produce in principle for a problem, the *competence set* of the planner for that planning problem. We call the particular element of this set which is in fact constructed in response to the problem the *performance* of the planner for that planning problem. We use the term ***planner bias*** to refer to the preference, no matter how it is realized, for the particular performance plan from the planner's competence set. By systematically altering a planner's bias in response to observed plan execution failures, the same planning competence can yield quite different planning performance, resulting in improved real–world behavior. The permissive adjustment of a planner's bias so as to improve the real–world success of its performance behavior can be seen as applying the permissive planning principle above: the planner is blamed and its bias adjusted even though the offending projection failures are due to inaccurate operator definitions.

In this view of planning, actions may have different *internal* and *external* effects. We will say that an action sequence *ISolves* a planning problem the goal holds in the projection of the initial state through the sequence. The sequence *ESolves* the problem if the goal is achieved in the real world by executing sequence s from the initial state.

In the planning literature, it is not uncommon to view a plan as a collection of constraints [Chapman87, Wilkins88]. We subscribe to this notion but carry it a bit further. For us, a *plan* for a planning problem is any set of constraints which *individuates* an action sequence such that the action sequence ISolves the planning problem. The accepted nonlinear view (e.g., [Chapman87, Sacerdoti75, Wilkins88]), is similar but does not require the individuation of an action sequence. A typical non–linear planner imposes constraints only until *all* action sequences consistent with the constraints are guaranteed to reach a goal state (i.e., each ISolves the planning problem). Stopping here allows a notion of minimal planning commitment. Remaining ambiguities are left for plan execution when they are resolved in the most propitious manner available in the execution environment. Our definition of a plan is more restrictive. We allow no ambiguities for later resolution. This requirement follows from our desire to adjust the planner's bias. We wish to incorporate the entire decision procedure resulting in an action sequence within the planner proper. Only then can the permissive adjustment procedure have access to the full bias reflected in the executable actions.

We use the informal term *partial plan* to refer to a set of constraints which is intended to solve a particular planning problem but does not yet individuate an action sequence.

A planner (including its bias) is *adequate* if 1) whenever a plan is produced that ISolves a problem it also ESolves the problem, and 2) whenever the planner fails to find a plan its competence set contains no plan whose action sequence ESolves the problem.

Finally, a *planning computation* is a finite sequence of decisions, D_i, $\{i=1,...n\}$. Each decision selects a constraint, c_i, to be added to the partial plan from a (possibly infinite) set of alternatives $\{a_{i,1}, a_{i,2}, a_{i,3}...\}$ entertained by the planner for that decision, so that $c_i \in \{a_{i,1}, a_{i,2}, a_{i,3}...\}$. The partial plan (which is initially the empty set of constraints) is augmented with the new decision's constraint, resulting in a (possibly) more restrictive constraint set. A planning computation is successful if, at its termination, there is exactly one distinct action sequence consistent with the set of constraints and that action sequence ISolves the planning problem.

Planning, in this framework, is repeatedly entertaining alternative constraint sets and for each, selecting one constraint to be imposed on the partial plan. This is not to say that every planner must *explicitly* represent the alternative constraint sets. But every planner's behavior can be construed in this way. From this perspective, a planner's competence is determined by the sets of alternatives that the planner can entertain. A (possibly empty) subset of alternatives from each constraint set supports successful plan completion. The planner's competence is precisely the set of all possible successful plans given the entertained alternatives. A planner's performance, on the other hand, is determined by its particular choice at each point from among the subset of alternatives which support a successful computation continuation.

The Permissive Planning Algorithm

A planning bias is any strategy for designating a particular element from among sets of valid continuation alternatives. Permissive adjustment of a planner is an empirically–driven search through the space of possible biases. Searching for an alternative bias is evoked whenever inadequate real–world planning behavior is observed.

In practice, the bias space is extremely large and the permissive planning search must be strongly guided by domain knowledge. If such domain knowledge is unavailable, the algorithm continues to have its formal properties. However, we believe that the practical ease with which suitable domain knowledge can be formulated will largely govern when the permissive planning approach will be useful. The permissive planning algorithm:

1. Initialize Candidate_Bias_Set to the space of all biases.
2. Using domain knowledge select an element from the Candidate_Bias_Set, call it Current_Bias.
3. Solve planning problems using Current_Bias. If an executed plan fails to achieve its goal go to 4.
4. Delete all biases from Candidate_Bias_Set that are provably inadequate using domain knowledge (including at least Current_Bias).
5. If Candidate_Bias_Set is not empty, Go To 2.
6. FAIL, no adequate bias exists.

As will become clear in the example, domain knowledge, in the form of qualitative information relating operators' effects to their arguments, can substantially increase the efficiency of permissive planning by guiding the selection of a promising bias in step 2 and increasing the number of untried but guaranteed inadequate biases rejected in step 4.

It can be easily proven when the algorithm terminates an adequate planner has been produced. Further, it can be proven that the algorithm will terminate so long as the bias space possesses some modest properties. On advice from an anonymous reviewer these proofs have been deleted in favor of more substantial discussions of other issues.

An Example of Permissive Planning

Suppose we have a two–dimensional gantry–type robot arm whose task is to move past an obstacle to position itself above a target (see Figure 4). The operators are **LEFT**, **RIGHT**,

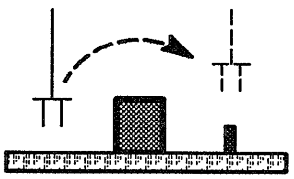
Figure 4: Goal to Move Past an Obstacle

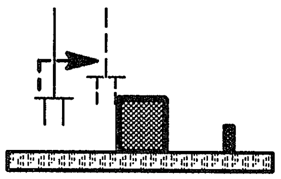
Figure 5: Collision Observed During Execution

OPEN, **CLOSE**, **UP**, and **DOWN**, which each take an amount as an argument. The obstacle is 2.5 units high, the tip of the hand is 1.2 units above the table, and the front face of the target is

4.201 units from the left finger. The planner's native bias results in a plan individuating the action sequence: UP(1.4) RIGHT(4.2). Figure 6 shows the alternatives entertained. Shaded alternatives indicate the competence selections given the constraints already adopted. The performance selection for each decision is outlined in a solid white box.

The first decision selects UP. The second decision selects a value for the parameter to UP. Any value greater than 1.3 up to the ceiling is possible according to the system's internal world. The value 1.4 is selected. Finally, RIGHT is selected with an argument of 4.2.

During execution a collision is detected. Permissive processing is invoked which explains the most likely collision in qualitative terms. The problem deemed most likely is the one requiring the least distortion of the internal projection. In this case, the most likely collision is with the obstacle block; the height of the gripper fingers is judged too low for the height of the obstacle as shown in Figure 5. The planning decisions are examined in turn for alternative competence set elements which can qualitatively reduce the diagnosed error. In this case it amounts to looking for alternatives that, according to the operators' qualitative descriptions, increase the distance between the finger tip and the top of the obstacle. Decision 2 is a candidate for adjustment. Higher values for decision 2 do not influence the projected goal achievement in the internal world but appear qualitatively to improve avoidance of the observed collision. The resulting plan is generalized in standard EBL fashion [DeJong86] resulting in a new schema which might be called REACH–OVER–OBSTACLE. It embodies specialized bias knowledge that, in the context of this schema, the highest possible value consistent with the internal world model should be selected as the parameter for the UP action. The new performance choice is illustrated in Figure 6 by a dashed white box.

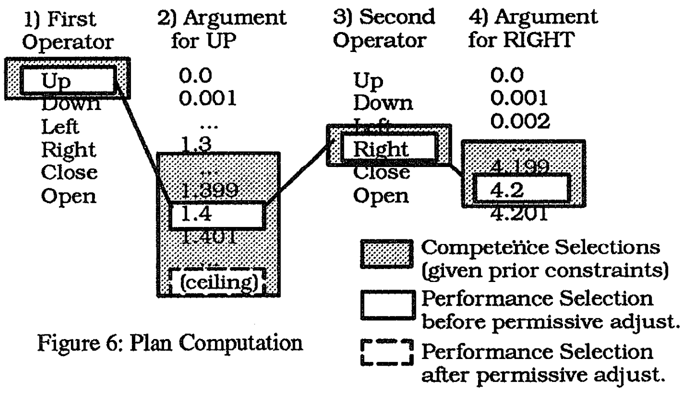

Figure 6: Plan Computation

From now on, the robot will retreat to the ceiling when reaching over an obstacle. If other failures are encountered, permissive planning would once again be invoked resulting in additional or alternative refinements. If no further refinement can be constructed, the schema forces a hard planning failure; none of the elements of the systems performance set is empirically adequate. Although the internal model supports solutions, this class of planning problems cannot be reliably solved in the external world. Any adequate planner must fail to offer a plan for such a class of planning problems.

Bias Space

What constitutes a bias space and how does one go about selecting a particular bias? These are important practical questions. If there is no convenient way to construct a searchable bias space, then permissive planning is of little consequence. The required *theoretical* properties are modest and do not significantly restrict what can and cannot serve as a bias space.

In fact it is quite easy to construct an effectively searchable bias space. We employ a method for the example above and for the GRASPER system based on problem–solving schemata (generalized macro–operators). Each schema represents a parameterized solution to a class of planning problems (like REACH–OVER–OBSTACLE). When the planner is given a new planning problem, the schema library is examined first. If no schema is relevant, a standard searching planner is applied to the problem. If a schema is found, the schema specifies how the problem is to be dealt with, and the native searching planner is not invoked. Thus, the schema library acts as a variable sieve, intercepting some planning problems while letting the native planner deal with the rest.

One practical difficulty with this method of bias adjustment is the utility problem [Minton88]. However, this is a separate issue from planner adequacy. Furthermore, recent research [Gratch92, Greiner92] has shown effective methods for attacking the utility problem that are consistent with this view of permissive planning.

Empirical Evidence

We have implemented a permissive planner, called GRASPER, and tested its planning on two domains using a real–world robot manipulator. Here we summarize the results of two experiments. Readers are referred to [Bennett93] for details of the system and the experimental domains.

Experiment 1. The task is to grasp and lift designated objects from the table with the gripper even though the gripper's actual movements are only imprecisely captured by the system's operator knowledge. Objects are known only by their silhouette as sensed by an over–head television camera. This experiment consists of an experimental and a control condition. Twelve plastic pieces of a children's puzzle were each assigned a random position and orientation within the robot's working envelope. Pieces were successively positioned as assigned on the table. For each, the robot performed a single grasp attempt. In the experimental condition permissive planning was employed; in the control condition permissive planning was turned off. The results are summarized in Figure 7A. In the control condition only two of the twelve attempted grasps succeeded. In the experimental condition ten of the twelve attempts succeeded. Failures due to three dimensional motion of the target, which cannot be correctly sensed by the robot, were excluded. One bias adjustment was sufficient to preclude recurrences of each of the two observed types of grasping failure. The two bias adjustments can be interpreted as 1) preferring to open the gripper as wide as possible prior to approaching the target, and 2) selecting opposing sides for grasp points that are maximally parallel. Other permissive adjustments that have been exhibited by the system in this domain include closing the gripper more tightly than is deemed necessary and selecting grasp points as close as possible to the target's center of geometry.

Experiment 2. Details of this task domain are borrowed from Christiansen [Christiansen90]. It is a laboratory approximation to orienting parts for manufacturing: a tray is tilted to orient a rectangular object free to slide between the tray's sides.. Christiansen employed the domain to investigate stochastic planning to which we compare permissive planning. The tray is divided into nine identical regions. The task is to achieve a desired orientation (either vertical or horizontal) of the rectangular object in a specified region.

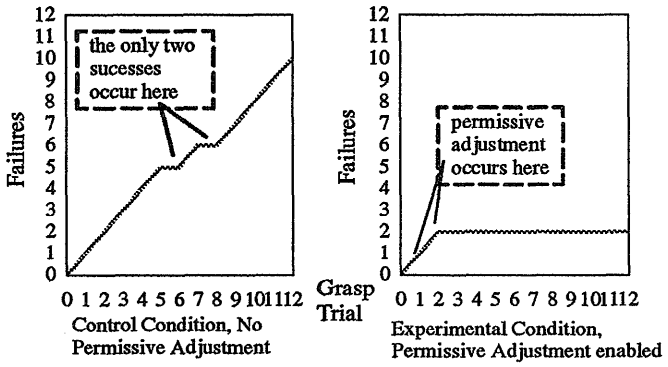

Figure 7: Effect of Permissive Adjustment on Grasp Success

We compared 1–step permissive plans to 1– and 3–step optimal stochastic plans. The optimal stochastic plans were generated using the technique described in [Christiansen90]. In the experiment, a sequence of 52 block orientation problems was repeatedly given to the permissive planner 20 times (1040 planning problems in all). Figure 8 shows the improvement

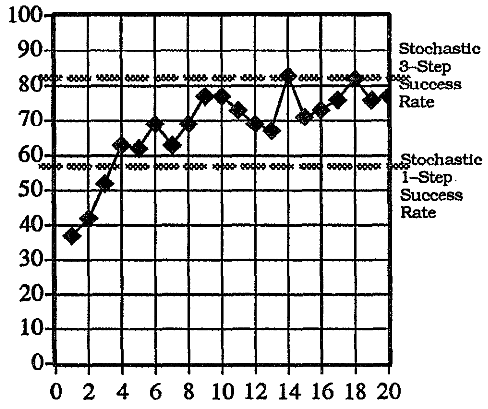

Figure 8: Average Success Rates over 20 Repetitions for 1–Step Permissive vs. 1– and 3–Step Stochastic Plans

in average success rate in the course of the 20 repetitions. Each data point is the average success over the 52 problems. Success rate increased from about 40% to approximately 80%. The final 1–step permissive performance approaches the 3–step stochastic performance, but requires fewer training examples.

Related Work and Conclusions

In this paper we have described our efforts toward formalizing permissive planning. Prior descriptions [Bennett91] have discussed our motivations and how these motivations have guided the implementation of experimental systems. We feel that we now understand the reasons behind the success of our implementations sufficiently to offer a beginning formal account of why our experimental systems work.

Our notion of bias was inspired by the use of the same term in machine learning [Utgoff86]. However, the bias referred to in permissive planning is quite separate from the bias of its underlying machine learning system. Likewise, our competence/performance distinction for planners is borrowed from a similar but fundamentally different notion in linguistics [Chomsky65].

In the current research literature the methods most similar to permissive planning trace their roots to dynamic programming [Bellman57] variously called reinforcement learning, temporal difference methods, and Q learning [92]. Like permissive planning the motivation is to improve real–world goal achievement. Some approaches utilize prior operator knowledge. However, like dynamic programming, the goal specification is folded into the acquired knowledge; the improved system is not applicable to other goals without redoing the machine learning. Furthermore, like stochastic planning, (but unlike permissive planning) a coarse discretization of continuous and fine-grained domain attributes is required.

One interesting consequence of permissive planning concerns minimal commitment planning. Permissive planning rejects this apparently attractive and generally accepted planner design goal. The desire to make the fewest necessary planning commitments is motivated by theoretical elegance, planning efficiency, and discrepancies with the real world. However, it denies access to a significant portion of the planning bias.

The primary significance of this work, we believe, lies in combining the internal constraints of a planner's *a priori* domain axiomatization (i.e., its definition of operators) with the external constraints obtained from examining real-world outcomes of action sequences. The machine learning formalism for combining these internal and external world models is problem-independent and domain-independent, although some amount of domain training is introduced. The approach offers real-world robustness previously associated only with reactive interleaving of sensing and action decisions.

The examples described here are modest. For each failure that initiates permissive planning, one could, after the fact, recraft the operator definitions. But this misses the point. After all of the axiom tuning a human implementor can endure, inaccuracies will still exist in a planner's world knowledge. As long as there are multiple ways of solving a problem, permissive planning can be employed to preferentially generate plans that work in the external world.

References

[Bellman57] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, 1957.

[Bennett91] S. Bennett and G. F. DeJong, "Comparing Stochastic Planning to the Acquisition of Increasingly Permissive Plans for Complex Uncertain Domains," *Proceedings of the Eighth International Workshop on Machine Learning*, June, 1991, pp. 586–590.

[Bennett93] S. W. Bennett, "Permissive Planning: A Machine Learning Approach to Planning in Complex Real-World Domains," Ph.D. Thesis, Electrical and Computer Engineering Department, University of Illinois, Urbana, IL, January, 1993.

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

[Chomsky65] N. Chomsky, in *Aspects of the Theory of Syntax*, MIT Press, Cambridge, 1965.

[Christiansen90] A. D. Christiansen and K. Y. Goldberg, "Robotic Manipulation Planning with Stochastic Actions," *Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA, November 1990, pp. 3–8.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation-Based Learning: An Alternative View," *Machine Learning 1*, 2 (1986), pp. 145–176.

[Gratch92] J. Gratch and G. DeJong, "COMPOSER: A Probabilistic Solution to the Utility Problem in Speed-up Learning," *Tenth National Conference on Artificial Intelligence*, San Jose, CA, 1992.

[Greiner92] R. Greiner and I. Jurisica, "A Statistical Approach to Solving the EBL Utility Problem," *Proceedings of the National Conference on Artificial Intelligence*, San Jose, CA, July 1992, pp. 241–248.

[McCarthy80] J. McCarthy, "Circumscription – A Form of Non-Monotonic Reasoning," *Artificial Intelligence 13*, 1 (1980), pp. 27–39.

[Minton88] S. Minton, in *Learning Search Control Knowledge: An Explanation-Based Approach*, Kluwer Academic Publishers, Norwell, MA, 1988.

[Sacerdoti75] E. D. Sacerdoti, "The Nonlinear Nature of Plans," *Proceedings of the Fourth International Joint Conference on Artificial Intelligence*, Tiblisi, Georgia, U.S.S.R., 1975, pp. 206–214.

[Sutton92] *Machine Learning (Special Issue on Reinforcement Learning) 8*, 3/4, R. Sutton (ed.), 1992

[Utgoff86] P. E. Utgoff, "Shift of Bias for Inductive Concept Learning," in *Machine Learning: An Artificial Intelligence Approach, Vol. II*, R. S. Michalski, J. G. Carbonell and T. M. Mitchell (ed.), MORGAN, 1986, pp. 107–148.

[Wilkins88] D. E. Wilkins, *Practical Planning: Extending the Classical Artificial Intelligence Planning Paradigm*, Morgan Kaufman, San Mateo, CA, 1988.