ELSEVIER

Artificial Intelligence 89 (1997) 173–217

Artificial Intelligence

Permissive planning: extending classical planning to uncertain task domains

Gerald F. DeJong [a,b,*], Scott W. Bennett [c]

[a] *Department of Computer Science, University of Illinois at Urbana-Champaign, 405 North Matthews Ave., Urbana, IL 61801, USA*

[b] *Beckman Institute, University of Illinois at Urbana-Champaign, 405 North Matthews Ave., Urbana, IL 61801, USA*

[c] *SRA Corporation, 2000 15th St. North, Arlington, VA, USA*

Received May 1995; revised July 1996

Abstract

Uncertainty, inherent in most real-world domains, can cause failure of apparently sound classical plans. On the other hand, reasoning with representations that explicitly reflect uncertainty can engender significant, even prohibitive, additional computational costs. This paper contributes a novel approach to planning in uncertain domains. The approach is an extension of classical planning. Machine learning is employed to adjust *planner bias* in response to execution failures. Thus, the classical planner is conditioned towards producing plans that tend to work when executed in the world.

The planner's representations are simple and crisp; uncertainty is represented and reasoned about only during learning. The user-supplied domain theory is left intact. The operator definitions and the planner's projection ability remain as the domain expert intended them. Some structuring of the planner's *bias space* is required. But with suitable structuring the approach scales well. The learning converges using no more than a polynomial number of examples. The system then probabilistically guarantees that either the plans produced will achieve their goal when executed or that adequate planning is not possible with the domain theory provided. An implemented robotic system is described.

Keywords: Planning; Learning; Uncertainty; Machine learning; Explanation-based learning; Planning bias

* Corresponding author. E-mail: dejong@cs.uiuc.edu.

PII S0004-3702(96)00031-8

1. Introduction

We seek a formal bridge between classical planning and real-world goal achievement. The chasm to be spanned is wide and deep. Classical planners are concerned with behavior in micro-worlds, hypothetical worlds that can be flawlessly formalized. Classical plans are often thwarted by the real world; a sequence of actions may be proven to achieve a micro-world goal only to produce an unacceptable real-world state when executed. We can trust that the micro-world behavior is indicative of real-world performance only if the planner's representations perfectly capture all relevant details of the real world. This is seldom the case. Often, the planner's domain representations only approximate their real-world changes. Combinations of apparently innocuous flaws may conspire to invalidate a plan in the real world.

One cause of this difficulty is the qualification problem [46], which is succinctly stated by Genesereth and Nilsson [27]:

> Most universally quantified statements will have to include an infinite number of qualifications if they are to be interpreted as accurate statements about the world.

Classical operator definitions are universally quantified first order expressions and can, therefore, only imperfectly capture most real-world changes. Representations of the initial world state may also be imperfect. Thus, it is inescapable that a planner may encounter situations in which its conclusions will contradict the real world.

Permissive planning is a learning approach to planning under uncertainty. To illustrate it, consider the simple problem of making breakfast in the morning. The initial state of our kitchen includes many standard cooking ingredients: milk, flour, eggs, butter, containers of hot and cold cereal, syrup, a selection of fresh and dried fruit, etc. The goal is to reduce our morning hunger. Operators include pouring, measuring, mixing, heating, and so on. There are many possible solutions our planner could in principle produce for us. It could make pancakes, crepes, Belgian waffles, hot oatmeal, cold raisin bran, sausage and eggs, gravy and biscuits, eggs Benedict, etc. We call the set of all solutions that could in principle be produced the *competence set* of the planner for the problem. In fact, of course, the planner will not generate all solutions; it will construct just one. This designated element of the competence set we call the *performance item* of the planner for the problem. We call the preference for a particular performance item from the competence set the planner's *bias*. As we shall see, bias is an inescapable facet of planning, and its adjustment is at the heart of permissive planning.

Suppose that the planner's performance item for the breakfast problem is to prepare eggs Benedict. Goal achievement is, of course, entailed in the planner's micro-world, but the result in the real world may be quite unsatisfactory. A perfectly reasonable looking plan might result in lumpy and separated hollandaise sauce reducing the eggs Benedict dish to a disgusting tasteless mess. The problem is that one or more of the system's internal representations are not sufficiently faithful to reality for the plan to succeed. Perhaps some initial state representations are subtly wrong—the eggs are warmer than represented or the butter is too low a grade, or perhaps the operators are incorrect, incomplete, or based upon false suppositions—the blender may not beat as fast as believed or the beating may result in an unmodeled heating of the mixture. Most likely

the blame cannot be laid at a single representation. Usually multiple discrepancies, each of which is slight, conspire together to produce the failure.

Permissive planning, when confronted with an unacceptable failure rate, adjusts the planner bias. This shifts the solutions to different performance items. For breakfast a different eggs Benedict recipe might be attempted or perhaps a different dish entirely is prepared. There are nearly always planning alternatives within the micro-world framework; few problems admit to only a single solution. It is foolish to persist making foul eggs Benedict ignoring all the attractive other breakfast options.

Why might preserving the micro-world be desirable? The system's operator representations are typically provided by a human domain expert. Presumably, the expert did his best at capturing his own coherent although possibly idiosyncratic conceptualization of world change. The qualification problem assures us that imperfections are unavoidable. Stumbling upon one is to be expected and is hardly evidence that the system can improve upon the work of the domain expert. Furthermore, tinkering with human-supplied knowledge may corrupt its faithfulness to the expert's conceptualization and introduce latent inconsistencies of a far worse nature. On the other hand, the planner's intial bias is likely to be quite arbitrary. It is supplied by the planner implementor and reflects little or no understanding of the domain. And yet each bias has grave implications for the planner's success in particular domains. It may be more prudent to alter the bias over the domain representations.

A major issue raised by permissive planning is that bias is a characteristic of the planner as a whole. Altering the bias may shift the performance element produced for many problems at once. It may be that the solutions generated with the old bias were quite acceptable but the plans now produced lead to real-world failures. Permissive planning must provide some guarantee of global acceptable behavior. Much of our research concerns this point.

Let us briefly examine how permissive planning relates to other approaches to uncertainty. These fall into three general categories. In the first, reasoning systems are provided with some explicit account of uncertainty and how it is propagated [18]. This general approach includes Bayesian reasoning [15,66], planning with error balls [25], fuzzy logic [87], and decision theoretic approaches to planning [19,30,31,74]. Systems in this general area are given representations of the world that are more sophisticated than required by conventional classical planning. These representations include some characterization of possible discrepancies that are likely to be encountered. The planner takes such variabilities into account in deciding how best to behave in the world. We can view each representation as specifying a potentially large disjunction of facts. Thus, each representation is less likely to be violated in the real world. Importantly, there is cost in reasoning with augmented representations which can be quite high [73].

In the second approach, machine learning is employed to refine the offending representations. As in classical planning, the system's action and object representations are simple and precise but possibly incorrect. As the system interacts with the world, it gains information through its observations. The domain theory is suitably modified to bring it into line with the observed world behavior. This can be seen as an induction problem: The hypothesis space consists of some (usually large) set of well-formed alterations of the initial domain theory. Examples are the observed real-world behaviors. The task is

to locate the hypothesis that best fits the examples. This may involve general theory refinement and discovery (e.g. [61,64,65,70]) or, more conservatively, operator learning and refinement (e.g. [28,35,83]). The reasearch connecting belief revision and update with theories of action [11,23,40] is also relevant.

The third approach relies on reactivity [1,12,49,60,76]. While conventional plan execution blindly applies the sequence of the planner-specified operators, a reactive system achieves robustness by extending execution to include some degree of world monitoring and decision making. Reliance on action projection is reduced or eliminated; the world is not assumed to behave precisely as anticipated at planning time. The "plan" bears little resemblance to a classical plan. Typically, it is composed of a large number of what-if decisions that specify actions for the world states likely to be encountered. It is perhaps better termed a *policy* for acting rather than a plan. Goal achievement emerges from the interactions of this policy with the dynamics of the world. While not essential, machine learning can be incorporated for the purpose of automating plan/policy construction [7,16,29,36,54,80].

Each of these three general approaches suggests a different class of solutions to the breakfast problem. In the first one, representations are enhanced to encode uncertainty information. Solutions are produced which are less sensitive (perhaps even minimally sensitive) to these uncertainties. For eggs Benedict the system might have selected a less exacting but less tasty initial hollandaise sauce recipe from the *Anyone can Cook* cookbook. Or perhaps it would make a fool-proof breakfast of cold cereal with milk. The second approach employs machine learning to revise the faulty world knowledge. By examining the lumpy sauce or perhaps through additional eggs Benedict experiences, the system concludes that the "whip" speed of this blender is significantly slower than previously believed. Or perhaps it determines that the brand of butter used is insufficiently homogenized. In any case, its internal projection ability is brought into line with observed reality. With the refinement it may be able to produce a more satisfactory eggs Benedict. Third, reactivity monitors the world after each operator execution. The next action is chosen to bring the observed world closer to a goal state. After pouring the melted butter the system observes undesirable lumps in the sauce. It may decide to add oil or to heat the mixture or to go next door to borrow the neighbor's *BlenzAll* super blender. Crucially, the system does not anticipate and plan around failures. Rather it attempts to recover from them as they are detected.

Permissive planning is cast from a somewhat different mold. It is unlike the first in that the representations of world objects and actions are crisp, simple, and precise encoding no information about expected uncertainty. Unlike the second approach, the domain knowledge—the system's represented beliefs about the world—are never altered. Finally, it is unlike the third in that it preserves the notion of a classical plan. Projection plays an important role in permissive planning and there are no execution-time action decisions.

Permissive planning is not the final word in planning under uncertainty. The ultimate treatment of uncertainty will most likely combine aspects of all of these approaches. But permissive planning is a promising and as yet little explored direction.

The permissive approach can be briefly summarized as follows: A planner can be characterized as having parameters that influence its search for a solution. Each combi-

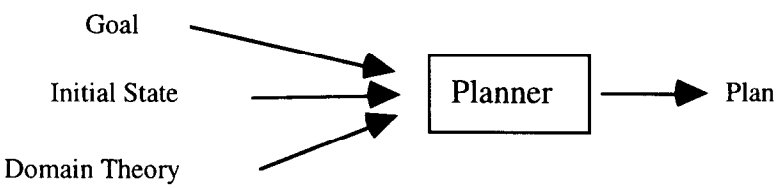


Fig. 1. Classical planning.

nation of settings for these parameters causes some bias to be exhibited by the planner. The collection of all biases forms the planner's bias space. Machine learning is employed to search this space for a bias element that yields acceptable planner performance for the kinds of problems the planner finds itself solving. Permissive planning is so named because the planner is adjusted to be permissive of less than perfect domain knowledge.

Many tradeoffs and alternatives are possible within this general approach. We define *ideal* permissive planning, one of the least restrictive and most naively appealing versions, and prove that it cannot be computed. We then investigate one alternative embodying a particular set of restrictions which yields tractable permissive planning. Finally we illustrate the permissive planning approach applied to a real-world planner.

2. A model of classical planning

In Sections 5 and 7 we build upon a particular characterization of classical planning. Here we present that characterization. For our purposes, a classical planner is a partially recursive algorithm that (a) inputs operator definitions, an initial state, and a goal description and, (b) outputs a set of constraints with the property that any sequence of ground operators consistent with the constraint set achieves the goal from the initial state in the micro-world defined by the operators. See Fig. 1. Note that there is no requirement that the operators be STRIPS-like [59]. They may have conditional effect, internal quantification, etc. (see [67,85]). We assume that the real-world counterpart is deterministic but is only approximated by the micro-world.

All classical planners (be they linear, nonlinear, hierarchical, etc.) can be viewed as conducting a search through a tree of constraint sets. This view is close to the conventional one in which the search is through a space of plans [14,38,47,51,68,86]. We interpret a constraint set as denoting the set of all action sequences (each a temporally ordered set of ground operators) that are consistent with the constraints of the set. A constraint set itself is inconsistent iff it denotes no action sequence. Each abstract planning step (each significant decision of the planner) adds a nontrivial constraint to the collection to generate a novel set of constraints. Each nontrivial constraint eliminates one or more action sequences that were consistent with the parent set. This defines a tree of constraint sets (see Fig. 2).

While classical planning can be abstractly characterized in this way, we do not mean that every classical planning algorithm must be overtly implemented as a search through

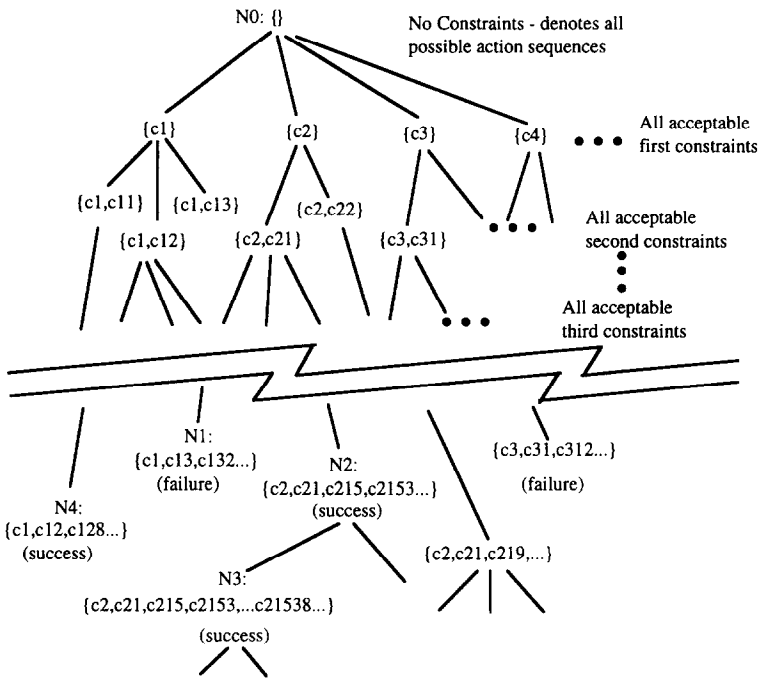


Fig. 2. The constraint tree for a planning problem.

a constraint space, or even that the constraints need be explicitly represented. However, the *behavior* of any classical planner can be accurately captured as such a search.

In this view, the only difference between planners lies in their vocabulary of constraints and their method of exploring a planning problem's tree of possible constraints (selecting which node of the tree to expand next). Although such difference results in quite different planners, it will be useful to ignore such details. In doing so, we are assured that our statements are general—applying to all classical planners.

Once a vocabulary of constraints is adopted, a constraint tree such as that in Fig. 2 can be associated with each planning problem. The root node (e.g., $N0$) is the empty set of constraints; it denotes all sequences of actions. The descendants of each node are the consistent constraint sets that can be formed by adding a single nontrivial constraint to the parent's constraint set. Thus, each child node denotes strictly fewer action sequences than does its parent. A leaf node is one to which no consistent nontrivial constraint can be added.

A node is a "success" node if the projection of the initial state through each and every action sequence denoted by its constraint set results in a goal state. In other words, the goal must be entailed by the initial state, the system's world knowledge, and the node's constraints. $N2$ is a success node. Success nodes may have descendants that are formed in the same manner as other children nodes. Every descendent of a success node (e.g., $N3$) is also a success node. This is necessarily the case: the child node's additional constraint can only remove action sequences, and all the action sequences denoted by

the parent already solve the planning problem. A node is a "failure" node if it has no success descendants. An example of a failure node is $N1$. Of course, many nodes, including $N0$, are neither success nor failure nodes. We call the set of all success nodes the *competence set* for the planning problem.

Some planners may appear not to fit this characterization. Some apparently search beyond the boundaries of the constraint tree. However, we need not interpret such computation as extending the search tree: It can be computationally difficult to determine redundancy and inconsistency of constraints. An implemented planner may unknowingly add redundant constraints, simply not appreciating that they have no effect. Similarly, it may happily accrete additional constraints to a constraint set unknown to be already inconsistent until the inconsistency is more readily apparent. These two behaviors can be viewed as implementation details outside our formal treatment. From a computational point of view, all of the planner's behavior relevant to constructing its plan is confined to a constraint search tree such as that shown in Fig. 2.

3. Real-world planning and real-world adequacy

Can we capture some notion of real-world planning adequacy within this general planning framework? To do so we must first define precisely what it means for a planner to be "adequate". Informally, we might say that a planning system is *adequate* if its user is satisfied with the real-world performance.

This informal definition sheds no light on how to attain planner adequacy but an important point becomes clear: the adequacy of a planner depends on more than its internal workings. Adequacy is measured against those problems that concern its user. A user who cares about stacking blocks in a simulated robot world might find a particular planner completely adequate. On the other hand, someone who schedules nation-wide rail freight deliveries might find the same planner quite inadequate.

Intuitively we also prefer that a planner appreciate its own limitations. The best state of affairs from the point of view of a user is that when given a problem, the planner produces a solution that works when executed. The worst is that the planner produces a solution that actually fails in the world. The planner deciding not to attempt the problem is somewhere in between, for then the planner does not misrepresent its output although, of course, neither is the problem solved.

We want a notion of planner adequacy consistent with the informal notions above. We begin with two different definitions of what it means to solve a planning problem. For the moment we assume that a *plan* denotes a particular sequence of actions that the planning system claims will solve the planning problem that gave rise to it. Of course, minimal commitment and nonlinear planners do not, in general, designate a single ground action sequence [14, 39, 45, 68, 75, 81, 86]. We will have more to say about this in the conclusion section. For the moment a plan is an individuated action sequence.

A plan produced by a planning system *ISolves* (solves according to its own internal model of the world) a planning problem if the initial state projected through the plan satisfies the problem's goal. We say that the plan *ESolves* (solves according to the

external world) a planning problem if the real-world state that results from executing the plan in the problem's initial state satisfies the problem's goal. As shorthand we will say a *planning system* ISolves (or ESolves) a problem if the performance item produced by the system ISolves (or ESolves) the problem.

We assume the following operating protocol: A planning system is called upon to solve a succession of problems given to it one at a time. This succession is unlimited—there is always another problem available on which to test the planner's behavior. Further, we assume that the problems are drawn according to a fixed but unknown distribution over a universe of well-formed problems. After some fixed amount of resources are expended on deliberation, the real-world planning system must either offer a solution to the planning problem (by outputting a plan) or choose not to offer a solution (by outputting the special symbol Λ). Planning in this framework is decidable but only by the subterfuge of forcing an output of Λ after a resource bound is reached; in the real world systems cannot deliberate forever.

We can now more precisely define the adequacy of a planning system:

Definition 1. The *adequacy* of a planning system P over a universe of problems U randomly sampled according to a distribution D is:

$$A_{UD}(P) = \begin{cases} \sum_{i \in U} \Pr_D(i) \cdot \alpha, & \text{if } P \text{ ISolves } i \text{ and } P \text{ ESolves } i, \\ \sum_{i \in U} \Pr_D(i) \cdot \beta, & \text{if } P \text{ does not ISolve } i, \\ \sum_{i \in U} \Pr_D(i) \cdot \gamma, & \text{if } P \text{ ISolves } i \text{ but does not ESolve } i, \end{cases} \tag{1}$$

where α, β, and γ are constants satisfying $\gamma \leqslant \beta \leqslant 0 < \alpha$ and $\Pr_D(i)$ is the probability of occurrence of problem i according to distribution D.

This definition reflects a user's concern for some problems above others (via the distribution) and is sufficiently expressive to distinguish between a plan failing and the planner choosing not to offer a plan. Thus, it is sensitive to the pre-theoretic notions discussed above. Each problem in the universe of problems contributes to a planner's adequacy as follows: Producing a plan that works in the real world accrues a positive reward (α). A penalty (β), which may be zero, is assessed for not producing a plan. A possibly more severe penalty (γ) is assessed for producing a plan that does not work in the real world. Each problem contribution is weighted by the probability that the problem will be encountered according to the distribution. Other things being equal, adequacy is higher if offered plans actually work in the real world. Furthermore, producing a bad plan cannot be better (and may be worse) than producing no plan at all. Without loss of generality the definition could be simplified by setting α identically equal to 1. But this somewhat obscures its influence in the analysis. Other formalizations of adequacy are also possible but this one is simple, intuitive, and sufficient for our purposes.

4. Properties of planner adequacy

There are several interesting properties that follow from the definition of adequacy given above.

(1) The adequacy of any planning system is finite. In fact, it is bounded above by α and below by γ.
(2) No classical planner in the traditional AI sense can be assigned a meaningful adequacy rating.
(3) Some form of machine learning is required to insure adequacy for a domain-independent planner.

The first property follows directly from the convention that probabilities over a universe sum to 1. If all problems ever asked of the planning system are correctly solved it will have an adequacy of α. If they are all solved incorrectly, the adequacy rating will be γ. This makes direct comparisons between planners possible, even if the planners are operating in different domains and for different users.

The second property is philosophically more interesting. It hints at how AI planning has so successfully ignored real-world performance. In domain-independent planning, a planner is only one factor in the adequacy assessment. Properties of the domain, characteristics of the problem distribution, and preferences from the user all contribute to a meaningful adequacy evaluation. With this in mind, property (2) is quite intuitive: classical domain-independent planning can guarantee reasonable adequacy only when the domain theory perfectly captures the real world so that ESolve and ISolve become the same relationship.

The third property is related to the second and stems from the qualification problem. A planner's adequacy is, among other things, a characterization of how well the planner's operator representations characterize their corresponding real-world behaviors. Consider using prior knowledge (rather than machine learning) to characterize the operators' real-world behavior. Only a set of operator representations that include no discrepancy with the real world would be sufficient to support guaranteed adequate domain-independent planning. The qualification problem assures us that, in general, this is not possible. Thus, knowledge of operator deficiencies with respect to the real world cannot, in general, be *a priori*. A characterization of a planner's adequacy must be *a posteriori*, formed from observations of instances of real-world plan executions. Automatically characterizing observed instances is one definition of machine learning [48,53].

5. Bias and bias accommodation

Any reasonable classical planner necessarily embodies a strong planning bias. Why should this be the case? Consider the characteristics of an unbiased planner. By definition it must avoid expressing any preference for one solution over another. Furthermore, the lack of bias must hold for all problems and across all domains. There are only two unbiased behaviors possible. The first is to return all elements of the competence set at the same time. The second is to return Λ (failure) regardless of whether a solution exists. It is clear that both options are in most cases not reasonable. Resources

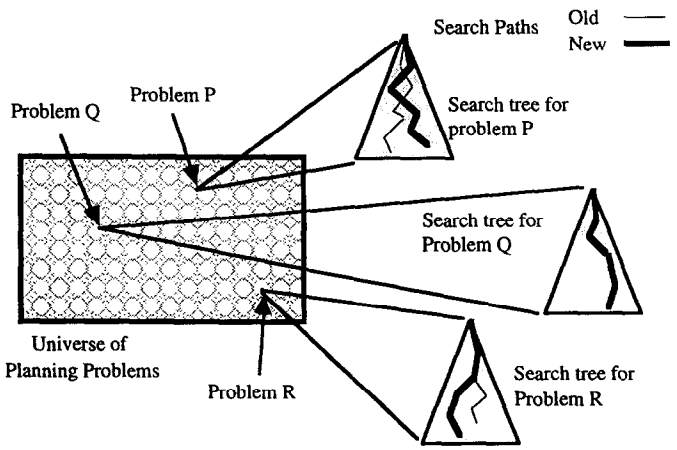


Fig. 3. The effect of altering a planner's bias.

should not be wasted finding solutions that will not be executed. Nor should a planner capriciously fail to solve problems when at least one solution is easily within its grasp.

Possessing a bias is not a disadvantage. Quite to the contrary: by its judicious use, one may prefer a solution that minimizes planning time, minimizes resources consumed during execution, or in the case of permissive planning, improves the likelihood of successful execution. However, it should be clear that bias is an unavoidable facet of classical planning even without taking real-world adequacy into account.

Consider the effect of altering the bias of a planner. We will view bias change abstractly as the substitution of one preference function for another in the search across all problems' constraint trees. We defer until later a mechanistic account of how such an alteration may be realized. The effect of such a substitution is illustrated in Fig. 3. Each element in the space labeled "Universe of Planning Problems" represents a well-formed problem that might be given to a classical planner. For each problem, the planner, with its original bias, conducts some search through the problem's constraint tree. The original search path is represented as a light line in the triangle that represents the problem's constraint tree. Using the new bias, most searches (like the one for problem P) will be quite different. Some (like that of problem R) may be only partially changed. Others (like the tree for problem Q) may remain unchanged. The solvability of the problem may change as well. Since classical planners are sound but not necessarily complete, a problem that is successfully solved by one preference function may be unsolvable under another.

The important point for us is that by changing the preference function, the planner's bias is altered. By altering the planner's bias we may exert some control over which solution is found. This shift can be quite dramatic. The change of a single search choice can lead to a previously untouched portion of the constraint tree.

Now consider a space of possible biases. The notion, though abstract, is well formed: the behavior of each planner can be described as the application of a particular preference

function to a constraint tree. Alternative preference functions result in different classical planners. A set of such preference functions forms a space of potential biases for the planner.

Given a bias space, a planner, and some utility function defined over the universe of possible problems, it is entirely possible (indeed quite likely) that some planners will be of higher utility than others. By some form of systematic search through the bias space we might hope to find a planner whose utility is optimal, or nearly optimal, or at least higher than the utility of the original planner.

We call such a search "accommodation" of the planner's bias. Note that such a search changes none of the human-supplied information. The original operator set and constraint vocabulary, supplied by the planner's implementor, are not altered. *Permissive planning* is bias accommodation using real-world planning adequacy as the measure of utility.

Importantly, the planning system does not become better at anticipating the effects of its actions. Its internal representations remain as they were, and its projection ability is not altered. Rather permissive planning results in the generation of action sequences whose real-world behavior tends to more closely follow the sequences' projected behaviors. Execution effects come into line with projected effects rather than the other way around.

6. Ideal permissive planning

There are many different ways to instantiate the above abstract characterization of permissive planning. Unfortunately, the straightforward, ideal realization of permissive planning is computationally untenable. In ideal permissive planning the bias space is unrestricted. There are no known *a priori* constraints upon the individual biases that make up the space and the bias space may be infinite:

Definition 2. *Ideal permissive planning* is a recursive procedure, that given:

- a planner,
- a bias space,
- adequacy values α, β, γ,
- a threshold of required adequacy, T_a, between α and β,
- a source of planning problems,

produces a new planner through bias accommodation that has the following properties:

- if the bias space contains any bias of adequacy $\geqslant T_a$, the bias with the highest adequacy is adopted,
- if the space contains no bias of adequacy $\geqslant T_a$, the new planner refuses to offer solutions to any further problem.

The recursiveness of the procedure requires that the algorithm always halt. Therefore, only a finite number of example problems can be examined before either selecting a bias or guaranteeing that no adequate bias exists. Under these conditions are incompatible with permissive planning:

Theorem 3. *Ideal permissive planning cannot be realized.*

This theorem follows directly from a lemma to which we now turn. The lemma concerns *simplified ideal permissive planning*, a greatly restricted version of the ideal formulation. In simplified ideal permissive planning, the bias space contains *exactly two* biases. After attempting to solve a finite number of example problems, the better of the two biases must be selected. Problems for which the planner offers no solution, are counted as failures ($\gamma = \beta$).

Lemma 4. *The simplified formulation of ideal permissive planning cannot be realized.*

The proofs are given in the appendix. The proofs of these results, like many proofs, are themselves somewhat unenlightening. Before proceeding to the positive results it is useful to develop an intuition for why in particular Lemma 4 holds. We can then more easily appreciate how the upcoming theorems avoid the computational pitfalls.

A successful permissive planning algorithm for the simplified formulation must begin by solving a first sample problem using one of the two possible biases. The solution constructed by the planner under the sampled bias is then executed in the real world. The outcome will either be a success (in the case that the goal was actually achieved) or a failure (if no solution was proposed, or if executing the plan in the world does not achieve the goal). If only we could know that one bias was always successful and the other was always unsuccessful, then one sample problem, regardless of its outcome, would be sufficient to choose the better bias once and for all: if the sample plan is successful, select the bias and quit; if it is unsuccessful, select the alternate bias and quit. But such knowledge is disallowed under condition (a) above. Even the better bias may often yield a failing plan. To be optimal, the bias need only do better, on average, than the alternative bias. This decision can be made to be arbitrarily difficult and thus no algorithm can in general guarantee a final correct decision.

What can we say about the description of the success/failure sequence due to a particular bias? With the bias held constant, the sequence of experienced successes and failures is entirely due to the sequence of world problems. Since these are, by hypothesis, drawn randomly (though, of course, according to some unknown fixed distribution), the outcome of sampling a particular bias can be viewed as a random variable. Since the outcome distribution is binomial (each observation of the bias's performance yields one of two possible outcomes: real-world success or real-world failure) the bias variables are said to be Bernoulli. A Bernoulli variable is characterized by a single probability: if the likelihood of one outcome (success) is ρ (with $0 \leqslant \rho \leqslant 1$) then the likelihood of the other (failure) is $1 - \rho$. Let B_1 be the random variable associated with the first bias and let its probability of a successful outcome be ρ_1. Similarly define B_2 and ρ_2 for the second bias. The parameters ρ_1 and ρ_2 are unknown but fixed; they reflect the underlying fixed distribution over the universe of problems. The adequacy of the first bias is $\rho_1\alpha + (1 - \rho_1)\gamma$ or $\rho_1(\alpha - \gamma) + \gamma$. Similarly the second bias's adequacy is $\rho_2(\alpha - \gamma) + \gamma$. Thus, and intuitively, the first bias is an optimal bias iff $\rho_1 \geqslant \rho_2$.

We now view this as a bandit problem. The discrete time, finite horizon, minimax bandit problem with two Bernoulli arms [9] can be conceptualized by analogy to a gambling machine. Suppose we are required to play N trials (hence discrete time with a finite horizon of N). At each trial we must pull one or the other arm of the two-arm bandit. After being pulled the arm randomly succeeds yielding a fixed reward, or fails yielding no reward (neither the amount of the reward nor the possible existence of a fixed penalty for pulling the arm affects the analysis). Thus, the arms are Bernoulli variables. The probability of payoff of each arm, though fixed, is unknown; additional knowledge of each arm's characteristics is limited to the observations of prior trials. We wish to discover a strategy (a decision procedure for selecting Arm1 or Arm2 at step i possibly making use of the results of steps 1 through $i-1$) that maximizes our expected wealth at the end of the N trials regardless of the payoff probabilities. Clearly, this is a problem of incomplete knowledge, for if we knew the payoff probabilities of both arms, then the solution, for any N, is simply always to pull the higher probability arm.

The *regret function* for a decision strategy applied to a bandit is the accumulated difference between the expected payoff (the average over repeatedly following of the strategy up to N) and the omniscient optimal payoff (playing the higher probability arm N times). By definition, no strategy can, on average, win against the omniscient optimal strategy, so the regret function must clearly be nonnegative for all realizable strategies. If some realizable strategy could always discover the better arm after some finite number of trials m (less than N) then we would expect the regret function to be flat from the mth through the Nth trial. Indeed for any N' larger than N, the regret function would accumulate no further deficit compared to the omniscient strategy. A successful permissive planning algorithm would select and continue to apply the better of the two biases after observing a finite number of samples. When considered as a bandit strategy, a permissive planning algorithm would result in a regret function that is *constant* as N grows beyond m. The established result from the bandit literature is that the regret function for the optimal realizable strategy cannot be flat. In fact it grows as $\sqrt{N}$ as N grows without bound [3].

For us, the significant feature is that the regret function cannot be flat. This follows from the requirement of global optimality. Any belief in the superiority of one arm becomes more certain with more confirming observations, but it cannot be guaranteed by any finite sample set. The average payoff probabilities between two bandit arms may be slight but with high variances, making the comparative evaluation arbitrarily difficult. If one commits to an arm to play for all the remaining opportunities, there will always be some chance it is the wrong arm. In the unbounded case, where horizon (the number of trials we care about) is infinite, the accumulated deficit (and therefore the regret function) can grow without bound. On the other hand, if we never select an arm to play once and for all, then we condemn ourselves to sample both arms unboundedly often (including the less desirable arm, whichever one it is). Again we incur an unboundedly large deficit. Thus, under these assumptions the regret function must continue to grow as the horizon recedes. Without an asymptotically flat regret function, no successful permissive planning algorithm can be realized. To be computationally viable, permissive planning must sidestep these difficulties.

7. With restrictions, permissive planning is tractable

Fortunately these negative results are not the last word on permissive planning. We now show that a particularly simple kind of permissive planning, that we call restricted permissive planning, is tractable. While we give only a single tractable method, there are two features that all tractable approaches must share: we cannot insist on global optimality and some structure must exist within the bias space.

Abandoning guaranteed optimality is a necessary step. We will content ourselves with some sufficient specified adequacy. Furthermore, we allow a small nonzero adequacy tolerance parameter ε that establishes an indifference region about the adequacy threshold. Our interpretation of this region is that if the adequacy of the best bias in the space falls within $\pm\varepsilon$ of the threshold we are indifferent to the outcome. The procedure may either adopt the bias as adequate or the planner may decide there is no adequate bias and choose not to solve further problems. Finally, restricted permissive planning need meet its goals only with a high probability. In particular we include a confidence parameter δ that specifies an acceptable probability that the permissive planning algorithm terminates with an inadequate bias which it misrepresents as adequate. More formally:

Definition 5. *Restricted permissive planning* is a recursive procedure, that is given:

- a planner,
- a bias space,
- adequacy values α, β, γ,
- a threshold of required adequacy, T_a, between α and β,
- a source of planning problems,
- two parameters, ε and δ both between 0 and 1,

produces a new planner through bias accommodation that with probability at least $1-\delta$ has the following properties:

- if the bias space contains any bias of adequacy $\geqslant T_a+\varepsilon$, some bias with adequacy of at least $T_a-\varepsilon$ will be adopted,
- if the space contains no bias of adequacy $\geqslant T_a-\varepsilon$, the new planner refuses to offer solutions to any further problem,
- if the best bias in the space falls within the indifference range of $T_a \pm \varepsilon$ then the planner may either adopt a bias of adequacy $\geqslant T_a-\varepsilon$, or refuse to offer solutions to further problems.

The ε/δ arrangement is borrowed from the PAC literature [82] and also earlier employed in [4]. From this point in the paper on, when we mention adequacy we will understand it to mean probabilistic attainment of an ε-adequacy threshold.

From (1) it is clear that the meaningful values of T_a are between α and β. For values $\leqslant \beta$, adequacy is no higher than the trivial planner that always returns Λ. On the other side, values $> \alpha$ are not realizable. It will occasionally be useful to consider a normalized utility parameter Θ in place of T_a. It takes on values over the interval $(0,1)$ as T_a varies between β and α.

Definition 6. The *normalized utility parameter*, Θ, is $(T_a-\beta)/(\alpha-\beta)$.

7.1. Schema-based planners

The lack of structure among bias space elements is another foe of tractability. Independence of bias elements means that evidence for or against the adequacy of one bias cannot count as evidence for or against another bias. Thus, under independence, any reasonable search of the bias space has an algorithmic complexity bounded below of linear growth in the size of the bias space. This is well within the accepted computational complexity bound for tractability (which is polynomial growth). However, the size of the bias space may itself be extremely large, even unbounded. Even with an efficient algorithm yielding a small constant, this may be unacceptable. To be more realistic, we will insist that the search complexity grow much slower than linearly with the size of the bias space.

Interdependence often complicates system analysis, but a bias space is a special kind of system. It captures whatever flexibility exists for bias adjustment. The assumption of independence here is overly general, and preserves a costly but nearly useless simplicity. With an advantageous interdependence, the observation of a planning outcome using one bias might be informative about many others. We, the implementors, are at liberty to design bias flexibility any way we like. We may hope to derive some advantage by carefully structuring our flexibility. Schema-based (or skeletal) planners [26] provide a particularly flexible bias structure.

In a schema planner, planning techniques are encoded as generalized solutions (called schemata). The system solves problems by retrieving an appropriate general pattern and instantiating it to fit the particular problem. To illustrate, consider the goal of getting to a distant city. The system might have three schemata relevant to this goal, `take-a-bus`, `take-a-commercial-airplane`, and `take-a-private-car`. These encode general knowledge about how to employ their respective methods of transportation. The system might have many additional schemata not relevant to the transportation goal (for making breakfast, for doing laundry, for removing snow from ones sidewalk, ans so on).

Building upon the characterization of classical planning given in Section 3, a schema can be viewed as a mechanism for supplying a set of constraints that depend in part on the problem's initial state and goal. The constraints are sufficient to individuate an action sequence intended to solve the problem. Thus, we can define a schema as follows:

Definition 7. A *schema*, SC, is a set of constant constraints C_{SC} on action sequences together with a function $F_{SC} : P \rightarrow C^* \cup \Lambda$ that maps a planning problem into either a set of additional constraints or the special symbol Λ.

The symbol Λ is interpreted as indicating that the schema's preconditions are not satisfied by the problem. If the preconditions are violated, no solution is offered for the problem. Otherwise we interpret the union of a schema's constant and functional constraints as embodying a solution guaranteed to ISolve the planning problem given to it.

7.1.1. Schema planner adequacy is by problem class

Notice that problems naturally cluster into problem classes. The potential goals, the domain theory, common initial state features, etc. involved in getting to a distant city

are different than those involved in having clean clothes. It is convenient and desirable to allow a place for this knowledge. The class of a problem can be very useful to a schema-based planner. We will assume problem classes are pre-existing and that it is easy for the planner to tell which class a problem is from. This is not to say that it is necessarily easy to know which schemata apply to a problem. Schema applicability is determined by the satisfiability of each schema's preconditions. The pre-existing classes into which problems fall reflect whatever *a priori* knowledge we desire to express about natural problem types. We may view all problems as coming from a single class. We will assume, however, that in the world's distribution, problems appear from each problem class with a finite nonzero probability.

It is desirable to take problem classes into account in computing planner adequacy. Some problem classes (like block stacking) are much easier than others (like bridge building). If we interpret adequacy as applying to the planner as a whole, then the system is free to balance poor performance on some problem classes against excellent performance on others. The system might unfairly exploit a discrepancy in problem difficulty achieving over-all adequacy through flawless execution of block stacking to make up for creating piles of rubble instead of bridges. This violates the spirit of real-world adequacy even though it may be faithful to the letter of the definition. Thus, we require that each individual problem class achieve the user's adequacy requirement independently.

7.1.2. The bias of schema-based planners

The collection of all the system's schemata is its schema library. The *bias* of a schema-based planner is determined by the particular library it possesses. Any schema change shifts the bias, even if the replacement schema is only slightly different from the original. For example, `the take-a-commercial-airplane` schema that originally prefers Fly By Night Airways may be replaced with one that prefers Reliable Air. The planner now has a different collection of schemata and, therefore, a different schema library yielding a different planning bias. Note, however, the effect of the change is local. Relatively few (and necessarily similar) problems are susceptible to the change. Other tasks (making breakfast, getting groceries, stacking blocks, etc.) are all handled in the very same way under the new bias.

With the bias of a planner supplied by a schema library, the bias space exhibits the kind of interdependence we hoped for. The bias space grows exponentially with the number of problem classes, but evidence against one bias element counts as evidence against all other biases that contain the offending schema. If the switch to Reliable Air results in higher adequacy within a bias in which, say, rental cars are preferred to one's own car, the Whippet bus line is preferred to Pathways, and scrambled eggs are preferred to waffles, then it is also likely to be a desirable airline substitution if the car, bus, and breakfast preferences are resolved differently.

7.1.3. An EBL-defined bias space

We wish to consider libraries composed of schemata acquired through EBL [20,22, 55] over problems sampled from the planner's distribution. A number of systems have been explored for acquiring planning and problem-solving schemata [56,77,78]. The

training example required by EBL is a sample planning problem together with any plan that ISolves it. EBL proceeds by explaining the training example in terms of its background world knowledge. The explanation is then generalized into a new schema. The training example's explanation can be viewed as a proof tree [52,55,57] in which the root of the tree is the achieved instance of the problem's goal, and the leaves are elements from the problem's initial state together with characteristics of the relevant actions. The explanation proves that the commitments taken on by the planner together with aspects of the initial state and background knowledge entail the goal instance. This explanation is generalized by pruning subtrees. This amounts to identifying certain subgoals within the proof tree and removing the commitment that these subgoals need be realized in the manner they happened to be achieved by the training example. For example, a training example might illustrate a particular flight to Baltimore in which Fly-by-Night Airways was taken after purchasing a ticket with money obtained from the Old Farms Mini-Mall cash machine the previous day. The constraints requiring the money be obtained from this (or any) mini-mall cash machine can be pruned leaving the subgoal of possessing money to be solved as a schema precondition prior to application. The destination, the particular airline selected, and so on can also be generalized subject to necessary background knowledge constraints (e.g., that the airline serve the destination). This transformation results in a generalized solution schema.

For our purposes, we will assume that the EBL training example is derived from a planning problem drawn by the planner according to the world's underlying distribution. The problem is then given to a sound constraint-searching planner within the EBL system that expends resources up to some fixed limit in attempting to solve the problem. If the problem can be solved, EBL generalization is applied to the solution yielding in a new schema.

Thus, we will view the EBL system as a partial function mapping problems to schemata. There are several wrinkles that must be discussed. First, a number of quite different schemata might result from any one problem. Second, there is a possibility that the planner internal to the EBL system will fail to ISolve the problem so that no schema is constructed. In regards to this latter problem, call the expected proportion of solvable problems for the world's distribution of problems ρ. We require that ρ be monotonically non-decreasing in the resource bound, R.

Now let us consider how a single problem can lead to several different schemata. Several distinct solutions may exist to a planning problem, resulting in quite different EBL training examples (composed of the problem and its solution). Furthermore, there may be alternative explanations for a single problem–solution pair [72]. Finally, there may be alternative generalizations possible for a single explanation. Thus, a problem corresponds to a set of schemata any one of which may be produced by the EBL system depending on the internal arbitrary choices made during processing.

7.1.4. Honest EBL and conformable schema application

When given a problem, we will require that the underlying EBL system produce at random one of the schemata from the set of possible schemata. Furthermore, the generation process must be somewhat fair. We require that any schema that can in principle be produced will be produced from the problem with no less than some finite

minimal probability, $p_{\min}$. That is, we exclude EBL systems that can hide schemata behind vanishingly small construction probabilities. From this it follows that a problem can lead to only a finite number of schemata, in fact no more than $1/p_{\min}$. This is usually the case in EBL applications. However, the requirement is not met by an important but less well-studied class of schemata in which real-valued parameters appear. For now, we simply note that our results do not hold for domains in which a single problem may lead to an unboundedly large number of schemata. When we refer to an EBL system we assume it has this property. Note that this places no upper bound on the number of schemata that might be built from the domain's problems. There may be an unlimited number of different problems each of which can introduce a finite number of novel schemata.

We also insist that a schema's preconditions not be artificially specific; the preconditions must accurately portray the capabilities of the schema's body. For example, a schema sufficient to ISolve general grocery shopping problems is not allowed to claim only to ISolve problems of acquiring dairy products. More formally, there cannot exist within the precondition language a more general goal pattern nor a less constraining initial state requirement that accurately capture the applicability of the schema. We will call such a schema "honest" since it can commit no sin of omission in advertising its applicability. We will say an EBL system is honest if it produces only honest schemata. Note that we have already forbidden EBL systems the sin of commission. A schema's preconditions cannot falsely claim it is applicable if it will not ISolve the problem.

Our EBL system, then, is assumed to have the following characteristics: We provide it with a resource bound R and access to an unlimited set of problems randomly generated according to the domain's distribution D. When given a problem, EBL produces some schema with probability ρ. With probability $1-\rho$, the EBL system produces Λ, indicating that no schema will be forthcoming. When a schema is produced, it is a random selection from the finite set of schemata for which this problem is an illustration. Each element appears with probability no less than $p_{\min}$. Only schemata that honestly characterize their preconditions are produced. For us, an EBL system is characterized by $p_{\min}$ and ρ.

Now we turn briefly to the planner in which the EBL system is embedded. This is the consumer of the EBL-acquired schemata. We will say that a schema-based planner is "conformable" if a schema can be applied to a problem only if the problem's goals require all of the schema's goal pattern. To illustrate, consider a schema for achieving the conjunctive goal pattern "$\mathrm{ON}(?x, ?y) \wedge \mathrm{ON}(?y, ?z)$". This schema is sufficient to achieve the simple problem goal "$\mathrm{ON}(\mathrm{Blk1}, \mathrm{Blk3})$" but such a use would not be conformable since the schema instantiation would also necessarily achieve an unneeded ON relation. It might seem more desirable to apply an overly specific schema than to fail to produce a solution altogether. However, a schema selection failure conveys important information to the learning component about the abilities of the schema-based planner. This information is obscured if schemata can be employed in a non-conformable (and often inefficient) manner.

These concepts point to one path around the difficulties with ideal permissive planning. Clearly there are alternatives. For our purposes it is enough that *some* form of permissive planning can be made tractable. We can now assemble the formal tractability result:

Lemma 8. *The probability that a schema which has adequacy T_a ESolves a random problem of the appropriate problem class is at least Θ, or equivalently, $(T_a-\beta)/(\alpha-\beta)$.*

Lemma 9. *The probability that a schema of adequacy T_a ISolves a random problem of the appropriate class is at least Θ, or equivalently, $(T_a-\beta)/(\alpha-\beta)$.*

Lemmas 8 and 9 provide a basis for a statistical adequacy test used by the permissive planning algorithm.

Lemma 10. *Given an honest EBL schema acquisition system, L, and a conformable schema application planner, A, it is the case that if a planning problem P can be ISolved by A using a schema, SC, then SC is acquirable by L from P.*

Thus, there is a two-way bridge between problems and schemata, though the mapping is many-to-many. This is important in providing a link from the problems sampled by the planner to a set of candidate schemata worth evaluating.

Lemma 11. *If a planner contains a schema, SC, acquirable through honest EBL which has adequacy T_a or higher for a problem class, then the probability that EBL will in fact construct SC from an arbitrary problem of the appropriate class is at least $p_{\min}\Theta$, or $p_{\min}(T_a-\beta)/(\alpha-\beta)$, where Θ is the normalized utility threshold, α and β are adequacy parameters from Definition 1 and $p_{\min}$ is minimum schema probability factor of the EBL system.*

Lemma 11 will play an important role in our tractability results. Informally, it specifies a limit to how successfully an adequate schema can hide from random probing. The permissive planning algorithm exploits this limit.

7.2. Single-schema planning

We now wish to consider a very simple conformable schema-based planner, called a *single-schema planner.* It is given problems drawn from just one problem class and it may acquire at most one schema from its bias set by applying honest EBL to problems drawn from the world's fixed unknown distribution.

Definition 12. *Single-schema permissive planning* is restricted permissive planning in which the planner is a single-schema planner.

Theorem 13. *Single-schema permissive planning is performed tractably by algorithm SSPP.*

By "tractable" we mean that the sample complexity—the number of problems consumed by the permissive planning algorithm in selecting an adequate bias or determining that none exists—is independent of the cardinality of the bias space and is polynomially bounded in other relevant parameters $(1/\Theta, \alpha, \beta, \gamma, \rho, 1/p_{\min}, 1/\varepsilon, 1/\delta)$. From the

definition of permissive planning, single-schema permissive planning must, with high probability, halt with a correct answer.

Algorithm SSPP(ε, δ, T_a).

GLOBAL: α, β, γ
(* Constants for adequacy computation *)
GLOBAL: ρ, $p_{\min}$
(* Expected characteristic SEBL solution rate reflecting resource choice *)
Set $\Theta = (T_a - \beta)/(\alpha - \beta)$
Set $M = \max(\lceil (3/(p_{\min}\Theta)\ln(6/\delta)\rceil)$
(* number of schema hypotheses needed *)
Set $K = \lceil (\delta\rho M + 3 + \sqrt{6\delta\rho M + 9})/(\delta\rho^2)\rceil$
(* number of problems to reliably generate M schemata *)
Set $N = \lceil M \cdot 2(\alpha - \gamma)/(\varepsilon^2\delta)\rceil$
(* number of problems needed to test a hypothesis *)
Set Tried = 0
Set Schemas = 0
WHILE (Tried $< K$) and (Schemas $< M$) DO
  Set Tried = Tried + 1
  Get a problem P_h
    (* a problem for hypothesis generation *)
  Set SC = SEBL(P_h)
  Set $A_{SC} = 0$
  IF SC $\neq \Lambda$ BEGIN
    Set Schemas = Schemas + 1
    REPEAT N times
      Get a problem P_t
        (* a problem for hypothesis testing *)
      Apply SC to P_t to yield PLAN
      IF PLAN = Λ THEN $A_{SC} = A_{SC} + \beta$
        ELSE IF Execute(PLAN) achieves Goal(P_t) THEN $A_{SC} = A_{SC} + \alpha$
        ELSE $A_{SC} = A_{SC} + \gamma$
      END REPEAT
    END BEGIN IF
  $A_{SC} = A_{SC}/N$
  IF $A_{SC} > T_a$ THEN Exit SSPP with SC
  END WHILE
Exit SSPP with Λ

Informally, the algorithm attempts to construct and evaluate as many as M schemata. M is chosen so that if an adequate schema exists, it is unlikely to be missed in all M trials. Each resulting schema is subjected to a statistical adequacy test composed of N sample problems for which solutions are generated and tested in the world. N is chosen

so that the measured adequacy is unlikely to differ very much from the schema's true adequacy. Since some problems may not yield schemata, a larger number, K, problems must be used to build the M schemata. K is chosen to probabilistically guarantee at least M schemata will result.

It is somewhat surprising that the higher the adequacy threshold, the easier for permissive planning. One might expect that it would take longer to find a good schema than a mediocre one. Upon reflection the true becomes quite intuitive. A schema of high adequacy must correctly apply to many problems and, from Lemma 10, each of these problems can give rise to the schema through EBL. Thus, statistically, random probing affords many more opportunities to learn a highly adequate schema than one that is marginally adequate. On the other hand, if a few random probings all fail, we can be quite certain that no extremely adequate schema exists.

7.3. The DB1 planner

Consider a slightly more complicated planner called a DB1 planner that may have at most one schema for each of a finite number of problem classes and whose bias space, as above, is define through honest EBL over its distribution of problems.

Definition 14. *DB1 permissive planning* is restricted permissive planning in which the planner is a DB1 planner.

Theorem 15. *DB1 permissive planning is tractable provided that problems appear for each problem class with some nonzero lower bound probability* p_{lb}.

7.4. The bias space complexity

We now turn to considerations of the intrinsic complexity of the bias space. We wish to develop an intuition for why permissive planning under the conditions outlined above should be tractable. We confine this brief investigation to single-schema permissive planning where adequacy is the measure of effective real-world single-schema planning over a fixed but unknown distribution of problems.

The difficulty of a learning task is associated with its Vapnik–Chervonenkis (VC) dimension [10]. The VC dimension is defined for classification learning. We are interested not in classification but in learning to plan adequately. To employ this complexity machinery, we define a classification task that is closely related to our planning task. The classification task is designed to have the property that any solution is, with high probability, also a solution to a single-schema permissive planning task. We show that the VC dimension for this related classification task is low.

A classification learning task requires a source of labeled examples that illustrate some target concept along with a space of concept hypotheses. The task is to select an element from the hypothesis space, through examination of the labeled examples, that is likely to be a good approximation to the underlying target concept.

We augment a standard single-schema permissive planning task with two additional provisions: (1) all problems are ISolvable (so that each problem can be unambiguously

labeled as a positive or negative example) and (2) some adequate schema exists. We define a classification task, CT(n) as follows: Let the example source for our classification task be the permissive planner's source of planning problems. Let the hypothesis space be a set of n candidate bias schemata constructed by applying honest EBL to n initial planning problems drawn from the example source. Each hypothesis partitions the space of examples into positive instances (those for which the schema supplies a plan that ESolves the problem) and negative instances (those for which the schema supplies no plan or whose plan does not ESolve the problem). A solution to the CT(n) classification task is any element of the hypothesis space that achieves the desired adequacy T_a.

Lemma 16. *Let*

$$n_0 = \frac{1}{p_{\min}\Theta} \ln\left(\frac{1}{\eta}\right).$$

Then with probability $1-\eta$ *any solution to* CT(n_0) *yields a solution to the corresponding single-schema permissive planning task.*

Lemma 16 assures us that the classification task has the desired relation to single-schema permissive planning. Remember that we are not interested in an algorithm that actually solves the classification task, but in characterizing the VC dimension of the hypothesis set. This is provided by Theorem 17:

Theorem 17. *The VC dimension of the* CT(n_0) *task is no greater than*

$$\log_2\left(\frac{1}{p_{\min}\Theta} \ln\left(\frac{1}{\eta}\right)\right).$$

The log of a polynomial function grows only quite slowly. While a high VC dimension is associated with intrinsically difficult learning tasks, a low VC dimension indicates that the task may admit to efficient learning. The implication of Theorem 17 is that restricted permissive planning itself may be tractable. This result lends strength and generality to results stated in Theorems 13 and 15. Equally important is the assurance it conveys that there is nothing special about the SSPP algorithm nor any subtleties hidden within it. Computer algorithms can be difficult to fully appreciate because the vocabulary employed is extremely expressive. Important features may first appear to be insignificant implementation details. By Theorem 17 we see that the problem itself has a very modest VC dimension. One can reasonably expect to find many alternative algorithms that would perform similarly.

8. Permissive planning in the real world

All formal results including those of the previous section should be treated with suspicion until confirmed with empirical support. This is not to say that a proven theorem or lemma does not hold, but only that its statement, though correct, may be

irrelevant in the real world. From a formal point of view, "tractable" complexity is typically taken to mean "polynomially bounded growth". In the real world, "tractable" requires reasonable *values* of the complexity function as well as reasonable *growth*. If the coefficients are sufficiently large, an algorithm's example complexity may require more observations than the real world can reasonably supply. Even though the cost of running the algorithm grows very slowly, it may be prohibitively expensive.

This is particularly true in the case of robotic planning. Here we may tolerate tens or even hundreds of learning episodes. But training a physical robot through thousands of examples begins to be expensive and a million robot experiences is far too many regardless of how flat the complexity function is at that point.

Yet, one million is a very small number in the context of the richness of robotic behavior. Even a modest robot arm can exist in well over ten billion distinguishable states. An action sequence may involve selecting from this set hundreds or thousands of times.

In this section we examine how permissive planning and the view of alteration of planner bias through machine learning can benefit a real-world robotic planning system. Where might real-world difficulties be lurking in the theoretical results? The first troublesome point concerns the determination of a consistent hypothesis. Many PAC results including those of the previous section depend on efficiently finding an element from the hypothesis space that is consistent with the known data. The previous section establishes that random probing combined with empirical testing suffices. While fine for tiny hypothesis spaces or ones dense with acceptable elements, random probing is intractable from a practical point of view for hypothesis spaces that are large and sparse.

A second difficulty concerns the important relationship between observed problems and potential schemata. We assumed a finite minimal schema production probability, $p_{\min}$. As noted in the last section, this assumption implies that a problem can lead to no more than $1/p_{\min}$ schemata. We also noted that such an assumption is violated for schemata in which real-valued parameters appear. A robot planning schema often involves just such real-valued parameters. Gripper forces, friction coefficients, preferred joint speeds, etc. may take on many different values, and each combination of preferred values embodies a distinct bias for that schema's problems.

The existence of such continuities implies an infinitesimal $p_{\min}$. The requirement of a finite $p_{\min}$ allows SSPP to test schemata in isolation while maintaining "tractability". We did not need to understand the implication of one schema's failure to any other schema.

There is no analog to the $p_{\min}$ requirement in, say, PAC learning of DNF concepts. The success of DNF as a concept representation language in PAC learning can be seen as due in part to the straightforward generalization of information it supports. In DNF concept learning, a single example can provide evidence against many hypotheses at once.

To be freed of the $p_{\min}$ assumption we must support a method by which the rejection of one schema results in the elimination of a large number of additional candidate schemata for that problem class, analogous to DNF concept learning.

The approach we have taken is to diagnose the failure of a schema. The diagnosis is used both to guide the selection of the next hypothesis schema and also to eliminate from consideration other schemata to which the failure diagnosis applies. In this way the

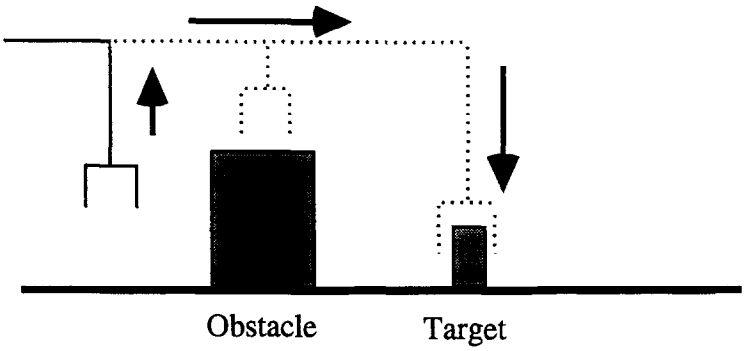


Fig. 4. Plan to reach over an obstacle.

system can be more directed than simple random probing, eliminating the dependence on the confining assumption of $p_{\min}$.

While constructing the conventional explanation, the EBL system also constructs a qualitative explanation for the schema. Qualitative inferences about the interaction of schema parameters can then be drawn. When a schema proves empirically to be inadequate, all other schemata which are qualitatively dominated by it can also be eliminated. These are schemata whose parameter values qualitatively exacerbate the problem for which the schema was rejected according to the qualitative explanation.

Consider the situation depicted in Fig. 4. Here, a gantry-style robot arm is to reach past an obstacle in order to grasp a target block. World objects are modeled as polygons positioned on a table. We will suppose that the system has a set of planning operators including MOVE-UP(x), MOVE-LEFT(x), OPEN-GRIPPER(x) to specify how the world changes in response to its actions. Further, we require that the operators are augmented by a qualitative characterization of their parameterized effects. Thus, the argument to MOVE-UP must be greater than zero, and as the argument increases in value the final distance of the gripper to the ceiling monotonically decreases.

The planning problem is a simple one. A solution, depicted by the dotted line trajectory, is quite easily constructed from the initial state:

Initial state	*Solution*
HEIGHT(OBST, 4.7)	MOVE-UP(2.4)
HEIGHT(TARG, 2.2)	MOVE-RIGHT(8.3)
WIDTH(OBST, 2.6)	OPEN-GRIPPER(1.0)
WIDTH(TARG, 1.0)	MOVE-DOWN(3.7)
POSITION(OBST, 4.5)	CLOSE-GRIPPER(1.0)
POSITION(TARG, 10.9)	
GRIPPER-AT(2.6, 2.1)	

and this solution is readily generalized via EBL into a schema for a class of similar problems:

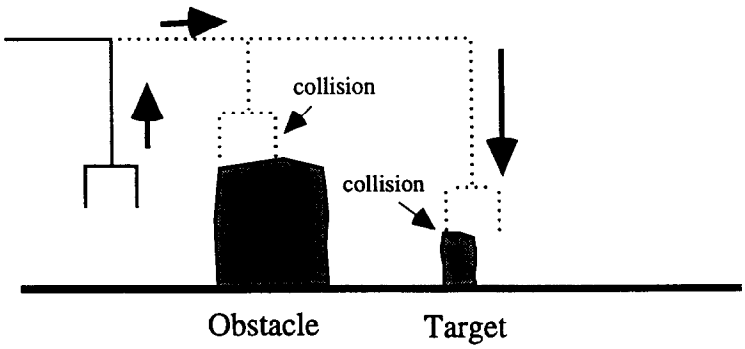


Fig. 5. Execution of the reach plan.

Initial state	*Solution*
HEIGHT(?Obstacle, ?OH)	MOVE-UP(?OH – ?GY)
HEIGHT(?Target, ?TH)	MOVE-RIGHT(?TP – ?GX)
WIDTH(?Obstacle, ?OW)	OPEN-GRIPPER(?TW)
WIDTH(?Target, ?TW)	MOVE-DOWN(?OH – ?TH)
POSITION(?Obstacle, ?OP)	CLOSE-GRIPPER(?TW)
POSITION(?Target, ?TP)	
GRIPPER-AT(?GX, ?GY)	

The real-world execution of the plan will look more like the one depicted in Fig. 5. Here, the plan suffers two distinct failures manifested by collisions of the gripper with world objects. Each collision arises because of a combination of slightly incorrect beliefs: (a) the MOVE-UP action fails to raise the arm as high as expected while the obstacle block is taller than believed and (b) the MOVE-RIGHT extends the arm farther than expected and the target block is located closer than believed.

The planning bias of the schema above includes minimally missing the obstacle and minimally surrounding the target. The impracticality of such a bias is obvious. However, the particular bias is not so important. The relevant point to us is that *some* bias is necessarily embodied and it is unavoidably based on very little experience in the class of problems that the schema claims to solve. No matter what the initial bias, we can have little confidence in its expected performance.

When the first failure of Fig. 5 is encountered, permissive planning dictates that the schema's bias be altered. With the additional qualitative knowledge the alteration can be directed rather than random.

First, the observed failure is diagnosed. All the system knows is that the MOVE-RIGHT ended prematurely. The failure observation is of an unexpected collision force. But there is no way to know the identity of the collising object. The system embodies the assumption that represented knowledge is approximate. This says that the real world is more likely to be a small perturbation from the represented world than a large one. The collision could be with the target block or the table but since the obstacle block is

nearest the observed collision point, it is selected as the first candidate in the collision explanation.

Second, a set of qualitatively relevant plan parameters is identified. These are constants employed in the executed whose values were not fully entailed during planning. As such they represent part of the planner's bias; the collection of all possible parameter settings forms the planner's bias space. Each constant is examined in light of the explanation to identify those whose value may qualitatively influence the failure. Here, there is only one such constant. It is the argument to the first action MOVE-UP. It is entailed to be at least ?OH − ?GY to satisfy the precondition of a clear path for the interrupted MOVE-RIGHT. The argument ?OH − ?GY was believed adequate for that purpose. It is observed not to be contingent upon this first failure explanation being correct.

Finally, the bias is adjusted. Qualitatively the likelihood of a collision with the obstacle is minimized as the argument to the action MOVE-UP is maximized. The value ?OH − ?GY, is noted to be too small. The schema is amended by adding a constraint and a preference. The constraint encodes the fact that the MOVE-UP argument must be greater than the value ?OH − ?GY. The preference is qualitative in nature and specifies that expected utility improves as the argument value is increased. From now on the system will have a bias of moving as high as possible when reaching over an obstacle. Together constraints and preferences constitute a generalized memory of the failure experience to influence future processing.

Such constraints and preferences can focus attention away from many additional schemata besides the one evaluated. This is the generalization of experience that was discussed earlier.

It may happen that no additional failure is encountered that can be explained by this plan parameter. In that case it has reached an adequate value. On the other hand, some later qualitatively different type of failure may occur for the reason that the arm is now reaching too high. When such a failure is encountered, the new and quite different explanation will post a constraint that the MOVE-UP argument be less than the discovered maximal bound. It will also post a qualitative preference that the argument be as small as possible. The utility for selecting the argument is then seen as qualitatively convex. It increases at the low end of the allowable interval but decreases at the high end. Qualitatively the midpoint is the best choice. Further failures would post further constraints and preferences. Thus, a rough binary search is realized in pinning down the best value of the MOVE-UP argument.

The system may also encounter a collision while attempting to grasp the target (the second collision of Fig. 5). In a similar way the bias is adjusted so that the gripper opens as wide as possible when approaching a grasp target. Eventually permissive planning produces a solution like the one shown in Fig. 6.

Fig. 7 diagrams the improved ESolve behavior of the reach-over-obstacle schema. The original schema tries to minimally miss the obstacle and minimally open to grasp the target. Its real-world success, therefore, is extremely limited. After permissive planning the schema applies much more broadly. In this example the result is similar to that produced by a planner which incorporates explicit uncertainty reasoning. This need not be the case, however. In permissive planning there is no guarantee that the improved ESolve coverage will contain the ESolve coverage of the original. Nor is there any

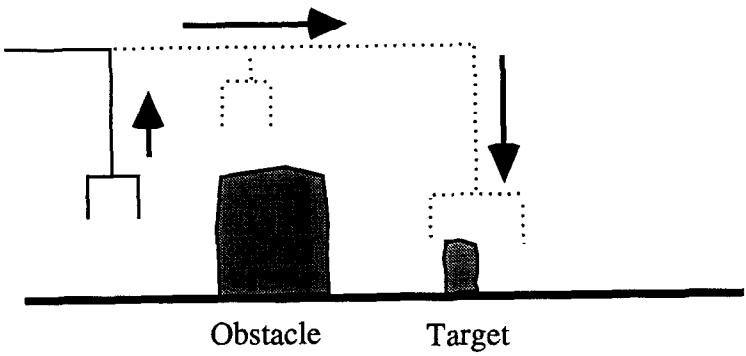


Fig. 6. A conservative solution.

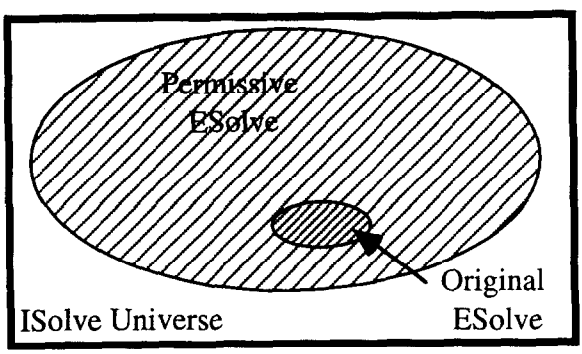


Fig. 7. Permissive versus original reach schema coverage.

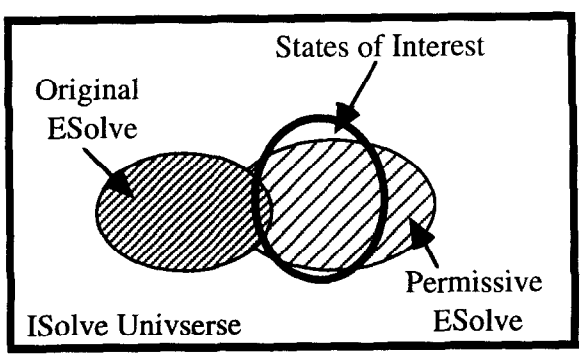


Fig. 8. More typical result of permissive planning.

guarantee that the ESolve coverage will be larger than that of the original schema, although often this is the case.

Fig. 8 shows a more typical relationship. The permissive planning guarantee is that after adjustment, the ESolve behavior will better match the user's needs. Here, the

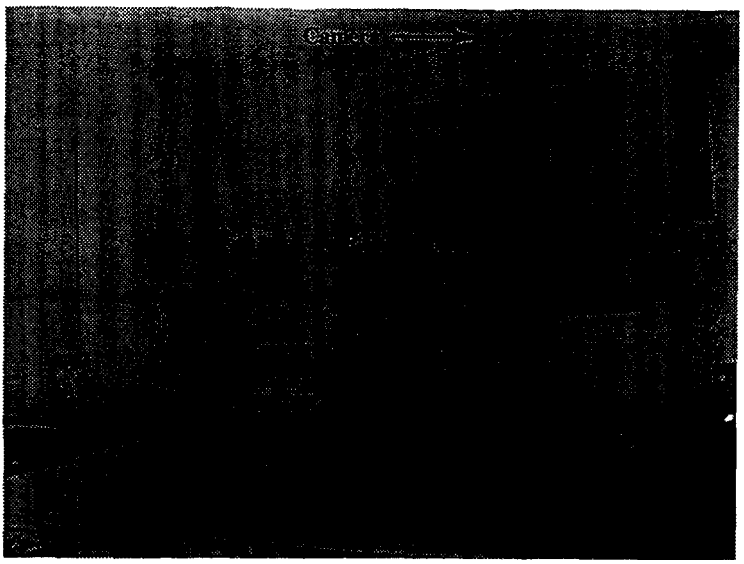


Fig. 9. The GRASPER system.

overlap between the permissive ESolve behavior and the states of interest is much improved over that of the original schema, although the original ESolve set is not contained in the adjusted set.

An important characteristic of permissive planning is that it does not try to eliminate planner weaknesses. Such an approach would be doomed; no planner can *solve* the qualification problem. In permissive planning, planner weaknesses remain but are moved away from the kinds of problems that the user asks the system to solve. Likewise, planner strengths are brought more into line with the user's needs.

9. The GRASPER system

In this section we present a brief overview of GRASPER, an implemented real-world permissive planning robotic system. The system, shown in Fig. 9, consists of a six degree of freedom Prab RTX scara-type manipulator, an overhead camera, a frame grabber, and an IBM RT computer running Lucid Common Lisp.

GRASPER plans action sequences to grasp and lift novel objects placed in its workspace, such as plastic pieces from a children's puzzle (Fig. 10). No *a priori* model of the objects are given to the system. They are known only through the silhouette they present to the overhead camera. The pixel patterns from the camera are converted into simple polygons. Fig. 11 shows a typical object, its contour edges as found by the vision system, and its polygonal internal representation. The polygon object representations are used for drawing conclusions about spatial occupancy. This representation (like any internal representation of a real physical object) is flawed, being only an approximation to the actual objects.

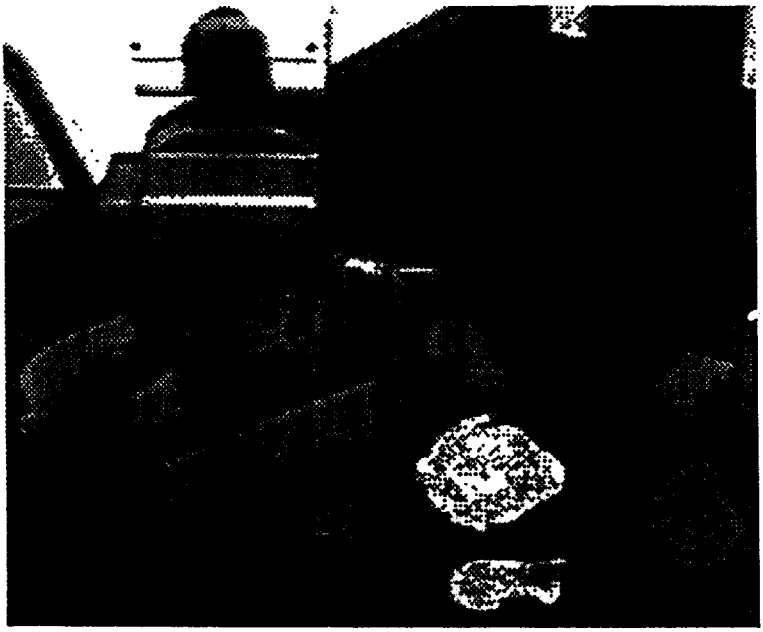


Fig. 10. The workspace with gripper and pieces.

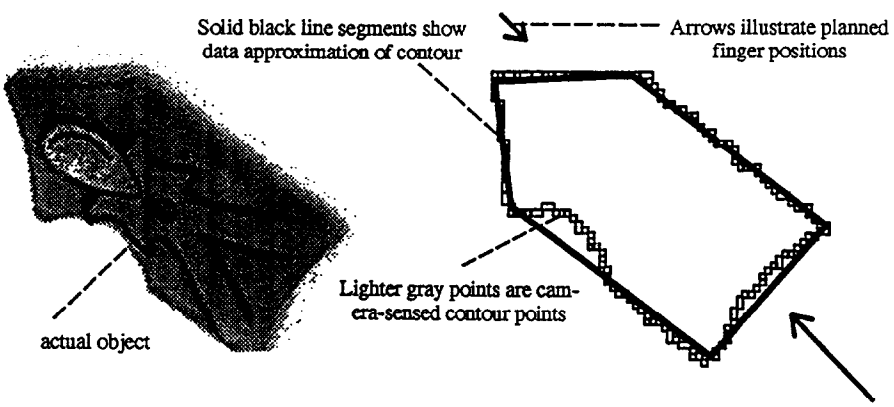


Fig. 11. An object, the sensed contour, and the represented polygon.

The system includes a typical axiomatization about how objects can be surrounded, how closing the gripper applies a friction force between the fingers and the object, how sufficient friction establishes a grasp, how a grasped object can be manipulated, etc. The axiomatization, like the represented sensory data, captures the real world only approximately. The coefficients of friction between the fingers and various objects is represented as a single scalar. This is a simplification. Operators to move the arm only

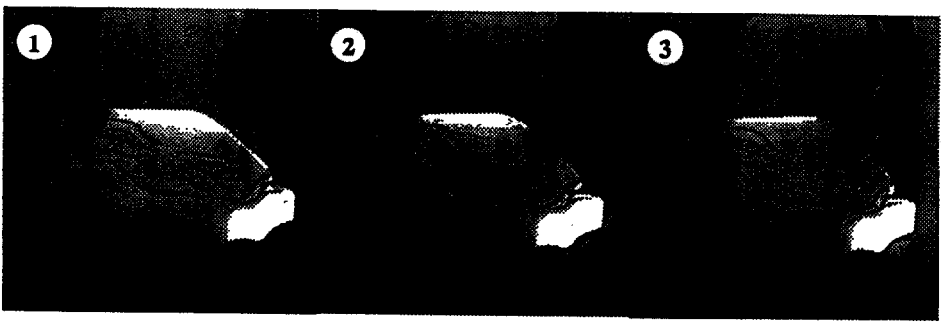


Fig. 12. A finger stubbing failure.

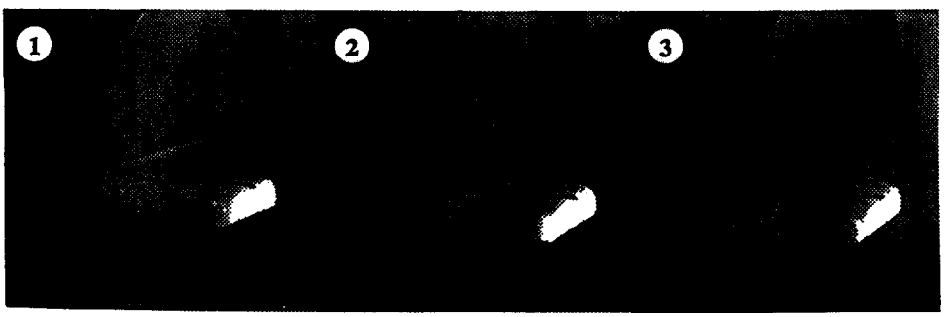


Fig. 13. A horizontal slipping failure.

approximate the motions they claim to perform. The forces that the robot fingers apply to a surrounded object also only approximate the operator's effects.

Using its initial knowledge, a plan is constructed to lift a designated object. Not surprisingly, the initial plan usually fails in the real world. Figs. 12 and 13 illustrate two of the failure modes observed.

Following the principle of permissive planning, the planning process, rather than the domain theory, is blamed for the shortcoming. The planning bias is adjusted. Through bias adjustment over several failures, the real-world effects of the schema are made to conform to the projection of the original action sequence. The final PICK-UP schema can be interpreted as (1) squeezing harder than the world knowledge claims is necessary, (2) selecting grasp points along faces that are more nearly parallel, (3) selecting grasp points closer to the object's center of geometry than believed necessary, and (4) opening the gripper wider than believed necessary while appoaching the target object.

Figs. 14 and 15 illustrate a typical application of the schema after learning. After bias adjustment has occurred, this object, like most novel 2D objects, is correctly grasped and lifted on the first attempt.

Fig. 16 quantitatively illustrates the increased robustness of GRASPER's PICK-UP schema. The solid region indicates how far the real world can vary without causing the grasp to fail before and after permissive adjustment. The ESolve envelope is much larger for schema after permissive planning indicating that these attributes can vary

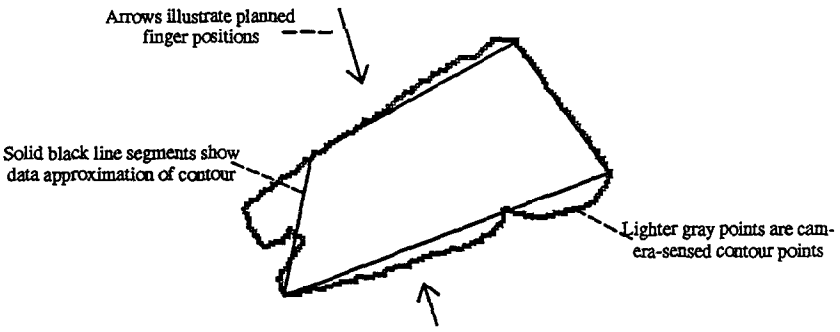


Fig. 14. Target object with planned finger positions.

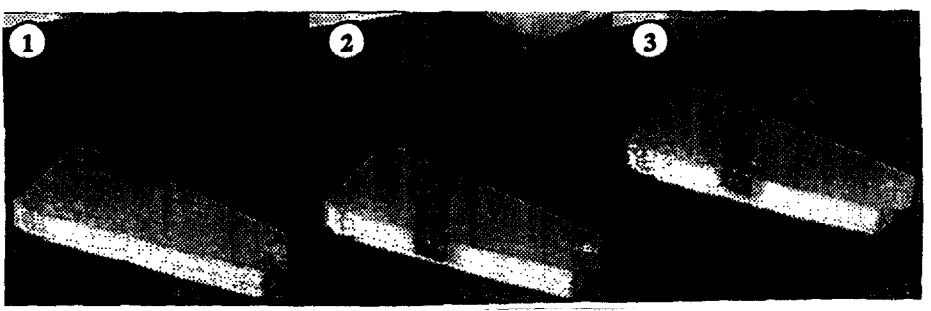
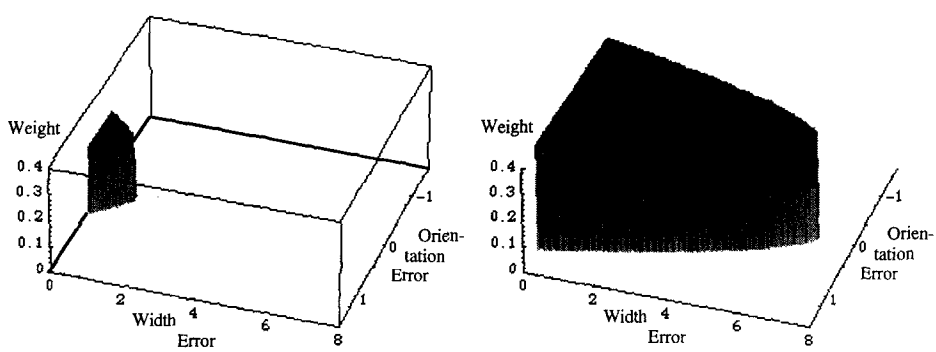


Fig. 15. A successful grasp after permissivization.



a) Before Permissive Planning b) After Permissive Planning

Fig. 16. ESolve envelopes.

much more widely and still be successfully picked up. Three attributes are represented: weight, excess width, and orientation. Other attributes (X & Y position on the table, the angle between the grasped faces, and the distance from the center of mass of the selected grasp points) would also show improved tolerance.

The object being grasped is similar to the one in Fig. 15. As heavier target objects are attempted by the schema it eventually fails with the target slipping out of the gripper.

As shown by the vertical axis this failure occurs at approximately twice the mass in the permissively adjusted schema. More interesting is the interaction between the width and orientation errors. As the true width increases (the difference between the true width and the represented width is shown on the axis extending to the right), the schema's tolerance of orientation errors (the axis perpendicular to the page) quickly decreases. Before permissive planning the tolerated variation in orientation is ± 0.09 radians (about 5 degrees) with an object just 1 cm wider than believed. After permissive planning an object 4 cm wider than believed can still be successfully grasped when approached with an orientation error of ± 0.64 radians (over 30 degrees).

The GRASPER system satisfies the requirements of Theorem 13. It is a single-schema planner; its only goal is to pick up objects. Exactly one schema is constructed for this purpose. The single-schema planner family was defined to establish a formal result, and while it is a particularly simple family of planners, GRASPER demonstrates that it is not without interesting real-world members.

How central is learning to GRASPER's performance? We could have invested in more expensive robotics hardware that would support a more certain domain theory. This would increase the chance that a grasping episode would fall within the original ESolve envelope of Fig. 16(a). Certainly the hardware supporting the GRASPER system is far from the state of the art. With the best available robot arm, with a state-of-the-art vision system, with a more sophisticated internal object representation, and with carefully crafted planning operators we could have avoided the pitfalls that made permissive planning necessary for the GRASPER system. But this would only defer the problem. With better hardware the same phenomenon arises at a smaller grain size. Manipulating high precision parts is just as susceptible to the qualification problem as manipulating GRASPER's puzzle pieces. Furthermore, improving accuracy is an expensive option and is not always available to users. Hardware costs skyrocket with higher precision and better reproducibility requirements. Most importantly, such a "solution" misses the point of the research. Permissive planning opens a different avenue to achieving adequate real-world performance. One that complements buying more expensive hardware.

Finally, we are making no claims that the GRASPER system solves the general non-model-based grasping problem. This is an exceedingly difficult and still open problem in robotics. Our claim is only that bias adjustment is an important tool for planners that interact with a real world. Robotic grasping is one task where a real-world connection is crucial. Thus, GRASPER is an implemented existence proof of the tractability of bias adjustment through permissive planning.

For additional details of the GRASPER system see [5]. Another task to which permissive planning has been successfully applied can be found in [6].

10. Summary and conclusions

We have outlined an extension of the classical planning paradigm to embrace real-world performance. We introduced a distinction between *ISolving* a planning problem—finding a plan that works according to the planner's micro-world, and *ESolving* a

problem—achieving the goal in the real world. We used these concepts to define the notion of *planner adequacy*. We discussed the *planning bias* of a planner and argued that bias is an inescapable facet of classical planning. We introduced *permissive planning* as a method of improving planner adequacy by purposeful alteration of planner bias. This is not a facet of traditional classical planning, which allows no formal connection between the system's internal micro-world and any real world. Permissive planning necessarily contains an empirical component. For it is only through observation that real-world behavior not captured by the planner's micro-world definition can be known. Thus permissive planning can be viewed as an empirically driven search through a space of possible planning biases so as to reduce the discrepancy between the projections of action sequences and their real-world effects. While adequacy is enhanced by permissive planning, the micro-world behavior and the projection abilities of the planner remain unchanged.

In its ideal form, permissive planning is impossible. However, with modest constraints, permissive planning is both possible and tractable. We are able to establish a slightly strengthened form of tractability. By convention in computational complexity, something is "tractable" if its growth is polynomially bounded. But, an algorithm whose running time is polynomial in a feature that is known to be extremely large can paralyze an implemented system as surely as hyper-polynomial growth. The one feature of permissive planning liable to such characteristics is the bias space. We offer two points to redress this concern. First, we can guarantee the algorithmic complexity of permissive planning is independent of the cardinality of the bias space. Second, we cite an implemented permissive planning robotic planning system as an existence proof that the theoretical results can be realized in a real-world planner.

We show that schema-based planners provide a convenient method to realize bias adjustment. In particular, schematic planners exhibit a locality property: changing a schema results in alternation of the planner's bias only in the local context of the schema. This yields a desirable form of interdependence among elements of the bias space, a key requirement for tractably achieving planner adequacy. However, an important topic for future research is to investigate what other forms the bias adjustment can take.

Regardless of how the bias space is realized, one somewhat surprising corollary of permissive planning is that minimal commitment plans may not be desirable. This is not to say that minimal commitment *planning* is incompatible with permissive planning. It most assuredly is not. It is the form of the output plan that causes difficulty. These planners cease the plan refinement process when all plan completions entail the goal [14, 17, 45, 68, 75, 86]. But the planner's output must be linearized into a single action sequence for executing in most real-world domains. This is done after planning ceases and is not part of the minimal commitment planning process proper. Thus, only a portion of the bias inherent in the executed action sequence is due to the planner. Any permissive adjustment of a minimal commitment planner is denied influence over that portion of the bias exhibited by plan linearization. To remedy this, the linearizing decisions must also be subject to bias control. One might attempt to exert a permissive planning-like control over both the planner and linearizer, but this introduces the artificial and potentially difficult complication of coordinating bias adjustment between the two

interacting bias spaces. A preferable solution is to put the linearization decisions back into the planner. The planner may still employ minimal commitment principles in making planning decisions, but it would not be allowed to cease making decisions until a totally ordered action sequence is produced.

Permissive planning is, in essence, an approach to planning with uncertainty. The centrality of machine learning sets it apart from most conventional approaches such as planning for sensing [34,63], decision theoretic planning [19,31,32], preimage backchaining [25,42,43], and disjunctive planning [69,76].

Its close tie to classical planning makes it quite different from the reactive approaches [1,29,44], and it has little in common with the fuzzy control approach [8,24]. As in conventional symbolic AI, everything is done using "crisp" representations. There are no fuzzification or de-fuzzification steps.

Much previous research has applied machine learning techniques to planning. Usually, machine learning is employed to improve the accuracy of a planner's model of the world. This includes acquiring or adjusting operator definitions [13,21,35], reinforcement learning [37,58,79,84], and scientific discovery [62,71]. This alters the planner's projection ability rather than its bias. The chunking of control knowledge (e.g., [41,50, 88]) can be viewed as altering planner bias as can some case-based approaches [2,33] though in these systems there is no formal connection to planner adequacy, which is central to permissive planning.

This is not to say that permissive planning is necessarily preferable to other uncertainty approaches. The approaches are different, each has its strengths and weaknesses. Some important strengths of permissive planning are (1) it offers a formal connection to a complex and fundamentally unmodelable real world, (2) a single adjustment of the planning bias can mitigate several domain theory flaws, for example in the GRASPER system, opening the gripper wider to surround the object helps alleviate uncertainties about its position, orientation, and shape, (3) desirable features of classical planning, such as domain independence and guarantee of (ISolve) correctness are retained as is the benefit from the extensive literature and prior research on classical planning, (4) it preserves the representations supplied by the human domain expert which may well embody a deep coherence based upon his own hard-fought experiences with the world.

Permissive planning also exhibits some notable weaknesses: (1) it requires a declarative treatment of planner bias; this adds a new and potentially difficult dimension to the design of a planner, (2) potentially costly real-world failures must be experienced, and (3) the acquired knowledge does not apply in different contexts. Unlike approaches that refine internal domain representations, permissive planning learns to identify and avoid contexts that lead to problematic interactions rather than fixing the underlying discrepancies. This is the flip side of advantage (2) above.

In a broad sense, permissive planning can be viewed as learning to be conservative in choosing actions. Why not simply build a preference for conservative choices into the planner to begin with? The answer is that being conservative is context dependent. It is sensitive to the pragmatics of the situation and to empirical tradeoffs in the world. These cannot be properly specified without examining the real world. For example, in a sparse workspace GRASPER is conservative if it opens its gripper as wide as possible before approaching a target piece. In a cluttered workspace, however, excessive opening may

cause new collisions with other objects. To be most conservative it learns to open to an intermediate amount guided by the observed failures. Likewise, closing more tightly than believed necessary is conservative in case the believed coefficient of friction is low. But squeezing harder can be destructive if the objects are fragile.

Finally, we wish to offer a more abstract (and more speculative) interpretation of the permissive planning approach. Let us consider the role of the domain theory, including operator definitions, object descriptions, etc. What kind of domain theory leads to acceptable planning behavior? For a conventional planner, accuracy is paramount. It is only through faithfulness of the domain theory to the external world that the planner's output can be trusted to achieve what it claims to achieve. Domain theory accuracy is also the crucial issue for many learning planners. Learning is most often directed toward refining internal representations of the domain theory when discrepancies are detected.

Permissive planning changes this. The primary concern is still that the planner's output do in the real world what it claims to do. But this no longer translates directly into a requirement for the domain theory to be accurate. The plans constructed by a permissivized planner reflect both the domain theory and the permissivized changes to the planner's bias. The permissivization procedure affords a degree of freedom in the planner's behavior beyond the dictates of the domain theory. What makes a good domain theory for a permissive planner? The best domain theory is the one that results in the best behavior after permissive bias adjustment. An inaccurate domain theory that allows more effective permissive adjustment may be more desirable than a more accurate domain theory that allows little adjustment. The shift may seem small but the implications can be profound. A similar issue arises in [21]. Correctness is now but a heuristic to be used as a guide in crafting the domain theory. In the GRASPER system simple polygons are employed to represent object shapes. These are known to be inaccurate in ways that would be pernicious to a conventional planner. However, GRASPER's permissive planning (which employs the same simplified representations) exhibits much more effective final behavior than if it were reasoning about the intricate complexities of objects' true silhouettes. The simplicity of the data representations together with the conservative bias to open wider than necessary yield planning that is both efficient and effective in the real world. This ability to tolerate less than perfect axioms can greatly simplify the task of the system implementor. Without it the implementor shoulders the impossible task of producing a perfectly accurate set of axioms. In reality this degenerates into a game to spread and hide the imperfections so that their mutual negative interactions are unlikely to be discovered by the anticipated lines of inference that the system will likely explore (which the implementor is now forced to anticipate). Using permissive planning, one can adopt domain axiom sets with latent inconsistencies, relying on empirical refinement to select preferences for adequate planning behavior. Not all types of inconsistencies can be tolerated; one cannot start with gibberish and expect permissive planning to sort things out. We hope that the kind of inconsistencies tolerated include the kind one tends to get as the initial world model from a domain expert. If this is the case permissive planning may be of great use in helping to understand what experts mean when they are forced to express their expertise in a formal language. But this is as yet only a hope and a question for later investigations.

Acknowledgements

The authors are indebted to Lenny Pitt for helpful discussions, to Renee Baillargeon for invaluable comments on earlier drafts, and to two anonymous reviewers whose comments greatly improved the clarity of presentation. This research was supported in part by the Office of Naval Research under grant N00014-94-1-0684.

Appendix A. Proofs of lemmas and theorems

Lemma 4. *The simplified formulation of ideal permissive planning cannot be realized.*

Proof. Under the conditions of simplified ideal permissive planning, the biases are independent Bernoulli random variables. We can know their characteristics only by sampling problems according to the user's interest and observing success or failure. This is precisely the discrete time minimax bandit problem with two Bernoulli arms as the horizon N grows without bound. A permissive planning algorithm is any decision procedure which after a finite sampling of the two biases selects, once and for all, the better of the two. This correspond to a regret function that is asymptotically flat in N. An important result from the bandit literature is that the optimal regret function grows at least as $\sqrt{N}$. Since no regret function can, under these conditions, be asymptotically flat, no algorithm can realize simplified permissive planning. □

Theorem 3. *Ideal permissive planning cannot be realized.*

Proof. The theorem follows directly from Lemma 4. By choosing a bias space of cardinality two, and choosing $\gamma = \beta$, any ideal permissive planning algorithm necessarily satisfies the more restrictive conditions of the lemma. Since no algorithm can satisfy the lemma's restrictive conditions, the more general case is also impossible to realize. □

Lemma 8. *The probability that a schema which has adequacy T_a ESolves a random problem of the appropriate problem class is at least Θ, or equivalently, $(T_a-\beta)/(\alpha-\beta)$.*

Proof. Consider a schema of adequacy T_a applied to a random problem of the appropriate class drawn from the universe and distribution of (1). Let r be the probability that such a problem is both ISolved and ESolved, p be the probability that it is not attempted by the schema, and q be the probability that it is ISolved but not ESolved. The definition of adequacy (1) can be written as

$$T_a = r\alpha + p\beta + q\gamma. \tag{A.1}$$

Suppose the lemma is violated for some schema. Then for that schema $r < (T_a - \beta)/(\alpha - \beta)$ which can be rewritten as $T_a > r\alpha + (1 - r)\beta$. With (A.1) we have $r\alpha + p\beta + p\gamma > r\alpha + (1 - r)\beta$ or $q\gamma > (1 - p - r)\beta$. But $r + p + q = 1$ because these probabilities exhaust the possible outcomes. So it must be the case that $\gamma > \beta$. This contradicts the definitional requirement that $\gamma \leqslant \beta \leqslant 0 < \alpha$ proving the lemma. □

Lemma 9. *The probability that a schema of adequacy T_a ISolves a random problem of the appropriate class is at least Θ, or equivalently, $(T_a - \beta)/(\alpha - \beta)$.*

Proof. This follows from the proof of Lemma 8 which established this as a lower bound on problems that are *both* ISolved and ESolved. There can be no fewer problems that are ISolved without regard to being ESolved. □

Lemma 10. *Given an honest EBL schema acquisition system, L, and a conformable schema application planner, A, it is the case that if a planning problem P can be ISolved by A using a schema, SC, then SC is acquirable by L from P.*

Proof (*Sketch*). We can think about each schema as embodying a planning strategy that can be applied directly, without recourse to searching the constraint tree. Acquiring planning schemata through EBL can be viewed as the process of

(1) observing an instance that illustrates an unknown strategy,
(2) explaining why the instance works,
(3) abstracting away unneeded and easily rederivable details of the example to yield a general characterization of the illustrated strategy.

If a planning strategy suffices to solve a problem, then some from-scratch plan can be constructed (by the planner underlying the EBL component) which is an example of the strategy. This plan, suitably explained, can therefore be generalized back into the original strategy. Of course, alternate from-scratch solutions might lead to other strategies, and, therefore, other schemata. But among the schemata acquirable from the planning problem must be the originally postulated schema. The only wrinkle in transforming this informal argument into a proof is to insure a tight fit between the problem and the strategy: the problem solution must necessarily illustrate all facets of the strategy. This property is enforced by the honesty and conformability requirements. □

Lemma 11. *If a planner contains a schema, SC, acquirable through honest EBL which has adequacy T_a or higher for a problem class, then the probability that EBL will in fact construct SC from an arbitrary problem of the appropriate class is at least $p_{\min}\Theta$, or $p_{\min}(T_a - \beta)/(\alpha - \beta)$, where Θ is the normalized utility threshold, α and β are adequacy parameters from Definition 1 and $p_{\min}$ is minimum schema probability factor of the EBL system.*

Proof. By hypothesis, the planner contains a schema of adequacy at least T_a. Call that schema SC. From Lemma 9 the expected ISolve solution rate of SC over relevant problems is at least $(T_a - \beta)/(\alpha - \beta)$. From Lemma 10 we know that each of the ISolvable problems could in principle give rise to the acquisition of SC through honest EBL. Thus, the expected fraction of problems of the appropriate class that could in principle give rise to SC is $(T_a - \beta)/(\alpha - \beta)$. For us, the minimum probability with which EBL produces a particular schema from a problem is $p_{\min}$. Thus, the probability that SC is produced by EBL on a random problem of the appropriate type is at least $p_{\min}(T_a - \beta)/(\alpha - \beta)$, or by the definition of Θ, $p_{\min}\Theta$. □

Theorem 13. *Single-schema permissive planning is performed tractably by algorithm SSPP.*

Proof. The algorithm always halts: From the loop structure, the algorithm's time complexity is $O(KN)$. As many as K problems are drawn in the outer loop. For as many as M (with $M \leqslant K$) iterations, the inner loop is executed N times drawing one problem in each iteration. Thus, the sample complexity is bounded by $K + MN$. The parameters M, K, and N are assigned values that are independent of the cardinality of the bias space and polynomial in the other relevant parameters. There are two ways the algorithm might yield an incorrect aswer: (1) it may output Λ, failing to find an adequate schema when one exists or (2) it may halt with an inadequate schema (whether or not an adequate one exists). The first error is a false negative. The second is a false positive. It suffices to show that Pr(false negative) $\leqslant \delta/2$ and Pr(false positive) $\leqslant \delta/2$. A false positive results if any one of the M hypothesized schemata is incorrectly judged to be adequate. To avoid a false positive, it suffices that $\sum_M \delta_i \leqslant \delta/2$ where δ_i is the independent probability that the ith hypothesized schema will be falsely confirmed as adequate. Thus, it suffices to show that $\delta_i \leqslant \delta/(2M)$ for each i. Each test applies its schema to N sample problems from which the sample adequacy is computed. Provided the variance over a randomly sampled distribution is finite, Chebyshev's inequality provides a useful relation for PAC-style proofs:

$$\Pr(|\mu - \overline{X}| \geqslant \Delta) \leqslant \frac{\sigma^2}{\Delta^2 N} \tag{A.2}$$

where m and σ^2 are the true mean and true variance respectively of a random variable X, and the symbol $\overline{X}$ represents the measured mean of N samples of X. Δ is positive. The relation quantifies the probability of observing a measured mean very different from the true mean.

Let

- SC be a hypothesized schema under evaluation,
- A_s be the computed sample adequacy of SC over N random problems,
- A_t be the (unknown) true adequacy of SC over the underlying distribution.

The underlying distribution of adequacies is trinomial: each sample problem adequacy is α, β, or γ. So σ^2 is also finite and Chebyshev's inequality applies to the adequacy of SC. For any positive Δ

$$\Pr(|A_s - A_t| \geqslant \Delta) \leqslant \frac{\alpha - \gamma}{\Delta^2 N}. \tag{A.3}$$

A necessary and sufficient condition for SC to be judged a false positive is that its sample adequacy is measured to be at least the desired adequacy threshold but its true adequacy is less than the threshold by at least ε: $(A_s \geqslant T_a) \wedge (A_t < T_a - \varepsilon)$. Thus, a sufficient condition for avoiding a false positive is that for each test

$$|A_s - A_t| < \varepsilon \tag{A.4}$$

Chebyshev provides a bound on the probability that inequality (A.4) is violated:

$$\Pr(|A_s - A_t| \geqslant \varepsilon) \leqslant \frac{\alpha - \gamma}{\varepsilon^2 N}.$$

It suffices to choose N according to

$$N \geqslant M \cdot \frac{2(\alpha - \beta)}{\varepsilon^2 \delta}$$

which is respected by the algorithm. N is independent of the cardinality of the bias space (the number of entertainable schemata) and that it is polynomially bounded in the other parameters provided M is likewise bounded which will be established shortly.

The probability of a false negative must also be bounded by $\delta/2$. For a false negative condition to arise there must exist at least one adequate schema but the algorithm claims none exists. There are three ways the algorithm might result in a false negative. First, the K sampled problems may be sufficiently difficult that fewer than M can be solved. In this case, the algorithm cannot construct the requisite M random hypothesis schemata. Call this false negative failure F_{fn1}. The second type of failure results in case no adequate schema is entertained among the M hypotheses. We call this F_{fn2}. Finally, it is possible that one or more adequate schemata are hypothesized but they are judged to be inadequate by the statistical adequacy tests. Call this failure F_{fn3}. There are no other ways a false negative can occur. Thus, we require that

$$\Pr(F_{\text{fn1}}) + \Pr(F_{\text{fn2}}) + \Pr(F_{\text{fn3}}) \leqslant \frac{\delta}{2}.$$

It suffices to show $\Pr(F_{\text{fn1}}) \leqslant \delta/6$ and $\Pr(F_{\text{fn2}}) \leqslant \delta/6$ and $\Pr(F_{\text{fn3}}) \leqslant \delta/6$. First, we consider F_{fn1}. We show for the given method of selecting K that with probability $1-\delta/6$ at least M of the K sampled problems are solvable. Since every solvable problem yields a schema through EBL, we will be probabilistically guaranteed to entertain the requisite number of schemata. We assume the difficulty of a problem is an independent property of the problem. Provided the resources allocated to the EBL planner are in the right ballpark for the intrinsic difficulties of the problem class, this assumption seems reasonable. Let us define a new random variable S. S_i is 1 if problem i is solved within the resources allowed and 0 if it is not. The mean of S, of course, is ρ, the expected proportion of problems that can be solved. If we choose K to be M/ρ then there is an even chance that at least M of the problems are solvable. But an even chance is not good enough. We must limit the chance of getting fewer than M solvable problems to $\delta/6$. Let $\rho_{\bar{s}}$ be the observed solution rate corresponding to the true rate, ρ. We require

$$\Pr\left(\rho_{\bar{s}} \leqslant \frac{M}{K}\right) \leqslant \frac{\delta}{6} \quad \text{or}$$

$$\Pr\left(\rho - \rho_{\bar{s}} \geqslant \frac{K\rho - M}{K}\right) \leqslant \frac{\delta}{6}. \tag{A.5}$$

If we let the number of samples be K, define $\Delta = (K\rho - M)/K$, and note that the variance of S is at most 1, then Chebyshev assures us that

$$\Pr\left(|\rho - \rho_{\bar{s}}| \geqslant \frac{K\rho - M}{K}\right) \leqslant \frac{1}{\left(\frac{K\rho - M}{K}\right)^2 K}. \tag{A.6}$$

The left-hand side of (A.5) is weaker than that of (A.6) since Chebyshev provides an unneeded bound on how much higher the observed solution rate, $\rho_{\bar{s}}$, can be than the true rate, ρ. We require only that the observed rate not often be much smaller. If we can choose K so that

$$\frac{1}{\left(\frac{K\rho - M}{K}\right)^2 K} \leqslant \frac{\delta}{6} \quad \text{or} \quad K \geqslant \frac{\delta\rho M + 3 + \sqrt{6\delta\rho M + 9}}{\delta\rho^2},$$

the SSPP algorithm chooses K consistent with this bound which is independent of the size of the bias space and polynomial in the other parameters. However, M again appears linearly. Thus, a tractable number of examples can probabilistically guarantee a low probability of F_{fn1} again on the tractability of M.

Next we turn to the possibility that no adequate schema is among the M hypothesized schemata though one exists. We require that this probability also respect the bound $\Pr(F_{\text{fn2}}) \leqslant \delta/6$. Let SC be an adequate schema. We know an adequate schema exists or the false negative condition could not arise. By Lemma 11, if a random schema is acquired using a randomly sampled problem, the probability that the constructed schema will be the adequate schema SC is at least $p_{\min}\Theta$. Conversely, the probability that the randomly acquired schema is not SC is no greater than $1 - p_{\min}(T_a - \beta)/(\alpha - \beta)$. Repeated randomly acquired schemata are independent events. Thus, the probability that M attempts all fail to acquire SC is no greater than

$$\left(1 - \frac{p_{\min}(T_a - \beta)}{\alpha - \beta}\right)^M$$

which we wish to be bounded by $\delta/6$. Solving for M, this constraint is met by

$$M \geqslant \frac{\alpha - \beta}{p_{\min}(T_a - \beta)} \ln\left(\frac{6}{\delta}\right). \tag{A.7}$$

This establishes the tractability of M which was also required previously.

Finally, there is the possibility that an entertained adequate schema fails its confirmation test and is incorrectly judged to be inadequate. We require $\Pr(F_{\text{fn3}}) \leqslant \delta/6$. A necessary and sufficient F_{fn3} failure condition is that a schema's sample adequacy is measured to be less than the desired adequacy threshold but its true adequacy is at least ε greater than the threshold: $A_s < A_t - \varepsilon$. Again, a sufficient condition for avoiding this form of false positive is inequality (A.4). This is precisely the false positive error condition. There we required a sufficiently large N to bound the error probability of each test by $\delta/(2M)$. Here we need to bound the error probability of each test by $\delta/6$. The algorithm chooses M to be at least 3. Thus, the probability for a false negative and the probability for a false positive are each required to be no greater than $\delta/2$, so the algorithm produces the correct answer with probability at least $1 - \delta$, completing the proof. □

Theorem 15. *Class-wise DB1 permissive planning is tractable provided that problems appear for each problem class with some nonzero lower bound probability* p_{lb}.

Proof (*Sketch*). Again, tractability requires the sample complexity to be independent of the bias space cardinality and polynomial in the other relevant parameters. The DB1 planner adds the complication of C classes whose problems may appear with different probabilities, though none with less than a fixed minimum probability, p_{lb}. Recall that the adequacy of a schema-based planner is by problem class, that problem classes are predefined. Thus, for permissive planning to succeed, a permissive planning algorithm must terminate after consuming only a polynomial number of examples. For each class the planner either outputs Λ, indicating that it believes no adequate schema can be constructed for the class, or it adopts a single schema that probabilistically meets the adequacy requirements. Each such schema is then applied to all problems of its class. The theorem's proof follows directly from Theorem 13. For C classes, we simply start C independent single-schema permissive planning algorithms funneling each class of problems to a separate instance of the algorithm. The planner's behavior must be correct with confidence $1-\delta$. This is assured if each algorithm instance is required to meet the tighter confidence bound $\delta' = \delta/C$. The convergence rate of the slowest converging algorithm instance determines the bound for the composite algorithm. That algorithm instance can expect to see a diluted stream of examples whose dilution factor is no worse than p_{lb}. Thus, the composite sample complexity is no worse than that instance's complexity further inflated by a factor of $1/p_{lb}$. These alterations increase the sample complexity by only polynomial factors. The sample complexity of the composite algorithm is therefore still plynomially bounded, though it is, of course, now sensitive to the number of classes, C, and the lower bound class probability occurrence, p_{lb}. □

Lemma 16. *Let*

$$n_0 = \frac{\alpha-\beta}{p_{\min}(T_a-\beta)} \ln\left(\frac{1}{\eta}\right).$$

Then with probability $1-\eta$ *any solution to* $\mathrm{CT}(n_0)$ *yields a solution to the corresponding single-schema permissive planning task.*

Proof. The derivation of this bound is analogous to the derivation of Eq. (A.7). The probability that n_0 attempts all fail to acquire the assumed adequate schema SC is no greater than

$$\left(1-\frac{p_{\min}(T_a-\beta)}{\alpha-\beta}\right)^{n_0}.$$

We are interested in characterizing n_0 such that

$$\left(1-\frac{p_{\min}(T_a-\beta)}{\alpha-\beta}\right)^{n_0} \leqslant \eta$$

which simplifies to

$$n_0 \geqslant \frac{\alpha-\beta}{p_{\min}(T_a-\beta)} \ln\left(\frac{1}{\eta}\right). \qquad \square$$

Theorem 17. *The VC dimension of the* $\mathrm{CT}(n_0)$ *task is no greater than*

$$\log_2\left(\frac{\alpha-\beta}{p_{\min}(T_a-\beta)}\ln\left(\frac{1}{\eta}\right)\right).$$

Proof. The n_0 elements in the hypothesis concept can shatter a set of at most $\log_2(n_0)$ examples. By definition, the VC dimension is the cardinality of the largest set shattered by the set of hypotheses. □

References

[1] P. Agre and D. Chapman, PENGI: an implementation of a theory of activity, in: *Proceedings AAAI-87*, Seattle, WA (1987) 268–272.

[2] J. Allen and P. Langley, Integrating memory and search in planning, in: *Proceedings Workshop on Innovative Approaches to Planning Scheduling and Control*, San Diego, CA (1990) 301–312.

[3] J.A. Bather, The minimax risk for the two-armed bandit problem, in: *Mathematical Learning Models: Theory and Algorithms* (Springer, Bad Honnef, 1982) 1–11.

[4] R. Bechhofer, A single-sample multiple decision procedure for ranking means of normal populations with known variances, *Ann. Math. Stat.* **25** (1954) 16–39.

[5] S.W. Bennett, Permissive planning: a machine learning approach to planning in complex real-world domains, Tech. Rept., Electrical & Computer Engineering Department, University of Illinois, Urbana, IL (1993).

[6] S.W. Bennett and G.F. DeJong, Comparing stochastic planning to the acquisition of increasingly permissive plans for complex uncertain domains, in: *Proceedings Eighth International Conference on Machine Learning*, Evanston, IL (1991) 586–590.

[7] S. Benson, Inductive learning of reactive action models, in: *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA (1995) 47–54.

[8] H. Berenji, Fuzzy logic controllers, in: E.R. Yager and L. Zadeh, eds., *An Introduction to Fuzzy Logic Applications in Intelligent Systems* (Kluwer Academic Publishers, Dordrecht, 1990) 69–96.

[9] D.A. Berry and B. Fristedt, Bandit problems: sequential alloction of experiments, in: D. Cox et al., eds., *Monographs on Statistics and Applied Probability* (Chapman and Hall, London, 1985).

[10] A. Blumer et al., Learnability and the Vapnik–Chervoninkis dimension, *J. ACM* **36** (1990) 929–965.

[11] C. Boutilier, Abduction to plausible causes: an event-based model of belief update, *Artif. Intell.* **83** (1996) 143–166.

[12] R. Brooks, A robust layered control system for a mobile robot, *IEEE J. Rob. Automat.* **2** (1986).

[13] J. Carbonell and Y. Gil, Learning by experimentation: the operator refinement method, in: Y. Kodratoff and R. Michalski, eds., *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1990) 191–213.

[14] D. Chapman, Planning for conjunctive goals, *Artif. Intell.* **32** (1987) 333–378.

[15] P. Cheeseman, Probabilistic vs. fuzzy reasoning, in: L.N. Kanal and J.F. Lemmer, eds., *Uncertainty in Artificial Intelligence* (North-Holland, Amsterdam, 1986) 85–102.

[16] S. Chien, M. Gervasio and G.F. DeJong, Learning to integrate reactiveity and deliberation in uncertain planning and scheduling problems, in: *Proceedings AAAI Spring Symposium Series: Practical Approaches to Scheduling and Planning*, Palo Alto, CA (1992) 117–121.

[17] G. Collins and L. Pryor, Achieving the functionality of filter conditions in a partial order planner, in: *Proceedings AAAI-92*, San Jose, CA (1992) 375–380.

[18] E. Davis, *Representations of Commonsense Knowledge* (Morgan Kaufmann, San Mateo, CA, 1990).

[19] T.A.B. Dean and M. Lejter, Planning and active perception, in: *Proceedings Workshop on Innovative Approaches to Planning Scheduling and Control* (1990) 271–276.

[20] G.F. DeJong, ed., *Investigating Explanation-Based Learning* (Kluwer Academic Publishers, Boston, MA, 1993).

[21] G.F. DeJong, Learning to plan in continuous domains, *Artif. Intell.* **65** (1994) 71–141.

[22] G.F. DeJong and R. Mooney, Explanation-based learning: an alternative view, *Mach. Learn.* **1** (1986) 145–176.
[23] A. del Val and Y. Shoham, Deriving properties of belief update from theories of action, in: *Proceedings AAAI-92*, San Jose, CA (1992) 584–589.
[24] D. Driankov, H. Hellendoorn and M. Reinfrank, *An Introduction to Fuzzy Control* (Springer, Berlin, 1993).
[25] M. Erdmann, Using backprojections for fine motion planning with uncertainty, *Int. J. Rob. Res.* **5** (1986) 19–45.
[26] P.E. Fricdland, Knowledge-based experiment design in molecular genetics, Tech. Rept., Computer Science Department, Stanford University, Stanford, CA (1979).
[27] M.R. Genesereth and N.J. Nilsson, *Logical Foundations of Artificial Intelligence* (Morgan Kaufmann, Palo Alto, CA, 1986).
[28] Y. Gil, Learning by experimentation: incremental refinement of incomplete planning domains, in: *Proceedings Eleventh International Conference on Machine Learning*, New Brunswick, NJ (1994) 87–95.
[29] D. Gordon, An enhancer for reactive plans, in: *Proceedings Eighth International Conference on Machine Learning*, Ithaca, NY (1991) 505–508.
[30] J. Gratch and G.F. DeJong, A decision-theoretic approach to adaptive problem solving, *Artif. Intell.* (to appear).
[31] P. Haddawy and S. Hanks, Representations for decision-theoretic planning, in: *Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1992) 71–82.
[32] P. Haddawy and M. Suwandi, Decision-theoretic refinement planning using inheritance abstractions, in: *Proceedings Second International Conference on Artificial Intelligence Planning Systems*, Chicago, IL (1994) 266–271.
[33] K. Hammond, T. Converse and C. Martin, Integrating planning and acting in a case-based framework, in: *Proceedings AAAI-90*, Boston, MA (1990) 292–297.
[34] S.A. Hutchinson and A.C. Kak, Spar: a planner that satisfies operational and geometric goals in uncertain environments, *AI Magazine* **11** (1990) 30–61.
[35] C.M. Kadie, Continuous conceptual set covering: learning robot operators from examples, in: *Proceedings Eighth International Conference on Machine Learning*, Ithaca, NY (1991) 615–619.
[36] L.P. Kaelbling, An architecture for intelligent reactive systems, in: *Proceedings Workshop on Reasoning About Actions and Plans*, Timberline, OR (1986) 395–410.
[37] L. Kaelbling, Learning to achieve goals, in: *Proceedings IJCAI-93*, Chambery (1993) 1094–1098.
[38] S. Kambhampati, On the utility of systematicity: understanding tradeoffs between redundancy and commitment in partial-order planning, in: *Proceedings IJCAI-93*, Chambery (1993) 1380–1385.
[39] S. Kambhampati and J. Hendler, A validation-structure-based theory of plan modification and reuse, *Artif. Intell.* **55** (1992) 193–258.
[40] H. Katsuno and A. Mendelzon, On the difference between updating a knowledge database and revising it, in: *Proceedings Second International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1991) 387–394.
[41] J. Laird, A. Newell and P. Rosenbloom, SOAR: an architecture for general intelligence, *Artif. Intell.* **33** (1987) 1–64.
[42] J.-C. Latombe, *Robot Motion Planning* (Kluwer Academic Publishers, Boston, MA, 1991).
[43] T. Lozano-Perez, M.T. Mason and R.H. Taylor, Automatic synthesis of fine-motion strategies for robots, *Int. J. Rob. Res.* **3** (1984) 3–24.
[44] P. Maes and R.A. Brooks, Learning to coordinate behaviors, in: *Proceedings AAAI-90*, Boston, MA (1990) 796–802.
[45] D. McAllester, Systematic nonlinear planning, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 634–639.
[46] J. McCarthy, Circumscription—a form of non-monotonic reasoning, *Artif. Intell.* **13** (1980) 27–39.
[47] D. McDermott, Robot planning, *AI Magazine* **13** (1992) 55–79.
[48] R.S. Michalski, Understanding the nature of learning: issues and research directions, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 3–26.

[49] D.P. Miller et al., Reactive navigation through rough terrain: experimental results, in: *Proceedings AAAI-92*, San Jose, CA (1992) 823–828.
[50] S. Minton, *Learning Search Control Knowledge: An Explanation-Based Approach* (Kluwer Academic Publishers, Norwell, MA, 1988).
[51] S. Minton, J. Bresina and M. Drummond, Commitment strategies in planning: a comparative analysis, in: *Proceedings AAAI-91*, Anaheim, CA (1991).
[52] S. Minton et al., Acquiring effective search control rules: explanation-based learning in the PRODIGY system, in: *Proceedings Fourth International Conference on Machine Learning*, Irvine, CA (1987) 122–133.
[53] T.M. Mitchell, Generalization as search, *Artif. Intell.* **18** (1982) 203–226.
[54] T.M. Mitchell, Becoming increasingly reactive, in: *Proceedings AAAI-90*, Boston, MA (1990) 1051–1058.
[55] T. Mitchell, R. Keller and S. Kedar-Cabelli, Explanation-based generalization: a unifying view, *Mach. Learn.* **1** (1986) 47–80.
[56] R.J. Mooney, *A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding* (Pitman, London, 1990).
[57] R.J. Mooney and S.W. Bennett, A domain independent explanation-based generalizer, in: *Proceedings AAAI-86*, Philadelphia, PA (1986) 551–555.
[58] A. Moore, Variable resolution dynamic programming: efficiently learning action maps in multivariate real-valued state-spaces, in: *Proceedings of the Eighth International Conference on Machine Learning*, Evanston, IL (1991) 333–337.
[59] N.J. Nilsson, *Principles of Artificial Intelligence* (Tioga, Palo Alto, CA, 1980).
[60] N. Nilsson, Teleo-reactive programs for agent control, *J. Artif. Intell. Res.* **1** (1994) 139–158.
[61] B. Nordhausen and P. Langley, Towards an integrated discovery system, in: *Proceedings IJCAI-87*, Milan (1987) 198–200.
[62] B. Nordhausen and P. Langley, An integrated framework for empirical discovery, *Mach. Learn.* **12** (1993) 17–47.
[63] D. Olawsky and M. Gini, Deferred planning and sensor use, in: *Proceedings Workshop on Innovative Approaches to Planning, Scheduling and Control*, San Diego, CA (1990) 166–174.
[64] D. Ourston and R. Mooney, Changing the rules: a comprehensive approach to theory refinement, in: *Proceedings AAAI-90*, Boston, MA (1990) 815–820.
[65] M.J. Pazzani, Integrated learning with incorrect and incomplete theories, in: *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI (1988) 291–297.
[66] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (Morgan Kaufmann, Los Altos, CA, 1988).
[67] E. Pednault, Generalizing nonlinear planning to handle complex goals and actions with context-dependent effects, in: *Proceedings IJCAI-91*, Sydney (1991) 240–245.
[68] J. Penberthy and D. Weld, UCPOP: a sound, complete, partial order planner for ADL, in: *Proceedings Third International Conference on the Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1992) 103–114.
[69] M. Peot and D. Smith, Conditional non-linear planning, in: *Proceedings First International Conference on Artificial Intelligence Planning Systems* (1992) 189–197.
[70] S.A. Rajamoney, A computational approach to theory revision, in: J. Shrager and P. Langley, eds., *Computational Models of Scientific Discovery and Theory Formation* (Lawrence Earlbaum, Hillsdale, NJ, 1990).
[71] S.A. Rajamoney, The design of discrimination experiments, *Mach. Learn.* **12** (1993) 185–203.
[72] S.A. Rajamoney and G.F. DeJong, Active explanation reduction: an approach to the multiple explanations problem, in: *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI (1988) 242–255.
[73] D. Roth, On the hardness of approximate reasoning, *Artif. Intell.* **82** (1996) 273–302.
[74] S. Russell and E. Wefald, *Do the Right Thing* (MIT Press, Cambridge, MA, 1991).
[75] E.D. Sacerdoti, The nonlinear nature of plans, in: *Proceedings IJCAI-75*, Tblisi (1975) 206–214.
[76] M. Schoppers, Universal plans for reactive robots in unpredictable environments, in: *Proceedings IJCAI-87*, Milan (1987) 1039–1046.

[77] A. Segre, *Machine Learning of Robot Assembly Plans* (Kluwer Academic Publishers, Norwell, MA, 1988).
[78] J.W. Shavlik, *Extending Explanation-Based Learning by Generalizing the Structure of Explanations* (Pitman, London, 1990).
[79] R. Sutton, Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, in: *Proceedings of the Seventh International Conference on Machine Learning*, Austin, TX (1990) 216–224.
[80] R. Sutton, The challenge of reinforcement learning, *Mach. Learn.* **8** (1992) 225–227.
[81] A. Tate, Generating project networks, in: J. Hendler, A. Tate and J. Allen, eds., *Readings in Planning* (Morgan Kaufmann, Los Altos, CA, 1990) 291–296.
[82] L. Valiant, A theory of the learnable, *Commun. ACM* **27** (1984) 1134–1142.
[83] X. Wang, Learning by observation and practice: an incremental approach for planning operator acquisition, in: *Proceedings Twelfth International Conference on Machine Learning*, Tahoe City, CA (1995) 549–557.
[84] C. Watkins and P. Dayan, Q-learning, *Mach. Learn.* **8** (1992) 279–292.
[85] D. Weld, An introduction to least commitment planning, *AI Magazine* **15** (1994) 27–61.
[86] D.E. Wilkins, *Practical Planning* (Morgan Kaufmann, San Mateo, CA, 1988).
[87] L. Zadeh, Commonsense and fuzzy logic, in: *The Knowledge Frontier: Essays in the Representation of Knowledge* (Springer, New York, 1987) 103–136.
[88] M. Zweben, E. Davis, B. Daun, E. Drascher, M. Deale and M. Eskey, Learning to improve constraint-based scheduling, *Artif. Intell.* **58** (1992) 271–296.