Machine Learning, 10, 279-309 (1993)
© 1993 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Opportunism and Learning

KRISTIAN HAMMOND, TIMOTHY CONVERSE AND MITCHELL MARKS AI@TARTARUS.UCHICAGO.EDU
The University of Chicago, Department of Computer Science, Artificial Intelligence Laboratory, 1100 East 58th Street, Chicago, IL 60637

COLLEEN M. SEIFERT SEIFERT@UM.CC.UMICH.EDU
Department of Psychology, University of Michigan, 330 Packard Road, Ann Arbor, MI 48104

Abstract. There is a tension in the world between complexity and simplicity. On one hand, we are faced with a richness of environment and experience that is at times overwhelming. On the other, we seem to be able to cope and even thrive within this complexity through the use of simple scripts, stereotypical judgements, and habitual behaviors. In order to function in the world, we have idealized and simplified it in a way that makes reasoning about it more tractable. As a group and as individuals, human agents search for and create islands of simplicity and stability within a sea of complexity and change.

In this article, we will discuss an approach based on the case-based reasoning paradigm that attempts to resolve this tension. This *agency* approach embraces, rather than avoids, this paradox of the apparent complexity of the world and the overall simplicity of our methods for dealing with it. It accomplishes this by treating the behavior of intelligent agents as an ongoing attempt to discover, create, and maintain the stability that is necessary for the production of actions that satisfy our goals.

Keywords: Case-based planning, opportunism, stabilization of environments, control of execution

1. Introduction

There is a tension in the world between complexity and simplicity. On one hand, we are faced with a richness of environment and experience that is at times overwhelming. On the other, we seem to be able to cope and even thrive within this complexity through the use of simple scripts, stereotypical judgements, and habitual behaviors. In order to function in the world, we have idealized and simplified it in a way that makes reasoning about it more tractable. As a group and as individuals, human agents search for and create islands of simplicity and stability within a sea of complexity and change.

In this article, we will discuss an approach based on the case-based paradigm that attempts to resolve this tension. This *agency* approach embraces, rather than avoids, this paradox of the apparent complexity of the world and the overall simplicity of our methods for dealing with it. It accomplishes this by treating the behavior of intelligent agents as an ongoing attempt to discover, create, and maintain the stability that is necessary for the production of actions that satisfy our goals.

Our framework for the study of agency consists of four parts:

- Case-based planning
- Learning from failure
- Learning from execution-time opportunity
- Stabilizing the environment through enforcement

Each of these components plays a role in the overall effort to establish a functional correspondence between the internal world of an agent and the complex environments in which it must function. The starting point, **case-based planning**, provides the structure by producing a framework for learning and reusing standard plans that constitute fixed reference points in the world. The ability to **learn from failure** allows a planner to incrementally search the simplified space of problems that *actually* arise during execution as opposed to the far more complex space of those problems that can *theoretically* arise. Likewise, the ability to recognize and **learn from execution-time opportunity** allows a planner to construct and save plans for the conjuncts of goals that actually arise in a problem space, while avoiding the problems of exponential search of the entire space of possible conjuncts. Finally, the ability to **stabilize an environment** with respect to an agent's plans and then enforce the resulting states through policy decisions (McDermott, 1978) allows the creation of niches in which the complexity of the world is reduced, thus making it easier to reason about and act within them.

Together, these four different aspects of planning and execution provide a mechanism for learning from interactions with the world. The theory of agency will be successful only to the extent that it provides a mechanism for learning from experience; in other words, *planning is only possible when supported by learning.*

In the sections that follow, we will first present a taxonomy of the different kinds of complexity that an intelligent agent must confront in the world, and then discuss how each of these four components addresses this complexity. We will then discuss particular aspects of case-based planning, learning from failure, and learning from opportunity. This discussion will concentrate on two systems: CHEF, a case-based planner, and TRUCKER, an opportunistic planner.

2. Problems in planning: The need for learning

Most AI work on problems of autonomous agency has been within the classical approach to planning and problem-solving, where a fairly strict separation between planning and execution was often assumed. In this view, theories of planning emphasized exhaustive preplanning in order to guarantee that a correct plan would be found (if one existed). Planners in this paradigm required certain constraints (rising out of the *closed-world* assumption and *STRIPS*):

- The world will be stable; it will behave as projected.
- Time consumed in planning is independent of the time that can be devoted to execution, so that the efficiency of the planner has no side-effects on the feasibility of the constructed plan.
- The information available to the planner is complete, and execution will be flawless.
- Any initially correct plan will remain correct and can in fact be carried out.

As we begin to relax these assumptions, new issues arise that planners operating under these constraints were able to avoid. It is the problems that appear in the absence of these restrictive assumptions that motivate departure from the classical view of planning and action.

- **The Immediate Complexity Issue**
  If a planner searches for a correct and safe plan by projecting forward the effects of early steps to compare with preconditions of later steps, goals, and preservation conditions, the computational complexity can rise to a high order. Chapman (1985) has shown the problem to be NP-hard, and undecidable in the worst case. Dean et al. (1987) have also shown, in the construction of FORBIN, that average case behavior is also far too time-consuming to be practical.
- **The Overall Complexity Issue**
  As a planner interacts with the world, it is confronted with a stream of goals, or sets of conjunctive goals, rather than independent problems. If these are all treated singly, independently, the total planning and execution effort would be at least the sum of the separate costs (possibly worse, due to unfavorable interactions).
- **The Planning/Execution Crowding Issue**
  If planning and execution are to be carried out by the same agent, essentially without parallelism, then time consumed in planning can deplete the time available for execution. This can create a situation where some series of actions would be a correct response to the goals and the world state, and could be feasibly executed within the total time available, but become infeasible within the time left after planning.
- **The Costly Information Issue**
  The planner cannot count on having complete information about a domain, either its underlying causal "physics" or its current state. To minimize the effects of this information shortage (namely, inefficient plan construction and inaccurate plans), the planner should have information-gathering as a background goal. But understanding the world correctly, and storing new information in a usable form, can be expensive operations. They must be carried out in the normal course of execution if they are to be a help and not a hindrance.
- **The Execution-time Failure Issue**
  A plan that looked correct when it was constructed may turn out to be incorrect during execution. This may be because conditions in the world have changed in the meantime, invalidating the preconditions of a plan step, or because the plan was incorrect in the first place, based on incorrect assumptions in the planner's limited world-knowledge. To cope with this, a planner must have some facility for replanning, recovery, and repair.
- **The Execution-time Opportunism Issue**
  A corollary to the Execution-time Failure issue is the Execution-time Opportunism issue. Because the world does not necessarily match the planner's projection of it, it is often the case that oportunities to satisfy goals are missed at planning time and must be noticed and exploited at execution time.

These issues are at the core of planning and activity, but what, one may ask, do they have to do with learning? The answer to this is simple: **learning from planning and execution is the solution to the problems posed by these issues.**

In general, we propose that the only way to deal with the intractability of exhaustive reasoning within complex domains is to integrate the tasks of planning and learning into a single-agent architecture. Rather than suppose that it is possible to construct functional plans from a static knowledge base, we suggest that a dynamic case base of plans and their effects be produced and used incrementally. In this way, a planner can improve itself over time through the understanding of its own success and failure of actions taken within a given domain.

The existence of the learner makes the incremental improvement of the planner possible, while the control of the planner gives the learner the guidance it needs to construct only those concepts that are useful to the agent.

As we stated in the last section, there are four learning issues that must be addressed to deal with these problems:

- Case-based planning, to support the reuse of plans and deal with the problems of plan complexity and execution crowding.
- Learning from failure, to avoid the problem of execution-time failure.
- Learning from opportunity, to exploit the optimizations learned from execution-time opportunities.
- Stabilization of the environment through enforcement (Hammond, 1991), which allows greater reuse of existing plans and avoids much of the cost of information gathering.

Planning and learning are one and the same: To ignore this is to ignore the complexity of the former and the functionality of the latter. Planning can only be made possible when done in the context of a system that incrementally adapts itself so as to learn about the plans, failures, and opportunities that make up the structure of a domain. Learning is only feasible when knowledge plays a functional role in a performance system, such as a planner that is able to inform the learner when that knowledge is faulty.

In the sections that follow, we will move back and forth between discussion of planning and learning. While we do this, it is important for the reader to understand that there is no division between the two, but rather, that they are fully integrated, and that each informs, guides, and supports the other.

3. Case-based planning

One technological method that addresses the problems of overall complexity, execution-time failure, and costly information comes out of emerging work in case-based reasoning (Alterman, 1986; Bareiss, 1989; Carbonell, 1981; Hammond, 1989; Kolodner, 1984; Kolodner & Simpson, 1989; Martin, 1990; Owens, 1990; Schank, 1982). This work has suggested an approach to problem-solving that may be a tractable alternative to the more traditional rule-based approaches (Fikes & Nilsson, 1971; Newell & Simon, 1972; Sacerdoti, 1975; Stefik, 1981). The idea of case-based reasoning also seems to correspond to the intuition that people make use of past experience in planning for new goals (Byrne, 1977).

Case-based planning, a special case of case-based reasoning, proposes that the way to deal with the combinatorics of planning and projection is to let experience tell the planner when and where things work and do not work: rather than replanning, a planner should reuse past successful plans, rather than projecting the effects of actions into the future, a planner should recall what the outcomes were in the past; rather than simulating a plan to identify problematic interactions, a planner should recall and then avoid those that have cropped up before.

The methodology of case-based planning is aimed at tackling different aspects of planning that have been traditional bottlenecks (plan generation, projection, optimization) by looking for way to amortize the cost of system's reasoning efforts through storing and reusing the results of the problem solving. The core notion of case-based planning is that learning through planning, execution, and recovery from failure is essential to the development of domain theory (a case library) that covers the space of plans for actual goals while allowing the planner to avoid those execution problems that arise when trying to run those plans.

The results of this methodology are systems with the following features:

- New plans are built from old plans.
- Projection is replaced with anticipation based on experience rather than simulation.
- Plans are indexed and retrieved by reference to the goals that they satisfy, the execution problems that they avoid, and the features in the world that have been associated with them in the past.
- Knowledge of the effects of individual operators is not used in planning per se.
- Execution failures lead to learning.
- Execution opportunities lead to learning.

3.1. A framework for case-based planning

In case-based planning, a plan library reflects the structure of a domain (e.g., the goals that tend to arise in conjunction, the interactions between steps that tend to occur, and the plans that resolve them), and is built up incrementally through learning.

This framework suggests seven case-based planning processes:

- An ANTICIPATOR that predicts execution problems[1] on the basis of the failures that have been seen by the planner in the past.
- A RETRIEVER that searches a plan memory for a plan that satisfies as many of the current goals as possible while avoiding any execution problems that the ANTICIPATOR has predicted.
- A MODIFIER that alters the plan found by the RETRIEVER to achieve any goals from the input that it does not satisfy.
- A PROJECTOR that uses cases indexed by plans rather than goals to predict outcomes.
- A STORER that places new plans in memory, indexed by the goals that they satisfy and the execution problems that they avoid.
- A REPAIRER that is called if a plan fails. It is here that we argue that causal knowledge is applied—if it is applied at all.
- An ASSIGNER that uses the causal explanation built during repair to determine the features that will predict this failure in the future. This knowledge is used to index the failure for later access by the ANTICIPATOR.

The flow of control through modules is simple (figure 1). Goals are handed to the planner through the ANTICIPATOR, which predicts any execution problems that might occur as

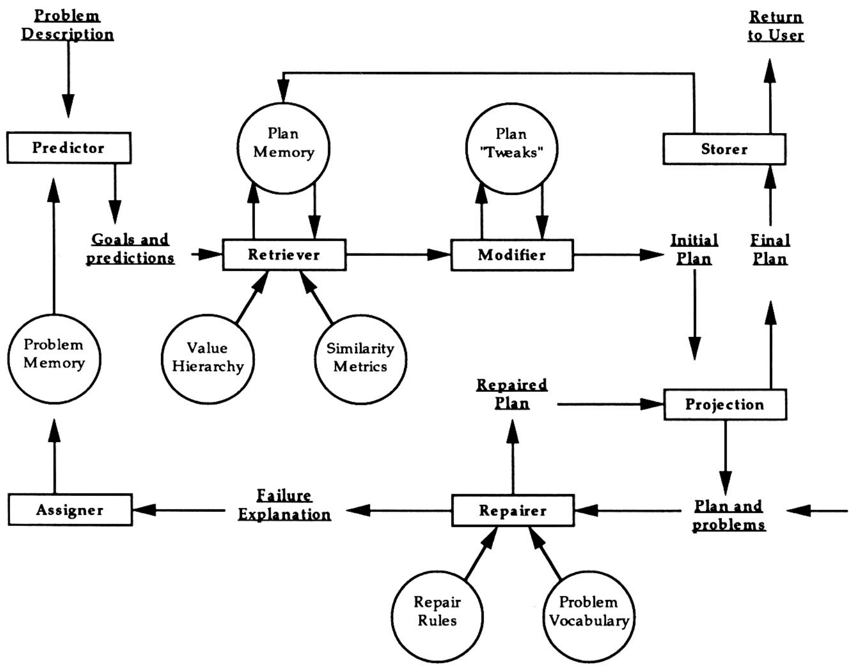

Figure 1. Flow of control through a case-based planner.

a result of planning for them. If a problem is predicted, a goal to avoid it is added to the set of goals to be planned for. Once this is done, the goals are handed to the RETRIEVER, which searches for a plan that satisfies as many goals as possible, including any goals to avoid the execution problems predicted by the ANTICIPATOR. In order to do this, the RETRIEVER makes use of a memory of plans indexed by the goals they satisfy and the execution problems they avoid.

Once a baseline plan is found, the MODIFIER alters it to satisfy any goals that are not already satisfied. The MODIFIER does this using a set of modification rules that are indexed by the goal to be added and the type of plan being altered. It also uses a set of critics for problematic items in the domain.

Once a plan reaches this stage, it is run. If the plan runs without fail, it is handed directly to the STORER, which indexes the plan in memory by the goals that it satisfies. If these are failures, the plan is handed to the REPAIRER for repair. The REPAIRER builds a causal description of why the plan has failed and then uses that description to find the repair strategies that will alter the situation and fix the fault. After the plan has been repaired, it is placed in memory by the STORER, which indexes the plan by the goals it satisfies and the execution problems it now avoids.

Because an execution problem has to be anticipated before it can be planned for, however, the case-based planner has to do more than just store plans by the fact they solve particular problems. It also has to build up a knowledge base that can be used by the ANTICIPATOR to infer problems in advance on the basis of the features in the world that predict them. The module that does this is the ASSIGNER. The ASSIGNER is called whenever a failure occurs, and it builds links between the features that have caused the failure and a memory of the failure itself. These links are later used by the ANTICIPATOR to predict failures so that the plan can be found that avoids them.

These seven modules define case-based planning. The RETRIEVER, MODIFIER, and STORER make up the central planning loop that allows old plans to be modified in service of new goals. The PROJECTOR and REPAIRER are required to identify and repair plans that fail. The ASSIGNER and ANTICIPATOR provide the learning and application modules that allow the planner to avoid making mistakes that it has already encountered. This organization defines an *anticipate and avoid* planner that can predict execution problems before they occur, apply a wide variety of repairs to faulty plans that it does create, and store plans that deal with problems so that they can be used when those problems arise.

3.2. CHEF

One example of this type of system is CHEF, a case-based planner in the domain of Szechwan cooking (Hammond, 1989). Its input is a set of goals for different tastes, textures, ingredients, and types of dishes, and its output is a single recipe that satisfies all of its goals. Its algorithm is to find a past plan that satisfies as many of the most important goals as possible and then modify that plan to satisfy the other goals as well.

Before searching for a plan to modify, CHEF examines the goals in its input and predicts any failures that might rise out of the interactions between the plans for satisfying them. If a failure is predicted, CHEF adds a goal of avoiding the failure to its list of goals to satisfy, and this new goal is also used to search for a plan. For example, if it predicts that stir frying chicken with snow peas will lead to soggy snow peas because the chicken will sweat liquid into the pan, it searches for a stir fry plan that avoids the problem of vegetables getting soggy when cooked with meats. In doing so, it finds a past plan for beef and broccoli that solves this problem by stir frying the vegetable and meat separately. The important similarity between the current situation and the one for which the past plan was built is that the same problem rises out of the interaction between the planner's goals, although the specific goals themselves are different.

There are two important points about the anticipation of problems by case-based planners. The first is that CHEF indexes its plans by both the goals that they satisfy and the execution problems that they avoid. This means that the anticipation of a problem can be used to find a plan in memory that avoids it, while also satisfying the goals that the planner has been given. The second is that the ANTICIPATOR predicts execution problems that will rise out of the current goals *before* any other planning is done. This means that the prediction of a problem is made before any planning is done. So CHEF is able to anticipate and then avoid problems before they actually arise.

The result of this strategy is that CHEF incrementally improves its ability to predict problems and create plans in response to its own failures. Its attempts at satisfying its own goals, successful or not, result in an improvement in its ability to do so in the future.

3.3. Learning from planning

Case-based planning depends on learning. Learning new plans allows a planner to shape its understanding of the domain to those goals that typically tend to co-occur within it. Learning specific repairs, such as stir-frying beef and broccoli at separate times, allows a planner to avoid errors that it has experienced previously. Learning the features that predict such errors further allows a planner to anticipate and thus avoid the problems by recalling debugged plans or reapplying newly learned repairs.

Without learning, a case-based system is dependent on hand-tailored cases and modification rules if it is to avoid the complexity of search through the space of *potential* plans. With learning, however, a case-based planner can construct and then search the smaller libraries associated with the plans that cover the *actual* problem space it confronts.

4. Opportunism and memory

Despite addressing some of the problems of traditional planning, case-based planning still shares some of its basic assumptions. The most central of these is the view of action as decomposable into separate planning and execution phases, where the execution phase is in some sense faithfully carrying out the dictates of the computation object called the plan. There are tasks and domains where this decomposition is still tenable. For example, ROENTGEN, a planner in the domain of radiation therapy, uses these separate phases; in fact, the domain has distinct job titles for the planners and executors. The desirability of this separation is due both to the high cost of real-world failure and to the existence of reliable simulation programs against which plans can be tested. Yet in recent years there has been a realization in the planning community that the domains that will support such a planning–execution separation are much rarer than had previously been supposed.

At a minimum, exhaustive pre-planning for a set of goals seems to require that an agent be aware of all of its goals at plan time and that it have complete knowledge of the physics and other agents in his environment. Unfortunately, this is simply not true of most interesting domains or situations that an agent must confront. It is this problem that led us to believe that execution-time failure was inevitable and pushed us in the direction of the work on coping with and learning from failure that we discussed in the last section. But failure is only part of the story. The other part is the issue of execution-time opportunity. Just as a system that cannot fully predict the future must contend with those futures in which plans fail, it must also deal with those futures in which they succeed in unanticipated ways. In order to be effective, it must also cope with and learn from opportunities that were not predicted in much the same way that it deals with failures.

The difference between the approach we discuss here and the one taken in CHEF lies in the relationship between expectations and plans. In CHEF, we studied *expectation failures* (Schank, 1982) that corresponded to actual *plan failures*. There we looked at the utility of learning from failures by changing the planner's knowledge of plans and repairs as well as its expectations about when previously experienced failure would tend to recur.

In this section, we will look at the issue of learning from expectation failures that are failures to anticipate *planning opportunities*. In CHEF, we argued that a planner has to respond to failure by repairing its current plan and by repairing the knowledge base (its expectations as to the results of its actions) that allowed it to create the plan. In this section, we argue that execution-time opportunities can be responded to in a similar way: the planner should exploit the current opportunity *and* change its expectations so as to properly anticipate and exploit the opportunity in the future.

The approach to the problem of execution-time opportunism that we have used in both TRUCKER and RUNNER uses episodic memory to organize, recognize, and exploit opportunities. Briefly, the algorithm includes the following features:

- Goals that cannot be fit into a current ongoing plan are considered blocked, and work on them is suspended (Schank & Abelson, 1977).
- Suspended goals are associated with elements of episodic memory that can be related to potential opportunities.[2]
- These same general memory structures are then used to interpret events in the world so that the planner can make routine execution-time decisions.
- As elements of memory are activated by conditions in the world, the goals associated with them are also activated and integrated into the current processing queue.

In this way, suspended goals are brought to the planner's attention when conditions change so that the goals can be satisfied.

Because the recognition of opportunities depends on the nature of its episodic memory structures, we call the overall algorithm presented here *opportunistic memory.*

Plan failures are indicators of when a system needs to learn about negative interactions that might be avoided. Likewise, execution-time opportunities are indicators of when a system needs to learn about positive interactions that might be exploited. Here again, planning, execution, and learning are interwoven into a system that provides itself with feedback about its own knowledge base through the active application of that knowledge base in creating and running plans.

4.1. Opportunistic planning

Our approach to opportunism builds on two views of opportunism in planning—that of Hayes-Roth and Hayes-Roth (1979), and that of Birnbaum and Collins (1984).

Hayes-Roth and Hayes-Roth presented the view that a planner should be able to shift between planning strategies on the basis of perceived opportunities, even when those opportunities are unanticipated. Thier model, which they called *opportunistic planning*, consisted of a blackboard architecture (Lesser et al., 1975) and planning *specialists* that captured planning information at many different levels of abstraction. These specialists included domain-level plan developers (e.g., specialists that know about routes, stores, or conditions for specific plans) as well as more strategic operators (e.g., specialists that would look for clusters of goals and goals with similar preconditions). The planner could jump between strategies as different specialists "noticed" that their activation conditions were present. For example, in scheduling a set of errands, a specialist with knowledge of clustering of errands by location could be invoked while another specialist was scheduling them by goal priority. In this way, the planner could respond to opportunities noticed at planning time.

Unfortunately, this view of opportunism includes no model of execution. As with many planners, all planning is done in the absence of the ability to execute the plan, and thus respond to the effects of execution. It is a model of opportunism at *planning time* rather than at *execution time*, and as such fails to capture the behavior we are interested in modeling.

More recently, Birnbaum and Collins (1984) presented a view of opportunism that does include a role for execution. Under their model, goals are viewed as independent processing entities that have their own inferential power (e.g., demons). When a goal is suspended because of resource constraints, it continues to examine the ongoing flow of objects and events that pass by the agent. If circumstances that would allow for the satisfaction of the goal arise, a demon associated with the goal recognizes them and projects a plan into the current action agenda. They present a simple yet compelling example of the behavior that interests them in a description of an agent trying to obtain both food and water in the wild. In their example, the agent suspends the goal to find water while trying to satisfy the goal to find food. While searching for food, however, the agent jumps over a stream and is able to recognize that the stream affords an opportunity to satisfy a suspended goal.

Birnbaum and Collins argue that this is managed by giving the suspended goal the ability to examine the current situation and initiate inference. They argue that this must be the case, since there can be no way to decide, at the time of suspending a goal, the exact conditions under which it should be activated. Any planner that has to store and then activate suspended goals will miss opportunities that it could not anticipate.

Birnbaum (1986) has argued further that indexing of suspended goals by descriptions of the conditions that allow their satisfaction is an unworkable approach. Not only will the indices be too complex, but the overall system will have to constantly compare the current state of the world to the features used to index suspended goals in what amounts to a memory of unsatisfied tasks. As Birnbaum notes, this is hardly opportunism. However, Birnbaum's arguments against indexing apply to models that separate representations of suspended goals from the planner's memory and understanding of the world.

We share Birnbaum and Collins' approach to explaining complex opportunistic behavior in terms of suspended goals. We suggest, however, that this behavior does not result from goal demons constantly monitoring the world, but instead, that indexing suspended goals provides a better account of recognizing opportunities during plan execution. This is possible because there is no distinction in our model between types of memory—suspended vs. achieved goals. Instead, there is only one memory base that is used to understand the world and to store suspended goals along with any other information.

Thus, the action of recognition is the same as the act of indexing: The processing done in attempting to recognize the current planning environment is the same processing that will activate suspended goals waiting under appropriate indices in memory. The advantage to this approach is that prepared indices make the recognition of later opportunities easy and effortless, because they can be formulated so as to utilize information that results from normal, default inferencing about the world. "Smarter" indexing can provide a solution to the problem of recognizing future opportunities for past goals by anticipating the circumstances that comprise an opportunity at the time of the impasse.

4.2. An example of opportunism

Before we go any further, it is important to understand an example of the type of behavior we want to capture. Although this example is couched in terms of a story, we are interested in modeling the planning behavior described, not in understanding the text.

> On making breakfast for himself in the morning, John realized that he was out of orange juice. Because he was late for work he had no time to do anything about it.
>
> On his way home from work, John noticed that he was passing a Seven-Eleven and recalled that he needed orange juice. Having time, he stopped and picked up a quart and then continued home.

There are a number of interesting aspects to this example. First of all, the planner is confronted with new goals during execution as well during planning. This indicates complete preplanning is impossible. Second, the planner is able to suspend planning for a goal before deciding exactly how to satisfy it. In effect, he is able to say "I don't have all the resources needed to completely integrate a plan for this goal into my current agenda." Using Schank's vocabulary, we call this the ability to *suspend* a goal (Schank & Abelson, 1977). And third, although the goal is suspended, the planner is able to recognize the conditions that potentially lead to its satisfaction.[3] The final element to this example is more subtle: in order to decide to suspend planning for the goal to possess orange juice, John has to do some reasoning about what a plan for that goal is lacking—lack of time to go to the store. As a result, he has a clear idea, at planning time, as to what an execution-time opportunity would look like.

In this example, our opportunistic memory algorithm translates into the following:

- John's goal to possess orange juice is blocked by lack of time to run the default plan. He decides, on the basis of the preconditions on his plan to possess orange juice, that being at a store would constitute an opportunity to get the orange juice. As a result, he links the suspended goal to the condition of being near a store.
- While coming home, he sees and recognizes a Seven-Eleven. This activates the goal to obtain orange juice that he associated with this condition earlier in the day.
- He then tests the preconditions of the plan and merges it into his current agenda.

In our example, having money, being at a grocery store, and having time are all preconditions for buying orange juice. But there is a difference between them, in that having money is a normative condition and as such does not constitute an opportunity, while being near a store is a non-normative precondition and as such does constitute an opportunity.

The notion of *normative* here is based on McDermott's idea of *policies* (McDermott, 1978) that maintain states in the world by generating goals when those states are violated and our own notion of *enforcement* (Hammond, 1991). If a state is associated with a policy (and as such is always maintained), then it can be considered a normative state. Likewise, if a state has been established and is under enforcement, it too can be considered normative. Always having money on hand is the result of a policy decision that causes a goal to get more money whenever the local supply drops below a certain threshold. As a result, the state of having money can be assumed by a planner that maintains such a policy. Likewise, enforcing a state such as keeping the coffee maker plugged in allows a planner to assume this state as well. In general, then, the world can be roughly divided into those states that the planner enforces, permanently stable states, and those that are in flux.

First, the planner suspends the blocked goals by associating them with the elements of memory that describe potential opportunities. This requires that the planner have access to a vocabulary that differentiates between the different types of planning problems (e.g., resource limitations, time constraints, and the planner's limitations).

Next, the planner executes the plans for its active goals. During execution, it has to monitor the ongoing effects of its plan as well as the effects of the plans of others in its world. The representational elements used to do this parsing are the same elements with which suspended goals have been associated. As a result, the planner's general recognition of a situation that constitutes an opportunity can immediately activate any goals that have previously been associated with that situation.

Finally, any activated goals are integrated into the current set of scheduled steps, and the plan is executed. This requires reasoning about resources and protections, as well as the effects of actions.

4.2.1. Suspending blocked goals

In general, opportunities to run plans can be derived from the preconditions on each of the steps of a plan. A planner could, given time, move through a plan step by step and collect the preconditions that have to be true at that point in the plan. But this would require the examination of many conditions that are not particularly useful in the context of opportunism. Some preconditions for obtaining orange juice—having money, having time, and being able to carry the carton—are not useful if we are looking for the features that will allow us to recall the suspended goal at the appropriate time. This is because there are more constraints on "opportunities" than on simple preconditions. These constraints include features such as ease of recognition, likelihood of occurrence, and predictiveness.

For example, having money is a strong precondition for buying orange juice, but it is also a normative condition. As a result, it is a bad predictor of an opportunity to satisfy the goal to have orange juice. If the suspended goal is tied to having money, the planner will be reminded of the goal far too often.

Rather than test all preconditions of a plan for these constraints, we propose a taxonomy of *opportunity types* to derive the conditions that will serve as opportunities to satisfy the plan. This taxonomy guides the planner's search through the plan for appropriate preconditions. Once a plan has been analyzed in terms of this taxonomy, it is annotated with pointers to the features associated with opportunities to run it. Features are removed from this list if they cause the goal and plan to be recalled at inappropriate times.

This taxonomy takes the form of a set of tests or questions that the planner asks of a plan:

- **Is there a special[4] resource that is needed to run the plan?**
  If so, index the goal in terms of the resource.

- **Is there a special tool that is required to run the plan?**
  If so, index the goal in terms of the tool.
- **Is there a special location indexed with the plan?**
  If so, index the goal in terms of the location.
- **Is there a special agent or skill associated with the plan?**
  If so, index the goal in terms of the agent or skill.
- **Is there a specific time constraint associated with the plan?**
  If so, index the goal in terms of the time.

There are also special-purpose rules for suspending particular goals. Possession goals, for example, are associated with the object of the possession.

Using these rules, it is possible to associate in memory the blocked goal to posssess orange juice with the location, GROCERY-STORE, and the object itself, ORANGE-JUICE. This association takes the form of a *SUSPEND* link from the representations of these items to the suspended goal itself. Associating the goal using the least likely conditions as indices assumes that most of the other normative conditions will be true whenever the suspended goal is activated. Thus, the other conditions are checked when the goal is recalled, but they are not linked with or used to index to the suspended goal in memroy.

4.2.2. Recalling suspended goals

An agent usually has a wide variety of planning options for any one goal. In our example, John can pick up orange juice at *any* grocery store, not just a particular one. It is necessary, then, to be able to recognize a wide variety of situations as opportunities for goal satisfaction.

To deal with this, we have been using a version of Martin's (1990) DMAP parser, a general-purpose recognition system. DMAP uses a marker-passing algorithm in which two types of markers are used to *activate* and *predict* concepts in an ISA and PART-OF network. *Activation markers* are passed from primitive features up an abstraction hierarchy. For TRUCKER—which functions in a "neighborhood world" of roads, signs, and buildings—these features include type descriptions such as "road," "wall," "window," and "sign," but no tokens such as "the Seven-Eleven on Cornell and 47th." When any PART-OF a concept is active, *prediction markers* are spread to its other parts. When a predicted concept is handed an activation marker, it becomes active. Likewise, when all parts of a concept are activated, the concept itself is activated.

For opportunistic planning, we have added a new type of link to the memory structures. This link associates suspended goals with concepts that represent opportunities to achieve them. Pointing from concepts to goals, this SUSPEND link is traversed by any activation marker that is placed on the concept. Consequently, the activation of a concept also activates any suspended goals associated with it.

In our example, the suspended goal to get the orange juice is associated with the concept representing "the planner is at a grocery store." As the world is parsed, a Seven-Eleven is recognized as a sequence of "parking lot," "building," and "Seven-Eleven sign." Because DMAP is passing activation markers up ISA links, the Seven-Eleven is recognized as a particular Seven-Eleven, an instance of Seven-Elevens in general, a CONVENIENCE-

STORE, a GROCERY-STORE, and a STORE. While the suspended goal is not directly associated with the concept "Seven-Eleven," it is associated with GROCERY-STORE. So the recognition of the Seven-Eleven causes the activation of the suspended goal. In general, we can use this property of DMAP to recall goals associated with general characterizations of opportunities through the recognition of specific situations.

The approach here is straightforward. In order to execute a plan, the low-level features must be disambiguated into tokens representing specific objects in the world. As this is done, each token is checked for an associated goal. If any goal is found, it is considered a candidate for immediate satisfaction.

Note that we are not arguing that we will notice all instances of opportunities for satisfying the goal. This approach also predicts that some actual opportunities will be missed by the processor, if the opportunities occur in a form that could not be anticipated at the time of the goal block. We will only catch those that fall within the standard set of plans for goal satisfaction. Truly unique oportunities will be missed simply because this technique is aimed at *recognition* of known solutions rather than *creation* of new ones.

4.2.3. Exploiting opportunities

Once a suspended goal is reactivated, it has to be evaluated for integration into the current execution agenda.

The integration is easier than expected because many of the planning steps can be ignored in developing the new plan. In our orange-juice example, the steps required to get the planner to a store can be ignored, in that being *at the store* is the condition that activated the goal in the first place. But the planner can also ignore other steps; in particular, the steps that are used to "recover" from the precondition of being at the store once the plan is over can be ignored. Because the planner did not need to run the steps in the GROCERY-STORE plan to get to the store, it will not have to run the steps in that plan that will get it away from the store. In general, precondition/cleanup steps will occur in pairs that can be cancelled together. This is because the planner knows that the active plan must include the same pair, and need not be concerned with the part of the recalled plan that includes it.

The remaining steps—going into the store, buying the orange juice, and exiting—can be integrated in a fairly traditional way. The planner checks the preconditions not set by the activation conditions, and it notes the use of resources and their interactions with existing protections. The final product is a small change in the overall plan that takes the planner into the store for a moment before resuming its trip home.

4.3. Predicting opportunities

In addition to being a functional requirement for an agent without complete knowledge of the world, the ability to recognize opportunities to satisfy goals relates to the background goal of gaining costly information about the world. Just as in the case of execution-time failure, an unexpected opportunity may indicate a chance for learning so as to improve future performance. This is particularly true if there is a good reason to believe that the conditions that produced the opportunity will recur, or can be induced from experience to recur. Just as failure-driven learning is an alternative to exhaustive projection in debugging of conjunctive goal plans, learning from encountered opportunities presents a method for constructing conjunctive goal plans without complete search of the space of possible plans.

This is best understood in the context of an elaboration of our "buying orange juice" example. The plan for this goal is simple: go into the store, find the orange juice, buy it, and go home. During the execution of this plan, a planner will have to move through the store looking for the juice. As he does so, he may see a bottle of milk and recall the need for it. He also may recall that he had run out of aluminum foil as well.

At this point, he does what any optimizing planner should do: he merges the separate plans for obtaining milk, orange juice, and aluminum foil into a single plan for the conjunct of goals. He buys them all at once while at the store, rather than buying them one at a time and returning home with each.

Here the planner has three options for storing the new plan in memory: he can forget that this particular version of the GROCERY-STORE plan exists, he can save the entire plan that satisfies all three goals, or he can reason about what parts of the plan should be retained and how to index the resulting plan in memory.

The first option is clearly wrong. This is a useful optimization of three separate instances of the GROCERY-STORE plan and could easily be re-used if these goals arise again.[5] The second option is better, but ignores that fact that the cycle of use of aluminum foil (based on a combination of the rate it is used, amount purchased, and non-perishable status) is much longer than the cycles of use for either the milk or the orange juice. A plan for the entire conjunct, then, is going to be of little use.

What seems to make the most sense is the third option—decide which parts of the plan should be retained and where the resulting structure should be indexed. We are proposing that blocked goals are encoded and indexed in memory so as to be recalled under more favorable solution conditions. That is, one anticipates features related to circumstances where success would be possible, encodes the blocked goal into memory with those features, and retrieves the goal from memory again when those features actually occur in the world. The indexing of goals, then, must be tied to the features believed to be related to successfully achieving the goal under later circumstances. Dormant goals recognize opportunity by waiting for them in the "right place" in memory; if those features are activated later, the goal will be reactivated among the more favorable conditions. We call this process *predictive encoding*, where the features that indicate the relevance of the plan are anticipated and used to index the plan in memory.[6]

With the predictive encoding process, a planner takes this shopping experience and uses it to form a new plan to pick up milk when it is at the store getting orange juice—without also picking up aluminum foil each time as well. The rationale for this choice of items to be included in this plan is clear: Given the rate of use of orange juice and milk, there is a good chance that at any given moment you may be out of either. Given the rate of use of aluminum foil, however, there is little chance that at any one time you will need more of it. The plan for buying all three items is not constructed for the simple reason that the cycle of use and buying for the three items cannot be coordinated. While the cycle for milk and orange juice can be brought together, the use cycle for the aluminum foil is so much longer that there is no way to bring it into line with the other two.

At this point, it is not enough to create the plan and associate it with the conjoined goals of wanting to obtain milk and juice. This would allow the planner recall this plan when *both* goals are active, but would not give it a plan that assumes a *set* of goals is active in the presence of any one of them. What we really want is to have a plan that is activated when *any one* of the goals arises, and then checks for the existence of the other goals—that is, a plan that includes knowledge of the other plans with which it is typically merged.

To perform predictive encoding, the planner faces a two-fold task. It must evaluate the likelihood that a similar conjunction will ever arise again—i.e., determine if the plan is worth saving at all, and which goals in the initial conjunct should be included. Then it must determine the set of features that predicts the presence of the conjunct. In the language of case-based planning, it must determine how to index the plan in memory so that it will be recognized in appropriate circumstances. In the extreme case—and when combined with some enforcement (Hammond, 1990c)—these plans will converge to become a regularly performed trip to the grocery store.

The problem of how to learn the predictive indices that indicate when an opportunity is relevant can be approached either empirically or analytically. The task can be done empirically by trying the new plan when any one of the goals arises, and removing links between it and those goals that do not predict the presence of the other goals. This is, in essence, the method implemented in the program IPP (Lebowitz, 1980). Or it can be done analytically, using explanation-based learning methods to construct explanations for why the goals should or should not be expected to arise in concert. Regardless of the sophistication of the reasoning involved in deciding what conjoined plan to save, though, the crucial point is that the possibilities for conjunction are suggested by occurrence in the world, and not by projection methods.

The issue here is not so much the nature of the algorithm used to make this decision as much as the effect that the decision has on learning. The predictive encoding process—the interaction between planning and learning processes—results in a learner that saves *only* those plans for conjuncts of goals that have been demonstrated to arise during execution. The result is a learning system that builds up a corpus of plans for the conjuncts of goals that will tend to arise in the course of its execution.

5. An implementation of opportunistic memory: The TRUCKER program

Our first experiments with an implementation of opportunistic memory were in the TRUCKER program. In terms of the taxonomy discussed in section 2, this project was most concerned with the planning/execution crowding problem and with the run-time recognition of opportunities. As with the more general discussion of opportunistic memory, the discussion here relates to the opportunities for *learning* that a memory-based reasoning system provides.

TRUCKER is a University of Chicago planner that interacts with a simulated world in order to test out its plans and learn from both failure and success. Its domain is a UPS-like pickup and delivery task in which new orders are received during the course of a day's execution. Its task is to schedule the orders and develop the routes for its trucks to follow through town. A dispatcher controls a fleet of trucks that roams a simulated city or neighborhood, picking up and dropping off parcels at designated addresses. Transport orders are "phoned in" by customers at various times during the simulated business day, and the planner must see to it that all deliveries are successfully completed by their deadlines.

TRUCKER's task involves receiving requests from customers, making decisions about which truck to assign a given request to, deciding in what order given parcels should be picked up and dropped off, figuring out routes for the trucks to follow, and in general monitoring the execution of the plans it has constructed. A number of limited resources must be managed, including the trucks themselves, their gas and cargo space, and the planner's own planning time. TRUCKER starts off with very little information about the world that its trucks will be negotiating; all it has is the equivalent of a street map, an incomplete and potentially inaccurate schematic of its simulated world.

TRUCKER creates new plans using two means: a map and a memory. TRUCKER uses its map when it initially builds a route for an area. Once it has built a new route and verified it by running it in the world, it stores the route in a case memory, indexed by its literal endpoints as well as by neighborhood and part-of-town discriptors. When it is able to find a plan for an order in memory, the plan is expanded and placed at the end of TRUCKER's agenda of actions. TRUCKER never tries to optimize over multiple goals unless it already has a plan in memory that does so.

TRUCKER optimizes its planning for multiple goals only when it notices an opportunity to do so during execution. If TRUCKER notices an opportunity to satisfy a goal that is scheduled later in the agenda, it stops and reasons about the utility of merging the later plan with the steps it is currently running. If it is able to construct a plan that is significantly better than one which treats the plans independently, it uses the new plan. It also stores the new plan in memory, indexed by each of the separate goals. When either goal recurs, TRUCKER searches its agenda for the partner goal and uses the plan that it has created for the pair. The point here is that TRUCKER is able to construct a library of plans for the conjuncts of goals that will *tend* to arise, evidenced by the fact that they *do* arise.

In order to accomplish this, TRUCKER takes even goals placed on the action queue and treats them as though they were blocked. That is, with the predictive encoding process, it establishes the conditions that would allow TRUCKER to satisfy the goal, and then associates the goal with the memory structures that would be active during the recognition of those conditions. For example, while planning for a pickup at the Sears Tower later in the day, TRUCKER associates the goal with its internal representation of the Tower. This allows it to activate and then satisfy the goal if it recognizes the Sears Tower earlier in the day.

TRUCKER's approach to this task is a departure from conventional scheduling methods. These methods, however, with emphasis on exhaustive pre-planning, would be inadequate for this task due to a number of reasons:

- TRUCKER lacks perfect information about its world.
- TRUCKER does not know all of its goals in advance—new calls continually come in that must be integrated with currently running plans.
- Planning time is limited. TRUCKER's world does not wait for it to complete plans before new events occur.
- Even given perfect advance information, an optimal solution to the problem TRUCKER faces is computationally intractable. Even scheduling optimal pickup and delivery points for a single truck is a variant on the NP-hard traveling salesman problem.

TRUCKER must plan opportunistically, recognizing and acting upon opportunities for goal satisfaction as they arise, if only to deal with the fact that new goals are presented to it during the course of execution. Since planning time is limited and plan construction is costly, we further argue that plans should be stored and re-used as much as possible. Finally, patterns of opportunity that are recognized once should be learned, and should be easier to recognize again if they recur.

5.1. Opportunism in TRUCKER

We will illustrate the sort of opportunistic behavior that TRUCKER is capable of with reference to fragments of output from the program.

TRUCKER controls its fleet of trucks by deciding which trucks should receive given pickup-and-delivery orders, retrieving or calculating routes for the trucks to follow, and actually monitoring the progress of the trucks. The trucks are better viewed as effectors of the planner than as autonomous agents. Time spent planning and monitoring the progress of the trucks comes from the same budget.

TRUCKER's central control structure is a queue-based executor, reminiscent of Firby's RAP system (Firby, 1989), with planning and monitoring actions sharing space on the queue. In addition to planning the agendas for the trucks, and constructing routes for them to follow, the planner must react to new goals as they come in on the "telephone."

```
[Day #1]
7:58:00 AM   Planner Action: (ANSWER-TELEPHONE)
7:58:00 AM   Planner Action: (HANDLE-NEW-REQUEST REQUEST.41)
--- Planner relating request REQUEST.41 to memory ---
7:58:00 AM   Planner Action: (TRY-ASSIGN-TO-IDLE-TRUCK
REQUEST.41)
PLANNER assigning request REQUEST.41 to truck #1
```

In absence of good reasons to the contrary, the planner hands pickup and dropoff orders to trucks based on availability in the order they come in, and integrates the new order into the truck's agenda. In the case where the truck is idle, no real integration is necessary, and the standard plan of traveling to the pickup point and then to the dropoff point is used.

```
Starting INTEGRATE-REQUEST  REQUEST.41  #1  IDLE
Resulting new plan:   ((GOTO (800 E-61-ST))
                       (PICKUP PARCEL.42 (850 E-61-ST))
                       (GOTO (6200 S-COTTAGE))
                       (DROPOFF PARCEL.42 (6230 S-COTTAGE)))
```

When TRUCKER receives a new request for a pickup and delivery, it attempts to satisfy the order using a variety of methods. First it checks all active requests on its trucks' agendas for one that has a known positive interaction with the new request. If this fails, TRUCKER attempts to find a truck that is currently idle to take up the order. If this also fails, TRUCKER searches its "desktop" for a suspended request that might be usefully combined with the new order. If all else fails, TRUCKER is forced to place the request on a queue of orders waiting for idle trucks and must construct a new route for the truck using its map and current information about the available trucks. This takes time away from the planner's other activities.

```
7:58:39 AM   *** Truck #2 making delivery at 1450 E-62-ST ***
   --- Planner searching memory for route from
                    (6100 S-WOODLAWN) to (800 E-61-ST) ---
    Search unsuccessful.
   ========   PLANNER consulting map to build route ========
8:03:12 AM   *** Truck #1 has a new route:   ***
                         ((START N 6100-S-WOODLAWN)
                          (TURN W E-61-ST)
                          (STOP 800-E-61))
8:03:12 AM   *** Truck #1 is starting new route at 6100 block of S
Woodlawn ***
8:05:48 AM   *** Truck #1 making pick-up at 850 E-61-ST ***
*** Truck #2 done with delivery ***
```

Whenever TRUCKER is forced to construct a new route from scratch, it both constructs the plan and suspends the goal associated with it in memory, using predictive encoding to anticipate recognition conditions. To suspend a goal, TRUCKER marks its representative of the goal's pickup and delivery points with an annotation that there is a goal related to those locations. Because TRUCKER plans for only one type of goal, it does not have to do any more reasoning than this to identify good opportunities to satisfy the suspended goals.

TRUCKER ties execution of actions to locations, landmarks, and addresses that it recognizes in the world. It must parse and interpret the objects in the world, since in addition to planning routes and agendas, it must monitor execution and direct trucks to turn and stop at appropriate times. It is during this parse that TRUCKER recognizes and recalls previously suspended goals. A typical TRUCKER plan, when fully expended, is a route in the form of a list of the turns that have to be made, described in terms of street names and compass directions. So the plan step (GOTO (920 E-55th)) after a pickup at (5802 S-WOODLAWN) expands into

```
(START NORTH (5802 S-WOODLAWN))
(TURN EAST E-57TH)
(TURN NORTH S-CORNELL)
(TURN EAST E-55TH)
(STOP (920 E-55TH))
```

As TRUCKER moves through its world, it parses the objects at its current location and responds to any changes that the tokens it has recognized suggest: turning, for example, when it recognizes the 5700 block of Woodlawn. But TRUCKER does more than this when it recognizes an individual token. It also checks the token for any annotation of a goal that might be associated with it. If one is found, TRUCKER activates the suspended goal and attempts to integrate it into the current schedule. This allows TRUCKER to easily and effectively activate suspended goals when the opportunities to satisfy them arise. Further, the overhead on this activation is trivial, in that all that TRUCKER has to do is look for a specific type of link on each of the objects it recognizes.

The same memory for places and landmarks that is used to tell the trucks when to turn and where to stop is annotated with the delivery goals that have not yet been satisfied. When such a location is recognized in the course of executing another delivery, the possibility of opportunistically satisfying the goal is suggested. At this point the planner is invoked to construct a new route that satisfies both goals. The following fragment illustrates the process of noticing the opportunity, the subsequent reassignment of a request to a different truck, and the invocation of the planner to construct an optimized agenda that incorporates the opportunity.

```
8:17:12 AM   *** Truck #1 is starting new route at
               800 block of E 61st Street ***
*** Truck #1 has noticed an opportunity to make the
    pickup for request REQUEST.49 ***
*** Request REQUEST.49 is assigned -- Truck
     #1  inserting reassignment request in
     planner's agenda.
*** Noting combination opportunity in memory ***
8:17:45 AM   Planner Action: (REASSIGN-BY-NOTICED-OPPORTUNITY
REQUEST.49 #1)
--- Reassignment of request REQUEST.49 means that truck #2
   need not continue to its destination.
PLANNER assigning request REQUEST.49 to truck #1
Starting INTEGRATE-REQUEST  REQUEST.49  #1  NOTICED-
OPPORTUNITY
Current plan:    ((GOTO (6200 S-COTTAGE) #{Structure ROUTE 2})
                  (DROPOFF PARCEL.42 (6230 S-COTTAGE)))
Request-plan to integrate:    ((GOTO (6100 S-COTTAGE))
                               (PICKUP PARCEL.50 (6150 S-COTTAGE))
                               (GOTO (900 E-63-ST))
                               (DROPOFF PARCEL.50 (925 E-63-ST)))
8:17:45 AM   *** Truck #1 is stopping in  6100 block of S Cottage
Grove ***
Finishing INTEGRATE-REQUEST  REQUEST.49  #1  NOTICED-
OPPORTUNITY
Resulting new plan:   ((GOTO (6100 S-COTTAGE))
                       (PICKUP PARCEL.50 (6150 S-COTTAGE))
                       (GOTO (6200 S-COTTAGE))
                       (DROPOFF PARCEL.42 (6230 S-COTTAGE))
                       (GOTO (900 E-63-ST))
                       (DROPOFF PARCEL.50 (925 E-63-ST)))
8:17:48 AM   *** Truck #1 making pick-up at 6150 S-COTTAGE ***
```

After having noticed the opportunity and suggesting the reassignment, TRUCKER uses special-purpose techniques tailored to the domain to plan the new agenda. Scheduling the pickup is trivial, in that a truck is at the pickup location. The difficulty lies in scheduling the delivery. TRUCKER does this by stepping through each location already scheduled and finding the section of the route that will be the least altered by the insertion of the delivery. This can be done even before the exact routes are selected, by using the map and simple rules of geometry.

In this way the full planner is invoked only when the recognition of an opportunity to satisfy a pending goal suggests that a combination of delivery orders may be fruitful. The noticed opportunity is also saved in memory, so that the conjunction may be exploited again if it reoccurs.

5.2. Reuse and adaptation in TRUCKER

Plans in TRUCKER are routes. They are indexed by the goals that they satisfy as well as initial locations of the trucks that will satisfy them. When a pickup/delivery pair is given to the system, it is used to pull a particular plan out of long-term memory, using the specific locations as indices. If this indexing fails, the particulars are generalized upwards through BLOCK, STREET-SEGMENT, NEIGHBORHOOD, and CITY-SEGMENT. GOTO steps in the plan are marked as to whether they control action within each of these designations or between them. Once a plan is found, the linking sections associated with the level of generality that was used in the indexing is reused and the rest discarded and replanned around.

Because trucks move about, their locations are also used in the retrieval of plans in memory. The segments that are retrieved are adapted and then concatenated together to make up a new, complete route for the goal set currently under consideration by the planner.

5.3. Learning in TRUCKER

There is considerable regularity and repetition in the orders that the world simulation hands to TRUCKER. TRUCKER exploits this in a case-based manner, by saving particular constructed routes, remembering conjuncts of requests that have been profitably combined, and remembering the particular interleavings of steps that these conjuncts produced. When the conjuncts of goals reoccur, TRUCKER recognizes them as a known conjunct of goals for which it has a plan and uses the plan for that conjunct that it has saved in memory.

While the final version of TRUCKER did no explicit generalization, the system, like the CHEF system, created implicit generalization through the indexing process. While the route information remained very specific, the paths themselves were indexed by address, block, and neighborhood. This allowed TRUCKER to retrieve and use partially matching paths without losing any information. Such partially matching plans were then adapted using the same path finding techniques that were used in the construction of the initial plan.

When TRUCKER receives a request, the request's long-term record is inspected to see if there are any notations about combinations with other requests. If so, the planner looks to see if the other requests are currently active. If it finds the requests that it has previously been able to merge with the current one, the plan for the conjunction is added to the agenda rather than those for the idividual requests. It is important to note that plans for combination of requests are only activated when all the requests are active.

```
[Day#2]
8:17:00 AM   Planner Action: (ANSWER-TELEPHONE)
--- Planner relating request REQUEST.73 to memory ---
8:17:00 AM   Planner Action: (TRY-ASSIGN-TO-USEFUL-TRUCK
REQUEST.73)
 --- Truck #1 is pursuing a request that has been
previously associated with route of request REQUEST.73
 PLANNER assigning request REQUEST.73 to truck #1
Starting INTEGRATE-REQUEST  REQUEST.73  #1  GOING-NEAR
Current plan:    ((GOTO (6200 S-COTTAGE)) (DROPOFF PARCEL.72
(6230 S-COTTAGE)))
Request-plan to integrate:    ((GOTO (6100 S-COTTAGE))
                               (PICKUP PARCEL.74 (6150 S-COTTAGE))
                               (GOTO (900 E-63-ST))
                               (DROPOFF PARCEL.74 (925 E-63-ST)))
Resulting new plan:   ((GOTO (6100 S-COTTAGE))
                       (PICKUP PARCEL.74 (6150 S-COTTAGE))
                       (GOTO (6200 S-COTTAGE))
                       (DROPOFF PARCEL.72 (6230 S-COTTAGE))
                       (GOTO (900 E-63-ST))
                      (DROPOFF PARCEL.74 (925 E-63-ST)))
```

This automatic combination of requests that have been joined in the past will not necessarily lead to an optimal assignment of requests to trucks, at least in a sense of optimality that ignores the cost of the work done in arriving at the assignment. There is a side benefit to this standardization, however. Having the planner use existing interleavings means that the routes between internal points of the schedule can also be reused:

```
========   PLANNER consulting map to build route   ========
8:17:12 AM   *** Truck #1 has a new route:   ***
                        ((START W 800-E-61)
                         (TURN S S-COTTAGE)
                         (STOP 6100-S-COTTAGE))
8:17:12 AM   *** Truck #1 is starting new route at
               800 block of E 61st Street ***
8:17:39 AM   *** Truck #1 making pick-up at 6150 S-COTTAGE ***
   --- Planner searching memory for route from
                   (6100 S-COTTAGE) to (6200 S-COTTAGE) ---
    Search successful.
=== PLANNER knows route from 6100S-COTTAGE to 6200S-COTTAGE ===
```

In TRUCKER, the question of whether or not to save a plan for a new conjunct in memory is dealt with by keeping statistics on use and reuse of plans. Initial plans are saved, and then information about false alarms (the plan is suggested and then not used) as well as successful reuses is stored away in association with the plan. If the level of reuse versus false alarms falls below a preset threshold, the indexing is altered so as to make it impossible to access the plan except when both goals that it satisfies are active.

5.4. Analysis and evaluation

Given some regularity in the orders it receives, TRUCKER builds a library of planned routes and a library of conjoined plans for groups of requests that have occurred together. This opportunistic noticing of chances for combination is in part forced by the nature of the task, both because a complete solution is intractable and since action must begin before all the goals are known. Learning from these encountered opportunities advances TRUCKER's background goal of learning about interesting regularities in its environment, and helps amortize the complexity over the long term of satisfying its goals.

The complexity of planning is amortized across reuses in that TRUCKER needs to create a plan for a conjunct only once and then reuse it as the goals recur. The only problem with this is the possibility that it will be saving plans for conjuncts that were only coincidental and will never be seen again. TRUCKER deals with this by keeping statistics on plan activation and reuse that allows it to discard plans that are being suggested but never verified and reused. As with all case-based systems, the assumption is that the domain itself is regular with regard to the goals that will tend to be conjoined. In further work on RUNNER, this question of reuse is handled by building up explanations as to why the two plans have come together to begin with. If a repetition can be predicted, then the plan for the conjunct is saved for reuse. Otherwise it is discarded.

The learning of routes is not particularly interesting from a case-based perspective, since it amounts to memorization. Because TRUCKER stores constructed routes, it saves planning time if the same goals recur. The issues of indexing and retrieval are not all that interesting in TRUCKER's domain. However, the learning of plans that group requests is more interesting, partly because the situations in which the stored plans will be reused may be different than those in which they were originally applied.

This method of learning conjunctive plans confers three sorts of advantages to TRUCKER:

- First, there is the benefit of having learned about profitable combinations of goals in the course of activity that had to be done anyway.
- Second, there is the benefit of having saved the results of more conventional planning in those situations when there is already an indication of the profitability of combination.
- Third, the implicit policy decision to reuse combinations that have worked in the past reduces the complexity of the remaning planning and scheduling problem. The recurrence of particular goals provides a source of stability in the domain; the decision to use the same plans for those recurring sets provides an additional source of stability.

The payoff associated with the first and second advantage is immediate and straightforward. In the TRUCKER domain, where a major percentage of the orders are stable over time,[7] certain conjuncts of orders will also tend to be stable and thus predictable. This means that the plans for those conjuncts can be reused directly, saving both planning time (because the interactions no longer have to be reasoned about) and execution-time (because an optimized version of the plan for the conjuncts is being applied)

The more stable a domain with respect to the goals that will tend to arise, the more advantage there is to reusing the plans associated with the opportunities associated with conjuncts of goals. Even in relatively unstable domains, however, islands of stability can be recognized (in that the planner is collecting plans that are clustered around frequently recurring goals) and the efforts in building optimized plans associated with opportunities can still be amortized over multiple uses.

There is a trade-off associated with the third benefit. It would be easy to construct adversary arguments to show that an early decision to combine goals may lead to worse performance. There are two central assumptions necessary for this sort of learning to be of use: first, that there is strong regularity in the set of goals that arise and occur simultaneously, and second, that the domain is complex enough that policy decisions that simply reduce that complexity may be beneficial. We argue that many real-world domains have both of these characteristics.

In general, it is difficult to evaluate the performance of a system like TRUCKER. This is because CBP itself assumes a domain that is stable with respect to goals and the external environment, yet both of these are, within a simulated world, controlled by the system developers. To fit this assumption, we constructed a world for TRUCKER that reflected this assumption of overall global order. Within this world, TRUCKER runs extremely well, reusing a quickly stabilized case library. While the world that TRUCKER functions within was created by us, however, it does truly reflect our view of the nature of the structured environments which themselves support intelligent behavior.

Of course, in other simulated worlds in which there is a far greater degree of randomness, TRUCKER may very well exhibit all of the failings that we fear in a case-based system: reuse of inappropriate cases, explosion of a case library, and exhaustive search of the space of possible plans. To hand TRUCKER or any system this type of world, however, actually argues that our world is itself random. We confess that we have difficulties with this notion.

5.5. Psychological evidence

There is also strong evidence that human memory shares many of the organizational properties proposed in CHEF and TRUCKER. Psychological evidence also suggests that features that predict possible planning failures are more useful in retrieving past cases than are equally similar but less predictive features (Johnson & Seifert, 1991).

We conducted several experiments to investigate the retrieval of prior cases based on new cues. In a single-session reminding paradigm (based on Gentner & Landers, 1985), subjects studied a set of base stories and then later were given a set of cue stories and asked to write down any base stories that come to mind. Two types of cues were examined: predictive features subsets included only the thematic features apparent before the planning decision was made, and were predictive of a pending planning failure; by contrast, outcome features included the thematic elements containing the same planning decision and its outcome. The main memory findings were that both predict and outcome cues were retrieved from memory at higher-than-chance level. However, the predict-theme cues produced significantly fewer mismatch intrusions than did outcome features. Therefore, the stories containing predictive features led to more *reliable* access to matching stories in memory.

It appears that the predictive cues characterized the themes of the base stories more uniquely than the outcome cues did. Thus, predictive features appear to more *distinctively* characterize the theme, and therefore are more successful in recognizing when the past episode is in fact relevant in current processing. These results confirm the predictive features approach used in CHEF and TRUCKER. In human memory, indices related to when and how to make a particular decision are more useful than equally related information that is too late in the decision process to affect the choice.

Likewise, a classic finding by Zeigarnik (Zeigarnik, 1927) has claimed that interrupted (suspended) problems are remembered better than solved problems. Zeigarnik found that, among a set of 20 simple problems, interrupted tasks were better remembered than completed ones in a free recall test. This "Zeigarnik effect" was explained by appealing to social and motivational factors (Prentice, 1944). However, predicting whether interrupted problems would be more accessible in memory may involve more subtle cognitive factors operating in the experimental task (Seifert & Patalano, 1991). What, if any, are the differences in the way in which we encode and remember completed versus interrupted problems?

In a series of experiments, we explored factors including the nature of the interruption, the processing time spent on problems, and the context of set size of the incomplete problems as they affect memory. We attempted to match Zeigarnik's (1927) methods as closely as possible, and to replicate the effect of better memory for incomplete problems. Using word problems, we manipulated task interruption versus completion on each problem. Our results showed that free recall memory for completed tasks is *better* than memory for interrupted tasks. However, this is not surprising given that subjects spent substantially more time, both when correct and incorrect, on the completed problems than on the incomplete problems, a difference required if one wants to manipulate which problems are completed and keep the set size of the two conditions equal.

We hypothesized that, under blocked rather than simple interrupted conditions, a heightened memorability exists for blocked problems when they are outnumbered by successful solution attempts. We created such a situation by modifying the earlier experiments to include more difficult problems and allowed subjects less time to work on each problem. The results showed equal percentages of recall of the two problem types when roughly equivalent set sizes were used. Fewer unsolved problems may become memorable because they violate subjects' *expectations* about their performance. In these experiments, experience with the problem set may have created expectations of success or failure, so that when the problem solver is surprised, the problem becomes more memorable. The availability of exceptions in memory is a well-known effect in the person perception literature (Hastie & Kumar, 1979), and as expectation failure in inference and schema effects.

The enhanced memorability of the Zeigarnik effect appears to be due to differences in encoding based on expectation failures that mark significant events that deviate from predictions. This serves to highlight memories that may be important and to enhance availability for later processing. Under these conditions, the status of completion can serve as a useful index to past problem situations. Therefore, we conclude that heightened memorability for suspended goals is a reliable phenomenon, but may be best explained in terms of differences in the nature of encoding for some problems (motivated by expectation failure) rather than any separate memory store or content differences in suspended goal situations.

5.6. Beyond TRUCKER*: Memory and agency*

The TRUCKER domain was a good one for studying issues of opportunistic behavior and learning, since the inherent intractability and time pressure of the domain forced solutions that did not depend on complete projection and modeling of the world. However, there are a number of limitations with this domain. First of all, although opportunity recognition is crucial, there is really only one type of opportunity that can be exploited in this domain: the presence of a truck in a particular area. Secondly, although multiple trucks need to be tracked, and multiple orders need to be interleaved, there is not much additional interesting interaction between plans. Since we believe that a large part of the development of domain expertise involves learning the interactions between goals and plans that typically arise in conjunction, a model in a domain that is richer in interactions would provide more challenges to the approach.

TRUCKER executes plan steps by pulling them off an agenda one at a time. As a result, standard execution is rather rigid. On the other hand, activation of suspended goals is driven by recognition of features in the execution environment. As a result, this activation is quite flexible. In developing the opportunistic memory architecture, we noted that it could be used not only for the suspension and reawakening of goals, but for the control of planning and action on a finer grain. In essence, this amounted to a decision to discard the queue-based execution system of TRUCKER, and draw execution under the same uniform memory architecture as opportunity recognition. The result is an executive that is driven from the top by the activation of plans and actions aimed at satisfying the system's current goals and from the bottom by the presense of features in the world that allow their execution.

In current work on the RUNNER system (Hammond et al., 1990), we are looking at the idea that a case-based system can be used to control the execution as well as the production of plans. In this system, we are developing a unified framework for planning and activity that is based on the idea that most of these aspects of behavior in the world stem from the application of well-rehearsed plans, rather than creation of new ones. Our movement into the study of action, then, stems directly from the core concepts of case-based reasoning that define tasks in terms of memories of how they were accomplished in the past.

We are also looking at a further implication of case-based reasoning that we call *enforcement* (Hammond, 1991). Enforcement parallels learning in that both are aimed at building a functional correspondence between the external world and an agent's model. In learning, this is done by changing the agent's model and in enforcement, by changing the agent's world.

The goal associated with enforcement is the same as that associated with learning in the context of planning—the development of a set of effective plans that can be applied to satisfy the agent's goals. The path toward this goal, however, is one of shaping the world to fit the agent's plans rather than shaping the agent to fit the world. The idea of enforcement rises out of the observation that the results of long-term interaction between agent and environment include an adaptation of both the environment and the agent. Enforcement is linked to case-based reasoning in that it is aimed at stabilizing the environment with respect to the planner's already existing plans. The result of enforcement, then, is a case library of plans that can be run in a world that is itself fine-tuned for them just as they are for it.

6. Conclusions

Our current work rests on the notion that intelligent behavior is a long-term activity, and that much of it is aimed at learning the structure of a domain. Our central premise is that the stability of the world includes stability over the collections of goals that we will be called upon to satisfy, the types of difficulties we will encounter, and the kinds of conditions that we will be forced to overcome. It is only through learning that an agent can develop a true expertise in any domain. And it is only through an ongoing attempt to satisfy goals in a dynamic world that chances for learning can ever be encountered.

Our work began with the simple notion of applying case-based reasoning to the problem of deliberative planning (Hammond, 1986a). A natural progression of ideas has followed from this: reuse of plans (Hammond, 1986b; Hammond, 1990a), learning from failure (Hammond, 1986c; Hammond, 1990b), learning from opportunity (Hammond et al., 1988; Hammond et al., 1989; Seifert & Patalano, 1991; Johnson & Seifert, 1991), memory-based control of action (Hammond et al., 1990), and enforcement (Hammond, 1990c). These ideas are straightforwardly implied by case-based reasoning and are inextricably linked to issues of learning from planning and execution.

Case-Based Planning: Case-based planning both requires and facilitates learning. By definition, a case-based system reuses the products of its own reasoning, and as a result learns from that reasoning. More importantly, a case-based planner is able to determine *when* it should learn as well as *what* it should learn, because it is able to judge how well retrieved cases fit new situations (Hammond, 1986a). This is the simplest form of learning done by a case-based planner. The more interesting forms arise in response to failure.

Failure-Based Learning: By its nature, a case-based planner can create and execute plans without ever validating the effects of each action within them. As a result, a case-based planner can fail. While this can be problematic, it is actually one of the sources of power for a case-based system, in that failure provides the system with both the opportunity to learn and the indication that it is time to do so. In CHEF, this resulted in a system that would learn about the problematic interactions between goals and operators in the domain as well as learning about the plans that would avoid them. It learned to *anticipate and avoid* the actual problems that occurred in its domain.

Learning from Opportunity: TRUCKER was then designed to recognize and fill gaps corresponding to knowledge about planning optimizations and execution opportunities. Just as memory was used in CHEF to avoid failures experienced by the system, memory was used in TRUCKER to exploit opportunities. In both cases, the systems learned by using experience to shape their knowledge of plans and the conditions under which they should be run.

Opportunistic Memory: One unique aspect of TRUCKER was the understanding system it used to construct a dynamic model of its environment, as well as to recognize opportunities to satisfy suspended goals. This recognition system was built on top of DMAP (Martin, 1990), a case-based model of understanding. In particular, our main interest in TRUCKER was learning the plans for frequently occurring conjuncts of goals. A by-product of work, however, was the realization that DMAP was a powerful tool not only for the recognition of execution-time opportunities for goal satisfaction, but also for the production of plans and the control of action. It was this realization that has led to our current work on RUNNER.

In RUNNER, we are looking at the idea that a case-based system can be used to control the execution as well as the production of plans. In this system, we are developing a unified framework for planning and activity that is based on the idea that most of these aspects of behavior in the world stem from the application of well-rehearsed plans, rather than creation of new ones. Our movement into the study of action, then, stems directly from the core concepts of case-based reasoning that define tasks in terms of memories of how they were accomplished in the past.

Enforcement: One last implication of case-based reasoning that we have recently been exploring is an idea we call *enforcement* (Hammond, 1991). Enforcement parallels learning in that both are aimed at building a functional correspondence between the external world and an agent's model. In learning, this is done by changing the agent's model and in enforcement, by changing the agent's world.

The goal associated wth enforcement is the same as that associated with learning in the context of planning—the development of a set of effective plans that can be applied to satisfy the agent's goals. The path toward this goal, however, is one of shaping the world to fit the agent's plans rather than shaping the agent to fit the world. The idea of enforcement rises out of the observation that the results of long-term interaction between agent and environment include an adaptation of both the environment and the agent. Enforcement is linked to case-based reasoning in that it is aimed at stabilizing the environment with respect to the planner's already existing plans. The result of enforcement, then, is a case library of plans that can be run in a world that is itself fine-tuned for them just as they are for it.

Each of these pieces of research builds on the foundation of case-based reasoning and learning, and each is a necessary consequent of the commitment to case-based reasoning as a model of cognition. Further, each is linked to the idea that intelligent behavior is an ongoing process of planning, doing, and learning in response to the demands of a dynamic environment. Together, these individual processes—learning from failure, learning from opportunity, opportunistic memory, and enforcement—point toward a model of agency, or what it means to be an autonomous agent in a changing world.

Acknowledgments

Our work on agency is the part of a collaborative effort based at the University of Chicago. As such, this article owes much to Charles Martin, Christopher Owens, and Phil Agre, as well as the students at the University of Chicago Artificial Intelligence Laboratory. In addition, related psychological research has been carried out by Hollyn Johnson and Andrea

Patalano at the University of Michigan. Careful comments by reviewers and Mary Bellmar have also been very helpful.

This work was supported in part by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058, and the Office of Naval Research under contract N0014-85-K-010.

Notes

1. In what follows, the phrases "planning problems" and "execution problems" will refer to those problems that arise as a result of the interaction between planning steps. The former refers to those problems that are discovered during the planning phase, while the latter refers to those that are not discovered until execution. The word "goal" will be used in the standard way to refer to those states that a planner is trying to actively achieve.
2. By "associated with," we mean that the knowledge structure representing a goal is linked to the knowledge structures describing the opportunities under which it can be satisfied. This allows the former to be activated as soon as the latter is recognized.
3. One could argue that in this example the planner does completely preplan for this goal and that the plan is to get the orange juice on the way home. But this is begging the question, in that we can take any version of a plan designed to satisfy this goal and still argue that opportunities to satisfy it in other ways should be exploited. For example, if John passes by someone giving away free samples of orange juice on the way to work, we would certainly want him to recognize that his is a chance to satisfy a currently suspended goal. The point is that we want a planner to exploit opportunities to satisfy goals, whether or not it has already planned for them.
4. A feature is "special" if it does not predictably reoccur as a product of the planner's policy decisions.
5. One could argue that our planner does *not* need to learn the new plan in this example, in that he can just rebuild this plan from scratch the next time this situation arises. But this is just begging the question, because it is easy to change the example slightly—by raising the cost of execution-time planning or obscuring the visual cues—thus making the anticipation of the goal conjunct far more valuable.
6. The idea of *predictive encoding* has been implemented in both TRUCKER (Hammond et al., 1989; K.J. Hammond & Converse, 1989) and RUNNER (Hammond et al. 1990). Other examinations of this idea can be found in Robinson and Kolodner (1991).
7. There are two sources to the stability of the TRUCKER domain. The first is the fact that most shipping is between standard clients, and so stable sets of pickups can be established. The second is through overall patterns of activity in which certain neighborhoods (e.g., industrial, retail) have their own level of traffic even though there may be fluctuations at individual locations. The plans for the former can be reused without adaptation, while the plans for the latter can be used with only minor changes.

References

Alterman, R. (1986). An adaptive planner. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 65-69). Philadelphia, PA: AAAI.

Bareiss, R. (1989). *Exemplar-based knowledge acquisition*. Vol. 2 of *Perspective in artificial intelligence*. San Diego, CA: Academic Press.

Birnbaum, L., & Collins, G. (1984). Opportunistic planning and Freudian slips. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, CO.

Birnbaum, L.A. (1986). *Integrated processing in planning and understanding* (Technical Report 489). Ph.D. thesis, Yale University.

Byrne, R. (1977). Planning meals: Problem solving on a real data base. Cognition, *5*.

Carbonell, J. (1981). Counterplanning: A strategy-based model of adversary planning in real-world situations. *Artificial Intelligence*, *16*, 295-329.

Chapman, D. (1985). Nonlinear planning: A rigorous reconstruction. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 1022–1024). IJCAI.

Dean, T., Firby, R.J., & Miller, D. (1987). The forbin paper (Technical Report 550). Department of Computer Science, Yale University.

Fikes, R., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, *2*, 189–208.

Firby, R.J. (1989). Adaptive execution in complex dynamic worlds (Research Report 672). Computer Science Department, Yale University.

Gentner, D. & Landers, R. (1985). Analogical reminding: A good match is hard to find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, Tuscon, AZ, November.

Hammond, K.J., Converse, T., & Marks, M. (1988). Learning from opportunities: Storing and reusing execution-time optimizations. In *The Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 536–540). AAAI.

Hammond, K.J., Converse, T., & Marks, M. (1989). Learning from opportunity. In *Proceedings of the Sixth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann.

Hammond, K.J., Converse, T., & Martin, C. (1990). Integrating planning and acting in a case-based framework. In *The Proceedings of the 1990 National Conference of Artificial Intelligence*, August.

Hammond, K. (1986). *Case-based planning: An integrated theory of planning, learning and memory* (Technical Report 488). Ph.D. thesis, Yale University.

Hammond, K. (1986). Chef: A model of case-based planning. In *Proceedings of the Fifth National Conference on Artificial Intelligence*. Philadelphia, PA: AAAI.

Hammond, K. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. In *The Proceedings of the Fifth National Conference on Artificial Intelligence*. Philadelphia, PA: AAAI.

Hammond, K. (1989). *Case-based planning: Viewing planning as a memory task*. Vol. 1 of *Perspectives in artificial intelligence*. San Diego, CA: Academic Press.

Hammond, K.J. (1990). Case-based planning: A framework for planning from experience. *The Journal of Cognitive Science*, Fall 1990. Norwood, NJ: Ablex Publishing.

Hammond, K.J. (1990). Explaining and repairing plans that fail. *Artificial Intelligence Journal*, *45*(2).

Hammond, K.J. (1990). Learning and enforcement: Stabilizing environments to facilitate activity. In *The Proceedings of the Seventh International Conference on Machine Learning*, July.

Hammond, K.J. (1991). Learning and enforcement: Stabilizing environments to facilitate activity. In *aaai91*, August.

Hastie, R., & Kumar, A. (1979). Person memory: Personality traits as organizing principles in memory for behaviors. *Journal of Personality and Social Psychology*, *37*.

Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning. *Cognitive Science*, *3*(*4*), 275–310.

Johnson, H.K., & Seifert, C.M. (1991). Predictive features in analogical access. *Journal of Memory and Language*.

Kolodner, J.L., & Simpson, R.L. (1989). The mediator: Analysis of an early case-based problem. *Cognitive Science Journal*.

Kolodner, J. (1984). *Retrieval and organizational strategies in conceptual memory*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kristian, M.M., Hammond, J., & Converse, T. (1989). Planning in an open world: A pluralistic approach. In *Proceedings fo the 1989 Meeting of the Cognitive Science Society*, Ann Arbor, MI. Hillsdale, NJ: Lawrence Erlbaum Associates.

Lebowitz, M. (1980). *Generalization and memory in an integrated understanding system* (Technical Report 186). Ph.D. thesis, Yale University.

Lesser, V., Fennell, R., Erman, L., & Reddy, D. (1975). Organization of the Hearsay-II speech understanding system. *IEEE Transactions on Acoustics, Speech, Signal Processing*, ASSP, *23*, 11–33.

Martin, C.E. (1990). *Direct memory access parsing*. Ph.D. thesis, Yale University.

McDermott, D. (1978). Planning and acting. *Cognitive Science*, *2*, 71–109.

Newell, A., & Simon, H. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Owens, C. (1990). *Indexing and retrieving abstract planning knowledge*. Ph.D. thesis, Department of Computer Science, Yale University. In preparation.

Prentice, W. (1944). The interruption of tasks. *Psychological Review*, *51*.

Robinson, S., & Kolodner, J. (1991). Indexing cases for planning and acting in dynamic environments: Exploiting hierarchical goal structures. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society.* Chicago, IL: Cognitive Science Society.

Sacerdoti, E. (1975). A structure for plans and behavior (Technical Report 109). SRI Artificial Intelligence Center.

Schank, R.C., & Abelson, R. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. (1982). *Dynamic memory: A theory of reminding and learning in computers and people.* Cambridge: Cambridge University Press.

Seifert, C.M., & Patalano, A.J. (1991). Memory for interrupted tasks: The Zeigarnik effect revisited. In *Proceedings of the Thirteenth Annual Cognitive Science Society.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Stefik, M. (1981). Planning with constraints. *Artificial Intelligence, 16,* 141-169.

Zeigarnik, B. (1927). Das behalten erledigter und unerledigter handlungen. Psychologische Forschungen, *91,* 1-85.

Received July 5, 1990
Accepted February 15, 1992
Final Manuscript May 7, 1992