

Opportunistic Memory*

Kristian J. Hammond

Department of Computer Science

1100 East 58th Street

The University of Chicago

Chicago, IL 60637

Abstract

In this paper, we present a model of opportunistic planning that uses planning-time reasoning about the opportunities that might arise during plan execution. The model is composed of three parts: a planning-time mechanism that places blocked goals into memory for later activation, an understanding system that activates suspended goals as a by-product of parsing the world, and an execution-time process for evaluating opportunities and merging newly activated goals into the planning/execution agenda. We discuss this model in terms of examples from TRUCKER, a route-scheduling system, and RUNNER, an errand planner.

1 Planning and Acting

Current research in planning has taken a rather dramatic change in course in the past few years. The notion that a planner can exhaustively preplan for a set of goals prior to execution has been largely abandoned. In part, this change is the result of demonstrations that planning for conjunctive goals is undecidable [Chapman, 1987]. A more important factor has been the realization that the dual assumptions of traditional planning (a closed world and complete knowledge) are untenable in any but the simplest domains.

The planning theories that have grown out of this shift in paradigm differ from earlier theories in that they attempt to integrate planning and execution into a single process. The most extreme example of this has been the idea of *situated activity* [Agre & Chapman, 1987], which argues that plan-like behavior rises out of reflexive responses to external cues rather than from the guidance of a declarative plan. Less extreme theories center around the notion that plans have to be refined and repaired at execution time. These theories include:

- Alterman's *Adaptive Planning* [1985], in which execution-time problems are handled by moving between "semantically" similar plan steps.

*This work was supported in part by the Office of Naval Research under contract number N00014-88-K-0295 and by DARPA contract F4962-88-C-058.

- Firby's *RAPs* [1987], which allow a hierarchical planner to select between alternative plans on the basis of bottom-up information obtained at execution time.
- Georgeff and Lansky's *Procedural Reasoning System* [1987], in which changing goals and beliefs about the state of the world determine which plans are chosen to be put on the execution stack.
- Hammond's *Case-Based Planning* [1989] and Simon's and Davis' *Generate, Test, and Debug* [1987], both of which use causal explanations of execution-time failures to choose between a variety of repairs.

Moderate or extreme, each of these theories argues that the world changes, and that a planner must respond to those changes. They define a class of planners which, for lack of a better name, we will refer to as *active planners*: planners that both produce a plan and then actively alter that plan in the face of a changing environment.

Thus far, the stress on integrating planning and execution in this work has been on the issue of recovering from plans that unexpectedly fail. Little work has been done on the corollary concern of exploiting unexpected opportunities. In this paper, we will examine this concern and suggest a memory organization that addresses this issue. We will also describe two planners that are aimed at implementing the theory: TRUCKER, a planner developed at Chicago in the domain of pickup and delivery scheduling, and RUNNER, a new planner under development in the more general domain of errand running.

2 Opportunistic Memory

Our approach 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, as such, are suspended.
- Suspended goals are associated with elements of episodic memory that can be related to potential opportunities.
- These same memory structures are then used to parse the world so that the planner can make rou-

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

3 An Example

Before we get into any details, it is important to understand the type of behavior we want to capture. We will do this by looking at a simple example. Although this example is couched in terms of a story, we are interested in modeling the planning behavior described, not in understanding the text.

This example is taken from the RUNNER domain of errand running:

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 makes complete preplanning impossible. Second, the planner is able to stop planning for a goal before deciding exactly how to satisfy it. In effect, he is able to say "I don't have all the information or the time 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.¹

There is a final element to this example that does not lie quite so close to the surface: in order to decide to suspend planning for the goal to possess orange juice,

¹One 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 this 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.

John has to do some reasoning about what a plan for that goal entails. That is, he has to see that the goal is blocked by 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.

4 Opportunistic Planning

The idea of opportunistic memory 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. Their model, which they called *opportunistic planning*, consisted of a blackboard architecture [Lesser *et al.*, 1975] and planning *specialists* that captured planning information at many 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, there are some problems with this view of opportunism. Primarily, it 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 that execution. It is a model of opportunism at *planning time* rather than *execution time*. It 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. 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, the goal itself 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 inference directly off of it. 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 suspend 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 planner 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 says, this is hardly opportunism.

We share Birnbaum's and Collins' philosophical stance of trying to explain complex opportunistic behavior (even up to the subtle form of opportunism exhibited in Freudian slips). However, we disagree that this behavior results from goals constantly monitoring the world. We believe that indexing suspended goals is a far better explanation.

Birnbaum's arguments against indexing apply to models that separate a memory of goals from the planner's memory and understanding of the world. Under our model, however, there is no such distinction between types of memory. There is only one memory that is used to understand the world and to store suspended goals. Thus the act of recognition is the same as the act of indexing.

5 The Assumptions

The ideas of opportunistic memory in this paper exist in the context of a more general theory of planning and memory called *case-based planning* [Hammond, 1989; Kolodner *et al.*, 1985]. Case-based planning views planning as a memory task. Memories of past successes are used as prototypes for new plans, memories of past failures are used to avoid repeating the failures, and memories of past plan modifications are used to tailor old plans to new situations. This contrasts with the view of planning as a task of composition in which large plans are constructed piece-by-piece out of primitive actions.

There are certain assumptions that we make in case-based planning: first, we do not have a closed world. This limits the usefulness of preplanning and projection. Second, we have an incomplete and imperfect model of the world. This is assumed of not just the domain in general but also of the steps in plans that we use on a regular basis. In practice, this means that a planner cannot fully trust either its knowledge of the world's physics or its understanding of the world's current state. Third, we cannot do projection. That is, we cannot run complete simulations of our plans in order to tease out problems due to step interactions. This follows from our first and second assumptions. While we can project on the basis of *what the planner knows*, there is no guarantee that the projection will match what really happens.

We do not make these assumptions because we want to. We make them because we have been forced to. Such is the nature of real-world domains. But human planners are able to plan in complex domains, often with little knowledge of the true physics of those domains. This

makes us believe that it is possible for a planner to plan and do where it cannot understand.

Strangely enough, the need for opportunistic reasoning and the tools for developing it both grow out of these assumptions. Having assumed that the planner cannot completely model the effects of its own actions or predict those of other agents, we must provide it with some sort of mechanism that will allow it *react* to the world as well as *act* in it.

Likewise, because the planner cannot completely model the world, we must assume a mechanism for parsing or understanding the world that provides the planner with enough information to make execution-time decisions. In effect, the lack of a perfect world model requires that any planner must watch what it is doing as it is doing it. We will use this understanding process as the core of our opportunistic reasoning mechanism.

There are other ramifications to the use of a case-based planner, having to do with what constitutes a planning step and the nature of projection. The most important point we want to make in this section, however, is that a planner must understand and interact with the world in which it executes plans.

6 TRUCKER and RUNNER

Our opportunistic memory algorithm is only part of an overall approach to planning. Because of this, we must include some discussion of the two planners TRUCKER and RUNNER. Both planners are case-based in design and *active* in the sense that they test their plans through interaction with complex simulated worlds.

6.1 TRUCKER

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.

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 descriptors. 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 action agenda. 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 queue, 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 reoccurs, TRUCKER

searches its action queue for for the partner goal and uses the plan that it has created for the pair.

Even when a goal is placed on the action queue, TRUCKER treats it as though it were blocked. That is, 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.

6.2 RUNNER

The RUNNER project is the direct descendent of TRUCKER. RUNNER's domain is similar to the errand-running domain used by Hayes-Roth and Hayes-Roth to study opportunistic planning [1979]. Unlike TRUCKER, RUNNER has only a single agent to control, but RUNNER's agent is capable of a wider range of activity, and its domain allows for a richer set of goal interactions than does TRUCKER's. This change of domain was motivated by our desire to study a wider range of issues in both opportunism and execution-time failure recovery.

RUNNER's plans are MOP structures [Schank, 1982] that organize steps by their functions: establishing preconditions, side conditions, goal satisfaction, or postcondition cleanup. For each new goal being planned for, RUNNER checks for prototypical plan interactions that would allow it to merge preconditions, piggyback plans, or subsume them under single plans. For example, RUNNER looks for plans that have similar precondition steps in order to splice them together.

Like TRUCKER, RUNNER is designed to put off optimization until it obtains cues during execution that inform it of positive plan interactions. So RUNNER ends up suspending many of its goals in much the same way that TRUCKER does. The content of RUNNER's goals and plans, however, is more than just the simple pickup and delivery structures used by TRUCKER. As a result, RUNNER must do more of an analysis of what might constitute an opportunity to satisfy a goal than TRUCKER was forced to do.

RUNNER analyzes blocked plans and goals using two knowledge sources: precondition information associated with each plan and knowledge of opportunities with which it should be concerned. For example, in the case of John realizing that he cannot go to the grocery store to get orange juice, RUNNER would examine the preconditions (*e.g.*, the planner has money, is at the store, and has time) and look for specific conditions that might lead to an opportunity (*e.g.*, the planner has a missing resource, the planner is at the appropriate location or the planer has a time window). Each potential opportunity is evaluated as to how normative it is—with non-normative features ranking as better opportunities—as well as whether or not it is actually blocked in the current situation.

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.

7 Opportunism in TRUCKER and RUNNER

In both TRUCKER and RUNNER, blocked goals are associated with the elements of memory that will be active when opportunities to satisfy those goals arise. The main difference between the two planners is that establishing these conditions is relatively trivial in the TRUCKER domain—goals are of the same basic type. While the details of goal blockage and the analysis of opportunity differs between these planners, the basic approach to dealing with blocked goals is the same.

While these two planners are different, both require the ability to halt planning on a goal, suspend it, and then recall it when opportunities for execution present themselves. The approach used by TRUCKER and RUNNER to suspend and recall blocked goals has three basic parts.

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.

In our example of John and the orange juice, these steps translate into:

- 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.² 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.

²Other preconditions are noted as well (*e.g.*, having money and time) but these conditions are discarded because the first is a normative condition of the planner and the second is difficult to recognize.

- He then tests the preconditions on the plan and merges it into his current agenda.

7.1 Suspending blocked goals

When either TRUCKER or RUNNER finds a goal blocked, that goal is suspended. In TRUCKER, this requires associating a request with the memory token for the locations it involves. For RUNNER, this requires somewhat deeper reasoning about the various conditions that might provide opportunities to satisfy a blocked goal.

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 truck's 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.

When this is done, TRUCKER considers the goal blocked, and thus suspends it. To suspend a goal, TRUCKER marks its representation 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 doesn't have to do any more reasoning than this to identify good opportunities to satisfy the suspended goals.

RUNNER's domain and the goals it must plan for are more complex than TRUCKER's. As a result, it must reason far more than TRUCKER about the conditions that might constitute opportunities to satisfy a blocked goal.

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 obtain 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, RUNNER uses a taxonomy of *opportunity types* to derive the conditions that will serve as opportunities to satisfy the plan. This taxonomy guides RUNNER's search through the plan for appropriate precon-

ditions. 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 RUNNER asks of a plan:

- **Is there a special³ resource that is needed to run the plan?**
If so, associate the goal with the resource.
- **Is there a special tool that is needed to run the plan?**
If so, associate the goal with the tool.
- **Is there a special location associated with the plan?**
If so, associate the goal with the location.
- **Is there a special agent or skill associated with the plan?**
If so, associate the goal with the agent or skill.
- **Is there a specific time constraint associated with the plan?**
If so, associate the plan with 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, RUNNER associates the blocked goal to possess 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. RUNNER associates the goal with the least likely conditions with the hope that most of the other conditions will obtain when the suspended goal is activated. The other conditions are checked when the goal is recalled, but they are not linked to the suspended goal in memory.

7.2 Recalling suspended goals

Both TRUCKER and RUNNER tie execution of actions to locations, landmarks and addresses that they recognize in the world. Thus they must parse and interpret the objects in the world. It is during this parse that they both recognize and recall previously suspended goals.

A typical TRUCKER plan, when fully expanded, 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 pick-up 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

³A feature is "special" if it does not predictably reoccur as a product of the planner's policy decisions.

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.

RUNNER has a wider variety of planning options for any one goal. RUNNER can pick up orange juice at *any* grocery store, not just a particular one. As a result, we are using a much more general and realistic approach to parsing than we used in TRUCKER.

To deal with this, RUNNER is designed to use the DMAP parser [Martin & Riesbeck, 1986] as its recognition system. DMAP uses MOPs [Schank, 1982] to represent *concept sequences* to recognize concepts by recognition of their parts. DMAP uses a smart 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. In RUNNER, 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 our uses, we add a new type of link to DMAP. 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. So, 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, RUNNER uses this property of DMAP to recall goals associated with general characterizations of opportunities through the recognition of specific situations.

In both planners, the basic approach is the same. 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. And if any goal is found, it is considered a candidate for immediate satisfaction.

7.3 Exploiting the opportunities

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

Here again, the TRUCKER approach uses special-purpose techniques tailored to the domain. When a suspended goal is recalled by the planner, it attempts to find the best placement in the current route for the awakened request. 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.

Because this optimization is fairly time-consuming, TRUCKER saves the resulting route, so that it can reuse it when the same conjunct of goals arises again. We see the recognition of execution-time opportunity as a special case of expectation failure [Schank, 1982] and treat it as an indication of a gap in TRUCKER's knowledge base.⁴

RUNNER deals with a wider variety of goals than does TRUCKER. As a result, the special-purpose techniques used in TRUCKER are not applicable. Still, RUNNER's plan-merging techniques are not as general as those used by most planners. RUNNER falls back on the same techniques for merging plans that it uses for plan construction.

Steps in RUNNER's plans are specifically labeled as to function (PRECONDITION, GOAL-SATISFACTION, POST-CONDITION, CLEAN-UP). RUNNER uses these labels to search for specific ways to merge plans. Although RUNNER goes through the same process during preplanning, the task is somewhat easier when applied to a newly activated goal. Just as TRUCKER knows that it is already at the pickup location, RUNNER knows that the conditions associated with the activation of the suspended goal are already satisfied. If they were not, the suspended goal would not have been noticed in the first place.

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 pairs can be canceled together. The planner knows that the running plan must include the

⁴For a more detailed discussion of this type of learning, see [Hammond *et al.*, 1988].

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—have to 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 his trip home.

In both TRUCKER and RUNNER, the task of integrating recalled goals is essentially the same as the task of creating an initial plan. The only difference is that both planners have additional information about the conditions that currently hold and, as a result, are able to avoid consideration of the steps that establish those conditions. RUNNER is able to go beyond this and avoid consideration of steps involving postcondition cleanup.

8 Conclusions

In this paper, we have argued the need for an execution-time ability to recognize and exploit planning opportunities. Our goal was to present a model of opportunistic planning that provides this ability with little overhead.

We argue that our model of *opportunistic memory* does exactly that. By associating blocked goals with the same structures used to represent the planner's world, we are able to get activation of suspended goals as a by-product of the understanding process.

The process, implemented in TRUCKER and currently being expanded upon in RUNNER, requires three basic steps. Suspended goals are associated with the elements of memory that are related to potential opportunities. These memory structures are then used to parse the world during execution. As elements of memory are activated by conditions in the world, any goals associated with them are also activated and integrated into the current planning queue.

This combination of planning-time suspension and execution-time activation gives both TRUCKER and RUNNER the ability to halt consideration of a goal with the assurance that the goal will be brought back to mind when conditions change to allow its satisfaction.

9 Acknowledgements

I'd like to thank Mitch Marks and Tim Converse for their work on TRUCKER as well as Jeff Berger, Neil Hurwitz, and Greg Hajek for their comments and conversation. I'd also like to thank Larry Birnbaum and Gregg Collins for letting me steal these ideas in the first place.

10 References

[Agre & Chapman, 1987] Agre, P.E. & Chapman, D. Pengi: An implementation of a theory of activity. In: *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, Washington, July 1987.

[Alterman, 1985] Alterman, R. Adaptive planning: Refitting old plans to new situations. In: *Proceedings of*

the Seventh Annual Conference of the Cognitive Science Society, Irvine, California, August 1985.

[Birnbaum, 1986] Birnbaum, L. *Integrated processing in planning and understanding*. Yale Technical Report #480, 1986.

[Birnbaum & Collins, 1984] Birnbaum, L. & Collins, G. Opportunistic planning and Freudian slips. In: *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, June 1984.

[Chapman, 1987] Chapman, D. *Planning for Conjunctive Goals*. Technical Report TR 802, MIT Artificial Intelligence Laboratory, 1985.

[Firby, 1987] Firby, R.J. An investigation into reactive planning in complex domains. In: *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, Washington, July 1987.

[Georgeff & Lansky, 1987] Georgeff, M.P. & Lansky, A.L. Reactive reasoning and planning. In: *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, Washington, July 1987.

[Hammond, 1989] Hammond, K. *Case-based planning: Viewing planning as a memory task*. Academic Press, Cambridge, Massachusetts, 1989.

[Hammond et al., 1988] Hammond, K., Converse, T., & Marks, M. Learning from opportunities: Storing and reusing execution-time optimizations. In: *Proceedings of the Seventh National Conference on Artificial Intelligence*, St. Paul, Minnesota, August 1988.

[Hayes-Roth & Hayes-Roth, 1979] Hayes-Roth, B., & Hayes-Roth, F. A cognitive model of planning. *Cognitive Science* 2:275-310.

[Kolodner et al., 1985] Kolodner, J.L., Simpson, R.L., & Sycara-Cyranski, L. A process model of case-based reasoning in problem solving. In: *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, California, August 1985.

[Lesser et al., 1975] Lesser, V.R., Fennell, R.D., Erman, L.D., & Reddy, D.R. Organization of the Hearsay-II speech understanding system. In: *IEEE Transactions on Acoustics, Speech and Signal Processing*. ASSP-23, 1975, 11-23.

[Martin & Riesbeck, 1986] Martin, C., & Riesbeck, C. Uniform parsing and inferencing for learning. In: *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, Pennsylvania, August 1986.

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

[Schank & Abelson, 1977] Schank, R. & Abelson, R. *Scripts, plans, goals and understanding*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.

[Simmons & Davis, 1987] Simmons, R., & Davis, R. Generate, test, and debug: Combining associational rules and causal models. In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987.