Integrating Planning and Acting in a Case-Based Framework.*

Kristian Hammond Timothy Converse Charles Martin

The University of Chicago
Department of Computer Science
Artificial Intelligence Laboratory
1100 East 58th Street
Chicago, IL 60637
ai@tartarus.uchicago.edu

Abstract

This paper presents an outline of a theory of *agency* that seeks to integrate ongoing understanding, planning and activity into a single model of representation and processing. Our model of *agency* rises out of three basic pieces of work: Schank's structural model of memory organization (Schank, 1982), Hammond's work in case-based planning and dependency directed repair (Hammond, 1989d), and Martin's work in Direct Memory Access Parsing (Martin 1990). We see this paper as a first step in the production of a memory-based theory of *agency*: the active pursuit of goals in the face of a changing environment, that can exist within the computational constraints of a computer model.

Planning as Understanding

Research in planning has recently made a dramatic change in course. Planning researchers have begun to acknowledge that the world is too complex and uncertain to allow a planner to plan exhaustively for a set of goals prior to execution (Chapman, 1985). More and more, the study of planning is being cast as the broader study of planning, action and understanding (Agre and Chapman, 1987, and Alterman, 1986).

The particular cast of this relationship that we have been studying is a view of planning as embedded within an understanding system connected to the environment. The power of this approach lies in the fact that it allows us to view the planner's environment, plan selections, decisions, conflicts and actions through the single eye of situation assessment and response. Because of our further commitment to the use of episodic memory as the vehicle for understanding, it also provides us with a powerful lever on the problem of learning from both planning and execution. In this paper, we draw an outline of *agency*, our model of the relationship between planning and action.

*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.

Memory and Agency

Our model of planning and understanding rises out of three pieces of work: Schank's structural model of memory organization (Schank, 1982), our own work in case-based planning and dependency directed repair (Hammond, 1986), and the work of Martin and Riesbeck in Direct Memory Access Parsing (Martin 1989). Our model has been articulated in two programs, TRUCKER and RUNNER (Hammond, Converse, and Marks, 1988 and Hammond, 1989).

The model was first developed to deal with the problem of recognizing execution-time opportunities in the context of a resource-bound agent that is forced to suspend planning in order to attend to execution (Hammond, 1989). The goal of this model was to capture the ability of an agent to suspend goals, yet still recognize execution-time opportunities to satisfy them.

To accomplish this goal, we use a single set of memory structures both to store suspended goals and to understand the agent's circumstances in the world. In response to a blocked goal, an agent's first step is to do a planning-time analysis of the conditions that would favor the satisfaction of the goal. The agent then *suspends* the goal in memory, indexed by a description of those conditions. For example, a goal to buy eggs that was blocked during planning would be placed in memory associated with the condition of the agent being at a grocery store.

During execution, the agent performs an ongoing "parse" of the world in order to recognize conditions for action execution. Following DMAP (Martin, 1989), this parse takes the form of passing markers through an existing episodic memory. Because suspended goals are indexed in the memory used for understanding the world, the goals are activated when the conditions favoring their execution are recognized. Once active, the goals are then reevaluated in terms of the new conditions. Either they fit into the current flow of execution or they are again suspended.

We called the initial model *opportunistic memory* because the agent's recognition of opportunities depends on the nature of its episodic memory structures. Having turned to the broader issues of integrating planning and action, we now refer to our work

as the study of *agency*.

Initial Results

Our initial implementation of opportunistic memory, TRUCKER, exhibited exactly the behavior we wanted. A combined scheduling planner and executive, TRUCKER was able to suspend blocked goals and then recognize later opportunities to satisfy them. It also learned plans for goal combinations that it determined would occur again. The recognition of opportunity and the resulting learning of specific optimizations rose naturally out of the agent's ongoing understanding of its environment.

Our model of planning is case-based. Both of our planners, TRUCKER and RUNNER, plan by recalling existing plans that are indexed by a set of currently active goals. Our initial model of this indexing (Hammond, 1986) was based on the notion that a planner could amortize its planning efforts by caching plans in memory under descriptions of the goals that they satisfied and the problems (in the sense of interactions between steps of a plan) that they avoided. In TRUCKER, we worked on the idea that this method of indexing could itself be cast as a problem of characterizing the situations in the world under which a plan could be run. Retrieval then became a process of recognition, similar to that used in the understanding of the world. The result of this was that plans were cached in the same memory organization used to suspend blocked goals and to understand the changing states of the world.

In TRUCKER, and later in RUNNER, we tried to address the specific problem of recognizing execution-time opportunities. In the process, however, we actually built a general mechanism to control planning and action. That is, to achieve the desired execution-time opportunism, we had to build a general model of planning and action based on embedding the knowledge of plans and goals, as well as the control of action itself, in a memory-based understanding system.

A Model of Agency

We use the term *agency* to comprise the spawning of goals, selection of plans, and execution of actions. Our process model of agency is based on Martin's DMAP understander as well as its antecedent, Schank's *Dynamic Memory*. DMAP uses a memory organization defined by part/whole and abstraction relationships. Activations from environmentally supplied features are passed up through abstraction links and predictions are passed down through the parts of partially active concepts. Subject to some constraints, when a concept has only some of its parts active, it sends predictions down its other parts. When activations meet existing predictions, the node on which they meet becomes active. Finally, when all of the parts of a concept are activated, the concept itself is activated.

Direct Memory Access Processing

The DMAP architecture provides a computational mechanism for specifying and applying domain-dependent information to a general memory search process. Concepts are represented in the abstraction and packaging hierarchies familiar to artificial intelligence researchers. This hierarchy is not only a store of content, but also provides the structure for the indexing and application of process knowledge.

Operational features

In general terms, we can associate *operational* features of the world with the memory structures to which they refer. In language understanding, such features may include words or phrases; for planning, they may comprise detectable states of the world. We call these features *operational* because they can be derived from the input with little reliance on the use of memory. In general, an operational feature cannot be a necessary or sufficient condition for recognizing any particular memory structure.

For example, the RUNNER project (Hammond, 1989) associates "being near the coffee pot" and "the coffee pot is empty" with the FILL-POT action of the MAKE-COFFEE plan. These two conditions are presumed to be easily detectable features of the environment, relying on the planner's location and a capacity for object identification. These features are in no sense equivalent to the FILL-POT action but they are *operational* in that they serve to recognize the applicability of that action to the achievement of the MAKE-COFFEE plan.

Memory search strategies

Operational features are the primitive elements out of which memory search strategies are composed. A search strategy is the specification *relative to a memory structure* of how concepts in memory may become active. In general, a search strategy specifies other memory structures and features along with some ordering information to direct the search. Because memory search involves recognizing a sequence of concepts, we call these search strategies *concept sequences*.

For example, in language analysis, the original domain of the DMAP work, two simple concept sequences associated with the memory structure for the communication event MTRANS are the following.

> { (*actor*) "says" (*info*) }, representing that an MTRANS concept can be recognized as a result of recognizing, in sequence, a concept filling *actor* role of the MTRANS, the lexical item "says" (an operation feature for a language analyzer), and a concept filling the *info* role of the MTRANS.
>
> { "I" } → (*actor*), representing that the *actor* of an MTRANS concept can be recognized as a result of recognizing the referring lexical item "I".

The organization of memory

Because DMAP associates search strategies with specific concepts in memory, those concepts serve as the points of organization for expectations generated by the system. For example, RUNNER places expectations for the FILL-POT action on the *memory structures* for "being near the coffee pot" and "the coffee pot is empty".

Because these memory structures *already exist* in the system and serve as the organizing point for these expectations, the memory itself provides the medium for disambiguating which expectations are relevant in any situation. Expectations from existing memory structures prime other existing memory structures. Activation of an existing memory structure allows the system to focus on those and only those expectations that primed that particular memory structure.

Accomodating action

To accommodate action, we have added the notion of PERMISSIONS. PERMISSIONS are handed down the parts of plans to their actions. The only actions that can be executed are those that are PERMITTED by the activation of existing plans. Following McDermott (McDermott, 1978), we have also added POLICIES. POLICIES are statements of ongoing goals of the agent. Sometimes these take the form of maintenance goals, such as "Glasses should be in the cupboard." or "Always have money on hand." The only goals that are actively pursued are those generated out of the interaction between POLICIES and environmental features. We would argue that this is, in fact, the only way in which goals can be generated.

Most of the processing takes the form of recognizing circumstances in the external world as well as the policies, goals and plans of the agent. The recognition is then translated into action through the mediation of PERMISSIONS that are passed to physical as well as mental actions.

Goals, plans, and actions interact as follows:

- Features in the environment interact with POLICIES to spawn goals.

  For example, in RUNNER, the specific goal to HAVE COFFEE is generated when the system recognizes that it is morning. The goal itself rises out of the recognition of this state of affairs in combination with the fact that there is a policy in place to have coffee at certain times of the day.

- Goals and environmental features combine to activate plans already in memory.

  Any new MAKE-COFFEE plan is simply the activation of the sequence of actions associated with the existing MAKE-COFFEE plan in memory. It is recalled by RUNNER when the HAVE-COFFEE goal is active and the system recognizes that it is at home.

- Actions are **permitted** by plans and are associated with the descriptions of the world states appropriate to their performance. Once a set of features has an action associated with it, that set of features (in conjunct rather than as individual elements) is now predicted and can be recognized.

  Filling the coffee pot is **permitted** when the MAKE-COFFEE plan is active; it is associated with the features of the pot being in view and empty. This means not only that the features are now predicted but also that their recognition will trigger the action.

- Actions are specialized by features in the environment and by internal states of the system. As with Firby's RAPs (Firby, 1989), particular states of the world determine particular methods for each general action.

  For example, the specifics of a GRASP would be determined by information taken from the world about the size, shape and location of the object being grasped.

- Action level conflicts are recognized and mediated using the same mechanism that recognizes information about the current state of the world.

  For example, when two actions are active (such as filling the pot and filling the filter), a mediation action selects one of them. During the initial phases of learning a plan, this can in turn be translated into a specialized recognition rule which, in the face of a conflict, will always determine the ordering of the specific actions.

- Finally, suspended goals are associated with the descriptions of the states of the world that are amenable to their satisfaction.

  For example, the goal HAVE-ORANGE-JUICE, if blocked, can be placed in memory, associated with the conjunct of features that will allow its satisfaction, such as being at a store, having money and so forth. Once put into memory, this conjunct of features becomes one of the set that can now be recognized by the agent.

The Study of Agency

We do not see this model as a solution to the problems of planning and action. Instead, we see this as a framework in which to discuss exactly what an agent needs to know in a changing world. Advantages of this framework include:

1. A unified representation of goals, plans, actions and conflict resolution strategies.
2. Ability to learn through specialization of general techniques.
3. A fully declarative representation that allows for meta-reasoning about the planner's own knowledge base.
4. A simple marker-passing scheme for recognition that is domain — and task — neutral.

5. Provision for the flexible execution of plans in the face of a changing environment.

The basic metaphors of action as permission and recognition, and planning as the construction of descriptions that an agent must recognize prior to action, these fit our intuitions about agency. Under this metaphor, we can view research into agency as the exploration of the situations in the world that are valuable for an agent to recognize and respond to. In particular, we have examined and continue to explore content theories of:

- The conflicts between actions that rise out of resource and time restrictions as well as direct state conflicts and the strategies for resolving them.
- The types of physical failures that block execution and their repairs.
- The types of knowledge-state problems that block planning and their repairs.
- The circumstances that actually give rise to goals in the presence of existing policies.
- The possible ways in which existing plans can be merged into single sequences and the circumstances under which they can be applied.
- The types of reasoning errors that an agent can make and their repairs.
- The trade-offs that an agent has to make in dealing with its own limits.
- The different ways in which a goal can be blocked and the resulting locations in memory where it should be placed.

Our goal is a content theory of agency. The architecture we suggest is simply the vessel for that content.

RUNNER

Most of our activity in studying this architecture has been within the context of the RUNNER system. The RUNNER project is aimed at modeling the full spectrum of activity associated with an agent—goal generation, plan activation and modification, action execution, and resolution of plan and goal conflict—not just the more traditional aspect of plan generation alone.

RUNNER's world

The agent in RUNNER currently resides in a simulated kitchen, and is concerned with the pursuit of such goals as simulated breakfast and coffee. Such commonplace goals and tasks interest us in part because they are repetitive and have many mutual interactions, both negative and positive. We are interested in how plans for recurring conjuncts of goals may be learned and refined, as part of view of domain expertise as knowledge of highly specific and well-tuned plans for the particular goal conjuncts that tend to co-occur in the domain (Hammond, Converse, and Marks, 1988). We are also interested in the issue of exactly how these plans can be used in the guidance of action.

RUNNER's Representation

The knowledge and memory of the agent is captured in the conjunction of three types of semantic nets, representing knowledge of goals, plans and states. Nodes in these networks are linked by abstraction and packaging links, as in DMAP. In addition, we propose an additional SUSPEND link, which connects suspended goals to state descriptions that may indicate opportunities for their satisfaction. For example, the goal to have eggs would be suspended in association with the description of the agent being at a grocery store. In addition to being passed to abstractions of activated concepts, activation markers are always passed along SUSPEND links.

In general, the only other way in which these nets are interconnected is via *concept sequences*. A node may be activated if all of the nodes in one of its concept sequences is activated – a concept sequence for a given node can contain nodes from any of the parts of memory. The following is a partial taxonomy of the types of concept sequences we currently allow:

- Activation of a goal node can activate a node representing a default plan.
- Activation of a plan node and some set of state nodes can activate a further specialization of the plan.
- Activation of a goal node and some set of state nodes can activate a further specialization of the goal.
- Activation of any state node that has a SUSPEND link will activate the associated goal.

An Example: Making Coffee

The above discussion of representation may make more sense in the context of an example, currently implemented in RUNNER, of how a particular action is suggested due to conjunction of plan activation and environmental input.

One of the objects in RUNNER's simulated kitchen is a coffeemaker, and one of the plans it has available is that of making coffee with this machine. This plan involves a number of subsidiary steps, some of which need not be ordered with respect to each other. Among the steps that are explicitly represented in the plan are: fetching unground beans from the refrigerator, putting the beans in the grinder, grinding the beans, moving a filter from a box of filters to the coffeemaker, filling the coffeemaker with water from the faucet, moving the ground beans from the grinder to the coffeemaker, and turning the coffeemaker on.

The subplans of the coffee plan are associated with that plan via packaging links. In this implemented example, the agent starts out with a node activated which represents knowledge that it is morning. This

in turn is sufficient to activate the goal to have coffee (this is as close as the program comes to a theory of addiction). This goal in turn activates a generic plan to have coffee. This turns out to be nothing but an abstraction of several plans to acquire coffee, only one of which is the plan relevant to our kitchen:

```
Sending initial activations to memory
sending activation marker to [morning]
Activating concept: [morning]
concept sequence ([morning])
for node [GOAL: drink-coffee] is completed.
sending activation marker to
   [GOAL: drink-coffee]
Activating concept: [GOAL: drink-coffee]
Asserting new goal: [GOAL: drink-coffee]
sending activation marker to
   [PLAN: coffee-plan]
Node [PLAN: coffee-plan] has both permission
and activation:
   ((MARKER [GOAL: drink-coffee]))
    (TOP-LEVEL-PLAN)
Activating concept: [PLAN: coffee-plan]
Asserting new plan: [PLAN: coffee-plan]
Plan has no steps -- insufficiently specific
```

"Visual" input, in terms of atomic descriptions of recognizable objects and their proximities, is passed to memory. For example, the agent "sees" the following visual types:

```
countertop, white wall, box of filters
```

Among sets of possible visually recognized objects are concept sequences sufficient for recognition that the agent is in the kitchen. The recognition of the white wall and the countertop completes one of these sequences. The "kitchen" node in turn passes activation markers to its abstractions, activating the node corresponding to the agent being at home:

```
-----------------------------------------------
Straight ahead I see:
 a countertop, up close;
 a countertop, fairly close;
 a green square filter-box, up close;
 a countertop, fairly close;
 a countertop, far away;
 a white wall, far away;
 a countertop, fairly close;
 a countertop, far away;
 a white wall, far away
To the left is a countertop, up close
To the right, there's a countertop, up close
Straight ahead, there's a countertop, up close
-----------------------------------------------
MEMORY:
Active plans: coffee-plan
sending activation marker to [wall]
Activating concept: [wall]
sending activation marker to [filter-box]
Activating concept: [filter-box]
sending activation marker to [countertop]
Activating concept: [countertop]
concept sequence ([wall] [countertop])
for node [in-kitchen] is completed.
sending activation marker to [in-kitchen]
Activating concept: [in-kitchen]
sending activation marker to [at-home]
Activating concept: [at-home]
```

The activation of this node in conjunction with the activation of the generic coffee goal completes the concept sequence necessary for the goal for making coffee at home, which in turn activates the default plan for that goal. In this way a specialized plan is chosen in response to a conjunction of a recognized state and a more generic goal:

```
MEMORY:
concept sequence
    ([GOAL: drink-coffee] [at-home])
for node
     [GOAL: drink-coffee-at-home] is completed.
sending activation marker to
     [GOAL: drink-coffee-at-home]
Activating concept:
     [GOAL: drink-coffee-at-home]
Asserting new goal:
     [GOAL: drink-coffee-at-home]
sending activation marker to
     [PLAN: make-coffee-at-home]
Node [PLAN: make-coffee-at-home]
has both permission and activation:
     ((MARKER [GOAL: drink-coffee-at-home]))
     (TOP-LEVEL-PLAN)
Activating concept:
     [PLAN: make-coffee-at-home]
```

The activation of the coffee-plan causes permission markers to be sent down packaging links to the nodes representing the parts of the plan. The activation of the other object concepts from the "visual" input in turn have sent activation markers to the actions that contain them in their concept sequences. Among these is the plan step for taking a filter from the box and installing it in the coffeemaker, which is activated by seeing box of filters itself. In this way a sub-plan is suggested by the intersection of permission from its parent plan and cues from the environment that indicate that it is easily satisfiable:

```
Asserting new plan:
     [PLAN: make-coffee-at-home]
Sending permissions to steps of plan
Sending permission markers from
     [PLAN: make-coffee-at-home]
to steps
     FILL-CARAFE
     PUT-BEANS-IN-GRINDER
     MOVE-GROUNDS-TO-COFFEE-MAKER
     TURN-ON-COFFEE-MAKER
     GRIND-BEANS
     PUT-IN-FILTER
     GET-COFFEE-BEANS
concept sequence
     ([filter-box]
      [PLAN: make-coffee-at-home])
for node [PLAN: put-in-filter] is completed.
sending activation marker to
     [PLAN: put-in-filter]
Node [PLAN: put-in-filter]
has both permission and activation:
     ((MARKER ([filter-box]
               [PLAN: make-coffee-at-home])))
```

```
     ((MARKER [PLAN: make-coffee-at-home]))
Activating concept:
     [PLAN: put-in-filter]
Asserting new plan: [PLAN: put-in-filter]
Sending permissions to steps of plan
Sending permission markers from
     [PLAN: put-in-filter]
to steps
     PUT-FILTER-IN-COFFEEMAKER
     GET-FILTER
concept sequence
     ([filter-box]
      [PLAN: put-in-filter])
for node [PLAN: get-filter] is completed.
sending activation marker to
     [PLAN: get-filter]
Node [PLAN: get-filter]
has both permission and activation:
     ((MARKER ([filter-box]
               [PLAN: put-in-filter])))
     ((MARKER [PLAN: put-in-filter]))
Activating concept: [PLAN: get-filter]
```

After another level of passing permission markers to sub-plans, the process "bottoms out" in the suggestion of the primitive action of picking up the box of filters. With no suggestions to the contrary, the action is taken:

```
Asserting new plan:
     [PLAN: get-filter]
Sending permissions to steps of plan
Sending permission markers from
     [PLAN: get-filter]
to steps
      TAKE-OUT-FILTER
      PICK-UP-BOX
      LOOK-FOR-FILTER-BOX
concept sequence
     ([filter-box] [PLAN: get-filter])
for node [PLAN: pick-up-box] is completed.
sending activation marker to
     [PLAN: pick-up-box]
Node [PLAN: pick-up-box]
has both permission and activation:
     ((MARKER ([filter-box] [PLAN: get-filter])))
     ((MARKER [PLAN: get-filter]))
Activating concept: [PLAN: pick-up-box]
Suggesting action: (GRASP 'FILTER-BOX)
------------------------------------------------
ACTION:
Performing action: (GRASP 'FILTER-BOX)
------------------------------------------------
To the left is a countertop, up close
To the right, there's a countertop, up close
Straight ahead, there's a countertop, up close
Result of action: I'm holding on to a filter-box
------------------------------------------------
```

The final action is chosen both on the basis of active plans and goals, and in response to the immediate circumstances in which the agent finds itself. Given a change in either the top-down guidance or the bottom-up recognition, the selection of plan and action will change in response.

Conclusion

We've presented a sketch of an architecture for memory that we believe will be of use in exploring various issues of opportunism and flexible plan use. We do not view the architecture as a solution to the problems of interest, but instead as a framework that may be useful in exploring content theories of plan types, action suggestion and arbitration. As we said before, our goal is a content theory of agency. The architecture we suggest is simply the vessel for that content.

Acknowledgements

This work is the first stage of a collaborative effect at the University of Chicago. As such this paper owes much to Christopher Owens and Phil Agre as well as to Mitchell Marks and the others students at the University of Chicago's Artificial Intelligence Laboratory.

References

Phil Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth Annual Conference on Artificial Intelligence*, pages 268–72. AAAI, 1987.

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

David Chapman. Nonlinear planning: A rigorous reconstruction. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1022–24. IJCAI, 1985.

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

Kristian J. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*, volume 1 of *Perspectives in Artificial Intelligence*. Academic Press, 1989.

Kristian J. Hammond. Opportunistic memory. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. IJCAI, 1989.

Kristian J. Hammond. Explaining and repairing plans that fail. *Artifical Intelligence Journal*, In Press.

Kristian J. Hammond, Tim Converse, and Mitchell Marks. Learning from opportunities: Storing and reusing execution-time optimizations. In *AAAI 1988*, pages 536–40. AAAI, 1988.

Charles E. Martin. *Direct Memory Access Parsing*. PhD thesis, Yale University Department of Computer Science, 1989.

D. McDermott. Planning and acting. *Cognitive Science*, 2, 1978.

Roger C. Schank. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People*. Cambridge University Press, 1982.