Ad-Hoc, Fail-Safe Plan Learning

Roland Zito-Wolf
Richard Alterman

Computer Science Department
Brandeis University
Waltham MA 02254

Abstract

Artificial Intelligence research has traditionally treated planning, execution and learning as independent, sequential subproblems decomposing the larger problem of intelligent action. Recently, several lines of research have challenged the separation of planning and acting. This paper suggests that integration with planning and acting is also important for learning. We present an integrated system SCAVENGER combining an adaptive planner with an ad-hoc learner. Situated plans are retrieved from memory; adaptation during execution extends these plans to cope with contingencies that arise and to tease out descriptions of situations to which these plans pertain. These changes are then integrated into the plan and incorporated into memory. Every situation of action is an opportunity for learning. Adaptive planning makes learning *fail-safe* by compensating for imperfections and omissions in learning and variability across situations. We discuss a learning example in the domain of mechanical devices.

1 Introduction

Traditional models of planning in artificial intelligence have two key features. First, planning is *ahistoric*: the planner confronts each task in isolation. Plans are constructed from scratch using a limited repertoire of operators and a domain model defining their effects and interactions. Second, planning is *detached from acting*. The plan is constructed first, then given to an independent execution monitor that performs the specified operations. These assumptions help decompose the planning task, but unfortunately introduce other problems. Experience is ignored, leading to redundant planning. Separate execution makes it difficult to adequately address run-time contingencies and uncertainty. This critique of traditional planning models arose from work on adaptive planning[Alterman, 1988], case-based planning[Kolodner and Simpson, 1989; Hammond, 1986], reactive planning[Firby, 1987; Georgeff and Lansky, 1987], and situated activity[Suchman, 1987; Agre and Chapman, 1987; Agre, 1988].

Adaptive planning [Alterman, *ibid.*] was an early effort to address these problems. An adaptive planner is a commonsense planner. It retrieves from memory a plan that fits the situation, and adapts it during engagement – while focused on the task and immersed in the details of the situation – in conjunction with its developing interpretation of the situation.

Adaptive planning provides a congenial environment for learning. Adaptation is a rich source of ideas for plan modification; every situation of engagement is an opportunity to learn. Learning, planning, and action are integrated: planning and acting in particular situations generate data to be learned, while learning is distributed across many situations. Learning extends plans to cope with new contingencies that may arise and teases out descriptions of situations to which a given plan pertains.

This paper presents a system SCAVENGER combining an adaptive planner with a learner. SCAVENGER continually refines its knowledge with experience. Faced with a domain for which the system possesses no complete theory, what else can it do? Learning is *ad-hoc* because what is learned depends on both the particular situation and the learner's current knowledge; knowledge is forged by experience, grounded rather than abstract. We use the term *fail-safe* to suggest not infallibility but rather that adaptation functions during execution, compensating for imperfections and omissions in learning, and insulating the learner from small variations across otherwise similar situations. Learning is the accumulation of detailed experience that can be adapted to future situations.

SCAVENGER explicitly frames the learning problem to tolerate inconsistency and revision. This is an atypical assumption for planning and learning systems. Planners in the tradition descending from STRIPS[Fikes and Nilsson, 1971; Chapman, 1987] require a strong domain theory to predict the consequences of actions. Recent EBL techniques[Mitchell *et al.*, 1986; DeJong and Mooney, 1986; Mooney, 1988; Minton *et al.*, 1989] require a strong domain theory to guide explanation and constrain generalization. Pazzani[1988] shows that a domain theory for EBL can be constructed inductively; however, the implications of basing EBL on a changing or inconsistent theory are unclear. Our work assumes that knowledge of the domain is inevitably incomplete.

SCAVENGER differs from other memory-based planners that learn, such as MEDIATOR[Kolodner and Simpson, 1989] and CHEF[Hammond, 1986], largely because plan adaptation occurs throughout the period of engagement. For example, contrast the frameworks of CHEF and SCAVENGER. Unlike CHEF, SCAVENGER does not require a complete domain theory or abstract repair schemata. Where CHEF learns about plans, SCAVENGER learns about plans and their situations-of-use. SCAVENGER's adaptations are motivated by the resources available in the situation, rather than derived primarily from operationalization of the domain theory. SCAVENGER is able to acquire many independent items of knowledge in the course of a single trial.

2 SCAVENGER

The FLOABN project[Alterman and Zito-Wolf, forthcoming] addresses planning and plan acquisition in the the domain of everyday mechanical devices such as telephones, clocks and VCRs. This domain is challenging because there is much variability among devices and because only a minimal domain theory is available. In compensation, the execution environment is benign (errors in operation are normally recoverable) and there are multiple opportunities for interaction with each device. SCAVENGER, the core of FLOABN, is an adaptive planner coupled with an ad-hoc learner. Other modules of FLOABN are concerned with skimming instructions for adaptation ideas and simulating in detail the perceptual and effective interactions between agent and device. We are prototyping SCAVENGER in Quintus MacProlog.

A problem is posed to SCAVENGER by specifying a goal and a situation. SCAVENGER's semantic memory defines concepts; its episodic memory contains plans and expectations (see Figure 1). A simulator models the outcome of SCAVENGER's actions in the world. SCAVENGER has no access to the internals of this model; the outcomes of actions in the simulated world may differ from the system's expectations.

Adaptation is driven by *situation differences*: discrepancies between what is expected at each plan step and what is actually observed. Situations may differ in goals, preconditions, and outcomes; further detail can be found in Alterman[1988]. When such a difference is detected, SCAVENGER searches for an adaptation that will either account for or repair it. Adaptations are suggested using general and domain-specific knowledge and related known plans. Adaptation suggestions can also come from other modules of FLOABN; we will not discuss those here. Plans are modified via some combination of inserting, deleting, reordering, substituting and modifying steps. SCAVENGER can also satisfy missing preconditions and outcomes by substitution of similar features or outcomes. This is in many cases more natural than modifying an entire step; more importantly, this allows SCAVENGER to create new steps by modifying known ones. SCAVENGER keeps track of the adaptations made; afterwards, successful adaptations are used to *elaborate* the plan. Elaboration conditionalizes the plan so that it will act appropriately in situations resembling the newly learned situation. Execution of the plan in such circumstances will then entail little or no deliberation.

We demonstrate this process with an example of SCAVENGER learning to use a (simulated) touch-tone phone. SCAVENGER adapts its dial-phone procedure using a variety of clues: the common features shared by touch-tone and dial phones, knowledge of which features are pertinent in a situation, knowledge of which features participate causally in the plan, knowledge about the function of individual steps *within* the plan, and background knowledge about the world. SCAVENGER's learning module then incorporates the successful adaptations into the existing plan and concept structures. This results in the creation of a new device category for touch-tone phones, a generalized category subsuming both phone types, and plan steps specific to touch-tone phones. Lastly, reorganization of the telephone plan abstracts the step of dialing a number into separate plans for dial and touch-tone phones.

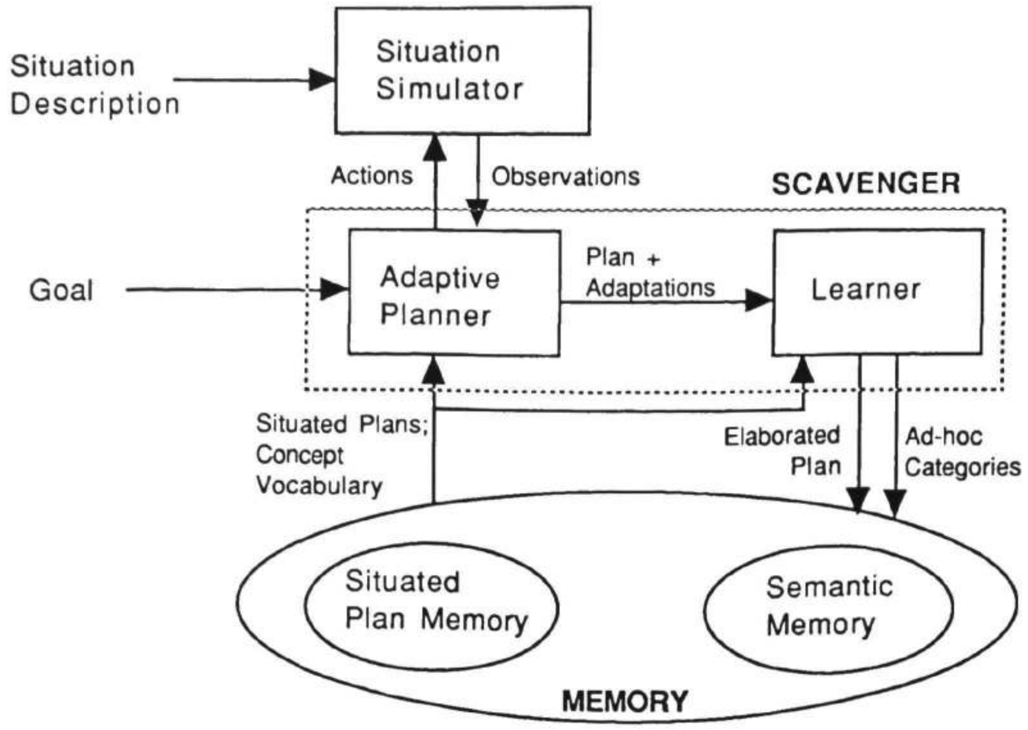

Figure 1: SCAVENGER system diagram

3 An Example of Ad-Hoc Learning

SCAVENGER is given the goal of placing a telephone call in a situation containing the following items: a touch-tone telephone and a personal computer. In predicate-calculus notation:

```
exist(tt_telephone)
parts_of(tt_telephone,[keyboard,telephone_rcvr,
  flexible_cord,receiver_cradle]),
exist(apple_computer)
parts_of(apple_computer,[keyboard,screen,
  slot,on_off_switch]),
exist(desired_number, value,[7,3,6,2,7,0,3])
```

The plan found in memory for this goal is **telephone_plan**, which presumes a dial phone. Part of **telephone_plan** is shown below. The first clause indicates that to make a call one needs to be at a telephone, pick up the receiver, check dial tone, dial, wait for the ring, and wait for an answer. The **dial** step is further decomposed. For each step, the expected preconditions (**prec**) and outcomes (**outc**) are declared. **Role** declarations specify the expected types of the items referenced by a step.

```
steps(telephone_plan,[pick_up_rcvr,
  hear_dial_tone,dial,wait_for_ring,answer,talk]),
role(telephone_plan,desired_number,
  telephone_number),

prec(pick_up_rcvr,[exist,dial_telephone]),
prec(pick_up_rcvr,[exist,telephone_rcvr]),
outc(pick_up_rcvr,[hear,dial_tone]),
reason(pick_up_rcvr,dial),
step_type(hear_dial_tone,observation),
prec(hear_dial_tone,[hear,dial_tone]),

steps(dial,[find_first_digit,select_digit,
  find_digit_on_dial,dial_one_digit,
  hear_clicks,dial_loop_test]),
prec(dial,[hear,dial_tone]),
outc(dial,[have,dialed]),
-- remaining steps and substeps omitted --
```

SCAVENGER has background knowledge about dial telephones, and a small taxonomy of sounds:

```
parts_of(dial_telephone,[dial,telephone_rcvr,
  flexible_cord,receiver_cradle]),
index(dial_telephone,telephone_rcvr),
index(dial_telephone,dial),

isa(tone, sound),
isa(clicking, sound)
```

4 Plan Adaptation

Adaptation of **telephone_plan** proceeds as follows. Execution begins; the first situation difference detected is a failing precondition **exist(dial_telephone)** in step **pick_up_rcvr**. SCAVENGER adapts this step by finding a substitute for **dial_telephone** which exists in the current situation, namely, **tt_telephone**, and modifying the step to use it. The substitution is justified by similarity of type and features.

Another substitution occurs later in the example, where SCAVENGER needs to re-interpret the situation to account for a failing outcome. The step **hear_clicks** represents the act of confirming that a digit has been transmitted (dialed) by listening for the clicking sound in the receiver. In this situation a tone is heard instead. The system justifies the substitution of TONE for CLICKING on the basis that both features are of type SOUND, but more importantly, by the temporal relation of the sound and the dialing action.

Similarity comparisons are based on class relations, as in **hear_clicks**, plus similarity of features, as in **pick_up_rcvr**. Similarity by class is measured by the distance of the items being compared along ISA relations, if present. Feature similarity is based on the features (for example, PARTS) associated with the desired item. Missing or excess features are not counted. In terms of Tversky's model of similarity[Tversky, 1977],

$$S(Sample, Pattern) = F(Sample \cup Pattern)$$

Our F allots extra weight to salient (causally relevant) features. Also, we do not require that features match exactly, so that a simple set-union model is inappropriate. The value v_i contributed to a match by pattern feature i is 1 if there exists a sample feature matching i; else it is the similarity of the best-matching sample feature. The similarity of any pair of features is computed by recursive application of the similarity function. Since recursive similarity computations are computationally explosive, each comparison is given a *search horizon* limiting the level of detail the comparison will explore. The horizon is reduced with each recursion. (Better yet would be horizons based on how significant the result might be in the context of the overall match.)

The substitution of **tt_telephone** for **dial_telephone**, for example, is justified primarily by the presence of a particular salient part, the **telephone_rcvr**. Salience is currently marked explicitly in the knowledge base by clauses of the form **index**(*object, feature*). The relevant knowledge clauses are:

```
parts_of(dial_telephone,[dial,telephone_rcvr,
  flexible_cord,receiver_cradle]),
index(dial_telephone,telephone_rcvr),
index(dial_telephone,dial)
```

Plus the observations about the touch-tone telephone present in this situation:

```
exist(tt_telephone),
parts_of(tt_telephone,[keyboard,telephone_rcvr,
  flexible_cord,receiver_cradle])
```

The features marked salient with **index** can be computed by examining the **telephone_plan** for parts and features that appear in preconditions of steps. That yields the list:

```
prec(pick_up_rcvr,[exist,dial_telephone]),
prec(pick_up_rcvr,[exist,telephone_rcvr]),
prec(dial,[hear,dial_tone]),
prec(select_digit,[exist,desired_number]),
prec(find_digit_on_dial,[exist,dial]),
prec(hear_clicks,[hear,clicking]),
prec(answer,[hear,ringing])
```

Intersecting this list with the static features of the telephone suggests **telephone_rcvr** and **dial** as the salient features of a telephone. Measured this way, salience of features is relative to the system's current state of knowledge.

5 Elaboration of Situated Plans

During adaptation, SCAVENGER notes each adaptation made and the reason – the situation difference – that prompted it. **Telephone_plan** is then *elaborated* to incorporate SCAVENGER's touch-tone phone experience. Elaboration conditionalizes a plan to recognize a new situation and make the appropriate modifications. Elaboration steps through SCAVENGER's memory of an episode, adding the modifications made (step insertions, deletions, and reorderings, and new bindings) to long-term memory. It inserts a *discrimination point* into the plan for each adaptation, specifying the change to be made and describing the *context* in which it applies. A discrimination point is activated when its context matches the situation of execution.

A context is a set of features characterizing a situation.[1] All the discrimination points created by a given learning episode share a common context. The context for the example situation is [**using(telephone), has_part(keyboard)**]. (We use

[1]The context is currently provided to the system as part of each situation. Preferably, the system should abstract out its own context features. This is an area of current research.

the term *keyboard* rather than *touchpad* to emphasize that the system does not need to distinguish the two; they are both "objects with arrays of buttons.") The latter condition discriminates this situation from the known one having context [**using(telephone), has_part(dial)**].

The clauses added by this elaboration are:[2]

```
step_binding(pick_up_rcvr,
  [isa, telephone, has_part, keyboard],
  [dial_telephone, tt_telephone])
plan_variation(dial,
  [isa, telephone, has_part, keyboard],
  substitute_step, [find_key_on_keypad, for,
  find_digit_on_dial, of, dial, ...])
plan_variation(dial,
  [isa, telephone, has_part, keyboard],
  substitute_step, [dial_one_tt_digit, for,
  dial_one_digit, of, dial, ...])
step_binding(hear_clicks,
  [isa, telephone, has_part, keyboard],
  [clicking, tone])
```

The two **step_bindings** arise from the feature-substitutions discussed in section 4. The first **plan_variation** represents the substitution of a step for finding a desired key on the keypad for that of finding one on a dial; the second represents the substitution of the physical process of pressing a key for that of dialing a digit.

Elaboration also reifies similarity-based associations as *ad-hoc categories*[Barsalou, 1983] making them available as guides for future adaptation. In our example, a new category **dial_telephone_5** is created and marked as subsuming **dial_telephone** and **tt_telephone**. We now have available the notion that dial and touch_tone telephones are intrinsically "similar" (more precisely, *functionally* similar in some set of situations) and we can use this fact in future action and adaptation. Linkages (PURPOSE and ISA*) are set up between the new concept and its specializations to encourage such adaptations to occur. **Dial_telephone_5** approximates the basic-level concept "telephone". Expressed as clauses:

```
isa(dial_telephone,dial_telephone_5),
purpose(dial_telephone,dial_telephone_5),
isa(tt_telephone,dial_telephone_5),
isa_star(tt_telephone,dial_telephone_5)
```

6 Reorganization of Memory

Elaboration makes no concerted effort to integrate discrimination points into memory. As knowledge accumulates, memory needs to be re-organized (cf. Dynamic Memory[Schank, 1982]). In models such as those of Kolodner[1983] and Lebowitz[1987], re-organization is the process that structures knowledge; it is an essential aspect of acquisition. Our re-organization is driven by the desire for improved access to data already structured by experience and the need to identify pertinent details linking situations with actions. It consists of local, syntactic modifications to the plan structure that reduce its complexity, as measured by the number of decisions that need to be made in executing the plan in a given circumstance. Through many such re-organizations, knowledge becomes integrated into memory, and drifts toward routines specialized to specific situations. This is in contrast to models such as that of Murray and Porter[1989], where integration is primarily concerned with maintaining the logical consistency of memory and occurs in a single operation.

The information that drives re-organization is:

- Where (at what steps) run-time decisions occur.
- How many run-time decisions are associated with a given plan and its immediate substeps.
- Which steps are *substantially changed*, meaning that more than 1/3 of their substeps[3] would be modified given the specified context.

For each substantially changed step, we create a new step with the changes "built in," plus a new discrimination point that substitutes the new step for the old one in the relevant context. This operation replaces several decisions with a single one, reducing the number of decisions to be considered at run-time for situations matching the given context.

The **dial** plan experiences a substantial number of modifications in the touch-tone telephone situation: of 6 steps, 3 are affected. Therefore the re-organizer creates a new step **dial_8** and adds a discrimination point to **dial** such that **dial_8** will replace **dial** whenever the current context matches the touch-tone one. **Dial** and **dial_8** share substeps, meaning that later generalizations of these substeps will also be shared. The creation of **dial_8** and conditionalization of **dial** in effect replace **dial** with an abstracted notion of dialing that subsumes both **dial** and **dial_8**. A **steps** clause defines the new step **dial_8**, and a **plan_variation** clause establishes the relationship between the new step and the old step **dial**:

[2]The formats for these clauses are:
step_binding(*stepname*,*context*,[*identifier_bound*,*value*])
plan_variation(*stepname*,*context*,*type*,*description*).

[3]This percentage was chosen heuristically; the particular value is not critical.

```
steps(dial_8,[find_first_digit,select_digit,
  find_key_on_keypad, dial_one_tt_digit,
  hear_clicks, dial_loop_test]
plan_variation(wildcard,
  [isa, telephone, has_part, keyboard],
  substitute_step, [dial_8,for,dial,of,*,
  reason, [split, dial, on_context,
    [isa, telephone, has_part, keyboard]]]])
```

SCAVENGER's plans are hierarchically structured; each node either represents a primitive action or is decomposed into subnodes. Re-organization merges a set of decision points into a single decision closer to the root of a plan, creating a specialized subplan of wider scope. The context associated with those discrimination points, the description of their situations of applicability, propagates upward, becoming more immediately accessible. Re-organization optimizes plan segments for simplicity of execution. Re-organization should be an experience-driven process, in that the effort will be best spent if the most-used plans receive the most attention. We keep track of how often each step that involves a decision is executed, and periodically reorganize those steps which have been run the most times since the last re-organization.

Other forms of re-organization are desirable. Aggregation [Weld, 1986] is one such form of plan reorganization; we assume that the dialing loop within the **dial** procedure was created by such a method. Others include elimination of redundant steps via detection of causal relevance or irrelevance of steps, elimination of redundant tests by merger or subsumption, and factoring of common subplans or step sequences. Other transformations could promote steps of a subplan into the current plan level, or demote steps, depending on which configuration required the least the number of conditionals. We envision a library of plan-transformation operators of which the above are special cases. These would be realized as a series of pattern-directed plan transformations. These transformations differ from those of Collins[1987] in that they are local, syntax-directed optimizations rather than large-scale transformations requiring an understanding of the overall strategy behind a plan.

7 Summary and Conclusions

We have presented a synthesis of learning, planning, and acting that emphasizes practical action over generality of knowledge and plan efficiency. In commonsense domains one's knowledge is inevitably incomplete. Each new situation has the potential to confound one in new and unexpected ways. Expectations are always subject to disconfirmation by experience. How is a planner to cope with this? Our example illustrates several principles:

The interpretation of a situation is always tentative; every action simultaneously applies and tests that interpretation[Heritage, 1984; Suchman, 1987]. SCAVENGER's success in interpreting the touch-tone telephone as a dial phone not only adjusts the plan, it confirms the system's understanding of the situation and its choice of plan. The correspondence between plan and situation also suggests a generalization that can be made across the two concepts.

Every situation of action is an opportunity for learning. Future calling episodes will expand the system's notion of what a phone can look like, where one is likely to be found, and how it can be expected to behave. Learning is generally ignored in deductive planners, because updating the domain model correctly and consistently is simply too hard.

Working from experience simplifies planning both by suggesting actions and constraining the space of possible actions and interpretations[Hammond, forthcoming; Kolodner and Simpson, 1989; Alterman, 1988; Schank, 1982]. SCAVENGER is able to interpret the touch-tone phone as a kind of dial phone because the dial-phone plan provides an interpretation of the situation that defines the relevant features of a (dial) telephone.

SCAVENGER acquires several types of knowledge. It extends the **telephone_plan** to cope with touch-tone phones, collects information that discriminates situations to which each variant is applicable, and acquires a new category *telephone*.

Knowledge that is genuinely new - not deducible from existing knowledge - can only come from experience. Methods that learn by restating existing knowledge, e.g., *operationalization* in EBL and *indexing* in case-based reasoning, contribute to our understanding of knowledge organization, but are excluded from "learning" in a larger sense. The only escape is to predefine all the basic concepts the system will ever need to know. While this has been proposed as a theory of mind[Fodor, 1975], as an implementation technique it simply defers the problem. The key question is: How is actual learning possible? Although it will probably not take the exact form proposed here, we believe that the concept of ad-hoc learning grounded in experience is a step toward such a method.

References

[Agre and Chapman, 1987] Philip E. Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87*, pages 268–272, 1987.

[Agre, 1988] Philip E. Agre. The dynamic structure of everyday life. Technical Report AI-TR 1085, MIT Artificial Intelligence Laboratory, 1988.

[Alterman and Zito-Wolf, forthcoming] Richard Alterman and Roland Zito-Wolf. Planning and understanding: Revisited. In *Proceedings of 1990 AAAI Spring Symposium*, forthcoming.

[Alterman, 1988] Richard Alterman. Adaptive planning. *Cognitive Science Journal*, 12:393–421, 1988.

[Barsalou, 1983] Lawrence W. Barsalou. Ad-hoc categories. *Memory and Cognition*, 11(3):211–227, 1983.

[Chapman, 1987] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.

[Collins, 1987] Gregg C. Collins. Plan creation: Using strategies as blueprints. Technical Report CSD/RR 599, Yale University, 1987.

[DeJong and Mooney, 1986] Gerald DeJong and Raymond Mooney. Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176, 1986.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971.

[Firby, 1987] R. James Firby. An investigation into reactive planning in complex domains. In *Proceedings of AAAI-87*, pages 202–206, 1987.

[Fodor, 1975] Jerry Fodor. *The Language of Thought*. Crowell, 1975.

[Georgeff and Lansky, 1987] Michael P. Georgeff and Amy K. Lansky. Reactive reasoning and planning. In *Proceedings of AAAI-87*, pages 677–682, 1987.

[Hammond, 1986] Kristian J. Hammond. CHEF: A model of case-based planning. In *Proceedings of AAAI-86*, pages 267–271, 1986.

[Hammond, forthcoming] Kristian J. Hammond. Case-based planning: A framework for plannng from experience. *Cognitive Science*, forthcoming.

[Heritage, 1984] John Heritage. *Garfinkel and Ethnomethodology*. Polity Press, Cambridge, England, 1984.

[Kolodner and Simpson, 1989] Janet L. Kolodner and Robert L. Simpson. The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science*, 13:507–549, 1989.

[Kolodner, 1983] Janet L. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:243–280, 1983.

[Lebowitz, 1987] Michael Lebowitz. Experiments with incremental concept formation: Unimem. *Machine Learning*, 2:103–138, 1987.

[Minton *et al.*, 1989] S. Minton, J. G. Carbonell, C. A. Knoblock, D. R. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: A problem-solving perspective. *Artificial Intelligence*, 40:63–118, 1989.

[Mitchell *et al.*, 1986] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[Mooney, 1988] Raymond Mooney. A general explanation-based learning mechanism and its application to narrative understanding. Technical Report AITR 88-66, University of Texas at Austin, 1988.

[Murray and Porter, 1989] Kenneth S. Murray and Bruce W. Porter. Controlling search for the consequences of new information during knowledge integration. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 290–295. Morgan Kaufmann, 1989.

[Pazzani, 1988] Michael J. Pazzani. Learning causal relationships: An integration of empirical and explanation-based learning methods. Technical Report UCLA-AI-88-10, University of California, Los Angeles, 1988.

[Schank, 1982] Roger Schank. *Dynamic Memory*. Cambridge University Press, 1982.

[Suchman, 1987] Lucy A. Suchman. *Plans and Situated Actions*. Cambridge University Press, 1987.

[Tversky, 1977] Amos Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[Weld, 1986] Daniel S. Weld. The use of aggregation in causal simulation. *Artificial Intelligence*, 30(1):1–34, 1986.