Multicases: A Case-Based Representation for Procedural Knowledge*

Roland J. Zito-Wolf and Richard Alterman
Computer Science Department - Center for Complex Systems
Brandeis University, Waltham MA 02254
rjz@cs.brandeis.edu; alterman@cs.brandeis.edu

Abstract

This paper focuses on the representation of procedures in a case-based reasoner. It proposes a new method, the *multicase*, where several examples are merged without generalization into a single structure. The first part of the paper describes multicases as they are being implemented within the FLOABN project (Alterman, Zito-Wolf, and Carpenter 1991) and discusses some properties of multicases, including simplicity of use, ease of transfer between episodes, and better management of case detail. The second part presents a quantitative analysis of storage, indexing and decision costs based on a decision-tree model of procedures. This model shows that multicases have significantly reduced storage and decision costs compared to two other representation schemes.

Introduction

A currently popular reasoning paradigm for AI systems is case-based reasoning (CBR: Rissland and Ashley 1986; Stanfill and Waltz 1986; Alterman 1988, Kolodner 1983). CBR proposes to de-emphasize reasoning from general principles in favor of a more memory-intensive approach. The representation of large numbers of cases is crucial to practical applications of CBR. However, to date case representation alternatives have not been explored in a systematic way.

This paper examines representations for procedures in a case-based reasoner. Two basic organizations have been proposed: *individual cases* and *microcases*. Individual cases (e.g., Kolodner 1983; Lebowitz 1983; Hammond 1990; McCartney 1990) equate the unit of knowledge presentation, the *example*, with the unit of retrieval from memory, the *case*. Under this method, case retrieval returns a single complete example episode for the target task. Microcases (e.g., Stanfill and Waltz 1986; Langley and Allen 1990; Goodman 1991) convert each example into multiple cases, one for each step of the episode. Procedure retrieval occurs incrementally, one retrieval per procedure step. Hybrids of these methods have also been proposed (e.g., Redmond 1990; Robinson and Kolodner 1991).

Although many representations have been proposed, no serious analysis of these alternatives has been attempted. Intuitively, individual cases suffer from redundancy and fragmentation of knowledge, while microcases suffer from increased retrieval effort due to the expanded number of cases.

This paper advocates a new representational method, the *multicase*, in which related examples are merged into a single structure. This paper will take a first step toward quantifying the consequences of different case representation and indexing methods. Multicase organization reduces both storage and retrieval costs, facilitates transfer, and helps manage the accumulation of case detail.

The Multicase

A multicase representation for a procedure is similar to a decision tree, in that it describes sequences of actions and decisions that achieve a given result. A multicase for making photocopies is shown in Figure 1. We define a *decision point* (DP) as any point in a plan requiring selection among alternatives: choices of action, expectations about events, or determination of values for step parameters. Each decision point contains the knowledge relevant to making a single decision, represented in terms of cases. For example, the branch to the **get copy card** step (marked **A** in the figure) would be associated with example episodes in which that action was appropriate. To make a decision at a DP, the features of the current situation are matched against the cases stored with each alternative. (For simplicity, the cases associated with each DP are not shown.)

Contrast this to the individual-case representation, where each (distinct) photocopying episode would be stored in memory as a separate case. To represent all the possibilities captured in Figure 1, one would have to store a case for every possible path through the multicase, or at least a high proportion of them (e.g., McCartney 1990; c.f. Hammond 1990). For individual cases, procedure execution basically consists of a single

*This work was supported in part by the Defense Advanced Research Projects Agency, administered by the U.S. Air Force Office of Scientific Research under contract #F49620-88-C-0058.

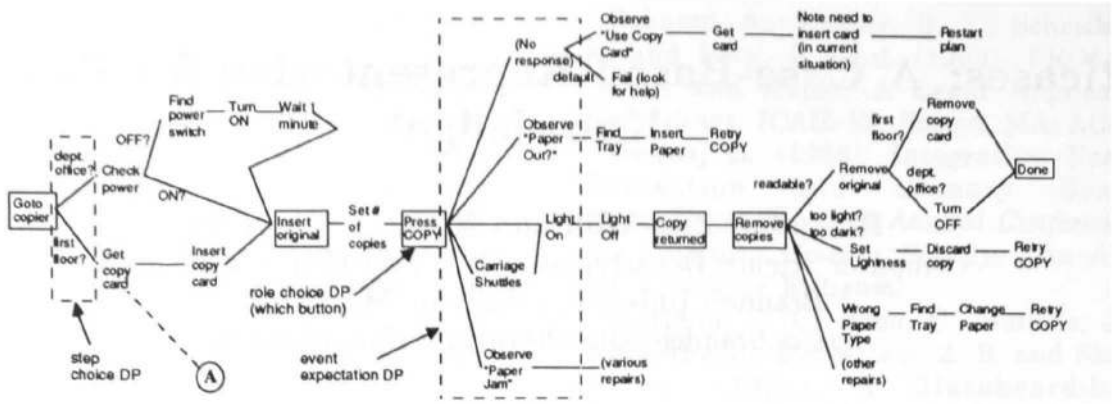

Figure 1: A Multicase for Photocopying

up-front decision among alternative cases (Figure 2a).

One popular variant of the individual-case method includes systems such as CYRUS (Kolodner 1983) and IPP (Lebowitz 1983) in which cases are organized via hierarchies of generalizations (MOPS, Schank 1982) as shown in Figure 3. A major disadvantage of this method is that it entails redundant index hierarchies having large storage requirements (which we will quantify later). Multicases reduce the need for generalization hierarchies by *conditionalizing* parts of the procedure – e.g., its steps and step parameters. A multicase corresponds to a generalized procedure, but **it is not a generalization** in the usual sense, as all of the details of its constituent examples are retained. While procedures containing branches are not new (e.g., Turner 1989, Robinson and Kolodner 1991) the substitution of conditionalization for generalization is novel.

The microcase approach breaks up the example multicase into many subcases, one (at least) for each step. The subcases will contain sufficient information to identify when each step should be executed, so that the structure of the photocopying procedure is encoded implicitly in the applicability conditions for the steps. Using microcases, procedure execution is a sequence of decisions, where each decision must decide among all the known steps (Figure 2b).

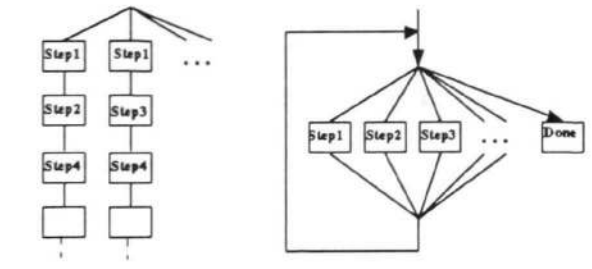

Figure 2: Procedure Representations

A Multicase Implementation

FLOABN (Alterman Zito-Wolf, and Carpenter 1991) is a project exploring the acquisition of plans for using everyday mechanical and electronic devices, such as photocopiers and telephones. FLOABN acquires and revises plans via adaptation and through the interpretation of instructions and messages read or received during interaction with a simulated environment. The core of FLOABN is an adaptive planner SCAVENGER (Zito-Wolf and Alterman 1991) which uses a multicase plan representation. The multicase provides SCAVENGER with background knowledge for operationalizing instructions or messages received during execution.

The multicase is well-suited to this type of application. It is simple to execute as it is in a directly procedural form. Incorporating knowledge from new episodes is simplified because the multicase is stored in a compatible (i.e., episode-like) form. New episodes are first *assimilated* to an appropriate multicase through adaptation (Alterman 1988), after which the multicase is *elaborated* by adding DP's to account for any elements or decisions of the episode not already present in the multicase. An interesting feature of this domain is that over time the multicase becomes customized to the agent's normal situations of action. For photocopying, the plan becomes habituated to the photocopiers the agent normally uses.

The algorithm SCAVENGER uses to act and to learn

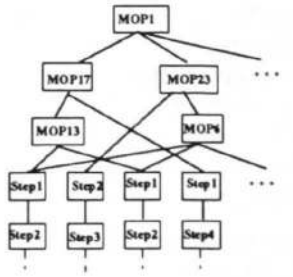

Figure 3: Individual Cases Plus MOPs

```
Given: a multicase-base and a goal
1. Select a multicase appropriate to the goal
2. Execute the next step specified by multicase
   a. if no steps remain to execute, then return(success)
   b. if type(current step(s))=EVENT then wait for one of the events to occur
      if excessive time passes (based on previous experience at this step), adapt or return(event failure).
   c. else   i. if multiple step alternatives exist
                then select one whose context best matches the current situation
             ii. Check preconditions (if missing then adapt, subgoal, or fail)
             iii. Execute step
             iv. Check post-conditions (either adapt, subgoal, or fail)
3. Check for unexpected events: if found then
   a. if adaptation limit exceeded then return(fail: situation too unfamiliar)
   b. if event is receipt of instructions relevant to current plan
      then interpret instructions (Alterman et al. 1991)
4. Whenever adding a step or alternative to the plan:
   a. Add decision point
   b. Find a way to discriminate from other alternatives at that point
   c. Call this the relevant context
```

Figure 4: Pseudo-Code for Multicase-Based Activity with Learning

using a multicase is summarized in Figure 4. The system begins with a skeleton procedure such as an agent might acquire by having the task explained to it or seeing it performed. The boxed steps in Figure 1 indicate this initial copying procedure. Each additional detail arises from some specific experience. Some experiences add new paths (e.g. running out of paper), some add detail to existing paths (e.g. observing lighting and movement as copies are made), and some modify existing steps or decision criteria (e.g., learning where to look for a power switch). Most paths through the multicase reflect contributions from several experiences.

Case Representation Comparisons

Storage Requirements While memory is becoming increasingly plentiful, it remains a finite resource. Individual cases have significant redundancy, as many steps will recur across cases. Worse, as the case-base fills up with variant episodes, retrieval becomes more expensive. Our analysis will show that multicases have the least storage requirements of the methods discussed.

Retrieval Cost The primary execution-time cost in case-base reasoning is in finding and processing the relevant examples. Retrieval cost is influenced by the number of cases searched and the complexity of matching them to the current situation.

Multicases, and to a lesser degree microcases, reduce the cost of each retrieval though partitioning of the procedure and the case-base, as will be shown later. In addition, the cost is distributed throughout procedure execution rather than incurred prior to execution.

These problems can *not* be solved simply though indexing, for two reasons. Indexing may not always be practical, due to, for example, the type of data involved. Second, our analysis points out that indexes involve costs of their own, and that not all indexes are created equal. Indexes require space on the same order as the case-base; hence individual cases have the largest index requirements. Moreover, schemes involving redundant index hierarchies (e.g., MOPs) have indexes much *larger* than the case base itself.

Level of Detail How much detail should be stored when a case is acquired is a perennial CBR dilemma. It is desirable to make case memory as detailed as possible, since the system cannot know in advance which features will prove important and which not. On the other hand, irrelevant details complicate matching, proliferate cases, and increase retrieval cost, so that one desires to store only "relevant" details.

Multicases permit a different approach to the management of detail, by allowing it to be acquired incrementally through the overlay of old episodes with newer ones. The learner can postpone the detail decision: since a multicase is always growing, there is no rush to recall any one case in complete detail.

Transfer When an exact match to a given situation is not available, relevant cases (i.e., partial matches) can usually still be found. Knowledge is in effect distributed over the relevant cases, and the reasoner needs to be able to select and transfer the relevant items of knowledge to the new situation.

Multicase methods facilitate transfer between episodes because a path through the multicase may be formed by plan modifications contributed by several different experiences. For the photocopier multicase, one episode might add the steps for using the ON switch, another those to fill the paper tray when it runs out, and a third the steps needed for doing reductions. Since the relevant information for performing a task is stored together, the need for patching together that information at execution time is largely avoided. Contrast this to individual cases, where transfer of knowledge across cases requires additional mechanisms such as abstraction of specific modifications into repair schema (e.g., Hammond 1990).

Microcases facilitate transfer also, by allowing any known step to be executed at any point in a procedure.

While this may be an advantage in domains that are described by a few underlying rules, such as a Tower-of-Hanoi type problem, it has the disadvantage (at least in domains like commonsense procedures) that each step selection must differentiate between every known step rather than the smaller set of steps known to be relevant in some past example.

Another measure for transfer is the ease of access to the underlying procedure. A multicase is simple to access as it is a single unit. In the other representations, procedures are represented in a distributed form, and must be reconstructed during execution.

Analysis of Storage and Decision Cost

This part of the paper presents a more formal comparison of the multicase, individual cases and microcases focusing on storage and decision cost. The approach is to first formally define a model of a procedure and then show how it would be encoded into cases using each representation method. Two assumptions of this model are that each retrieval returns exactly one case, and that irrelevant features are already "factored out".

Procedure Model

Let the procedure to be acquired be a complete binary decision tree T of uniform depth n. Each node $i \in T$ contains a *step* (or chunk of steps) to be performed plus a *decision* selecting the next node to be executed. T contains $|T| = 2^n - 1$ steps and $2^{n-1} - 1$ decisions (those in the leaves are ignored). Each procedure execution *episode* will consist of n steps and decisions along some path in T from root to leaf.

Let the input to the decision at a node i be the set of binary features F_i. Then $F = \bigcup_{i \in T} F_i$ is the set of all features referenced by the procedure. We assume there exists some upper bound $f = \max_{i \in T} |F_i|$ on the number of features tested by any specific decision, and that f is small compared to $|F|$. We have $n - 1 \leq |F| \leq (2^{n-1} - 1)f$ since there must be at least $n - 1$ features for all the paths in T to be distinguishable, while the upper bound applies in the case that all the F_i are disjoint. A reasonable estimate for $|F|$ should be proportional to f, and should allow for each feature to be referenced in some significant percentage of paths through the tree. We therefore estimate $|F|$ as $(n-1)f$, which models for example a procedure containing $n - 1$ distinct decisions in a fixed order.

Representation of Procedures

Case-based reasoning for procedure execution is the example-based selection of a sequence of steps to achieve a given goal. Each occasion for selection is a *problem* P_i, the process of searching through the case-base to solve a problem is a *retrieval*, and and the number of steps retrieved per problem is the *problem size* S_P. The solution to each problem will be encoded in memory as some set of *cases* C_P. The case is the unit of memory storage and retrieval. Each case pairs a problem solution with a conjunction of features for which it applies. Since it has been stipulated that a given decision references at most f features, at most 2^f cases will be required to represent a decision, one for each possible conjunction of the features and their negations.

We will first consider a linear search model of case retrieval[1], in which the *retrieval effort per problem* E_P is proportional to the number of feature tests made. E_P is the product of the number of cases to be searched through and the number of features to be tested per case. (We assume for simplicity that the cases can be used as retrieved, meaning that we do not attempt to account for adaptation costs.) Letting $|P|$ be the number of problems per episode, the *total retrieval effort* per episode $E = E_P|P|$.

The input parameters of the model are the depth n of the procedure tree T, the maximum number of features f referenced by a decision, and the total number of features $|F|$ referenced by the procedure. $|P|$ and S_P will vary with the specific representation. The outputs of the model are estimates for $|C|$, E_P and E.

Individual cases store each episode of (i.e., path through) T as a case, so $S_P = n$, the number of steps in an episode. Since the entire mapping from situation features to step sequence is performed in one retrieval, $|P| = 1$, with 2^{n-1} potential outcomes. The number of cases can be estimated from the total number of features referenced, yielding $|C| = 2^{(n-1)f}$.

Microcases represent a procedure as a set of independent decisions, making procedure execution a series of case retrievals, one per step. To encode T as microcases we make each selection of a step a separate problem. Then $S_P = 1$, and $|C_P| = 2^f$ cases per problem. There are $|P| = n$ problems per episode, but $2^n - 1$ problems to be encoded to represent the entire procedure, giving $|C| = 2^f(2^n - 1)$. Retrieval effort per problem is $E_P = (|F| + n)2^{n+f}$. Note that n additional features are added to distinguish the 2^{n-1} potential "current positions" within the represented procedure.

Multicases allow us to represent a procedure as a *sequence* of context relative decisions. The retrieval effort is divided up according to separate decisions, and the branching structure of the plan is expressed explicitly as part of the multicase rather than implicitly as extra features referenced by the cases. We have $S_P = 1$, $|P| = n - 1$ problems per episode, $|C_P| = 2^f$ with $2^{n-1} - 1$ problems overall, for a total[2] of $|C| = (2^f + 1)(2^n - 1)$.

Because we focus on only one decision at a time, the number of cases that must be searched through and features needing to be consulted at any given decision point are greatly reduced. For the multicase, only f features need be consulted per decision, and only 2^f cases need

[1] A model for indexed retrieval will introduced shortly.

[2] Since we are counting case nodes, the leaves of the tree add another 2^{n-1} steps.

Item	Individual cases		Microcases		Multicases	
	formula	example[1]	formula	ex.	formula	ex.
Total cases $\|C\|$	2^F	256	$O(2^{n+f})$	124	$O(2^{n+f-1})$	75
CB size[2]	$n2^F$	1280	$O(2^{n+f})$	124	$O(2^{n+f-1})$	75
Effort/problem E_P (unindexed)[3]	$F2^F$	2048	$O(F2^{n+f})$	1612	$f2^f$	8
Effort/problem E_P (indexed)	F	8	$O(n+f)$	7	f	2
Effort/episode E (indexed)	F	8	$O(n^2+F)$	35	F	8
Index size	2^{F+1}	512	$O(2^{n+f+1})$	256	$O(2^{n+f})$	128
Effort/problem E_P ($\|C\|$ fixed)	$\log_2 \|C\|$		$\log_2 \|C\|$		$\max(\log_2 \|C\| - n, f)$	

Notes: 1. Example figures are for a complete binary tree with $n = 5$, $f = 2$, $F = (n-1)f = 8$
2. Total case-base size is product of total decisions and case/decision.
3. Effort/problem is product of features/problem and number of cases per problem C_P.

Table 1: Storage and Retrieval Cost Summary

be examined; the rest of the cases are only relevant to *other decisions*. Thus $E_P = f \cdot 2^f$.

Indexing

Because case-retrieval via linear search involves effort exponential in case-base size, most CBR systems use some form of *indexing*[3] for faster retrieval. An index can be treated as a boolean discrimination network which tests just enough features to discriminate all the cases. Assuming that the network is balanced, the decision cost is proportional to the depth of the index, which is the log base 2 of the number of cases entering into a given decision: $O(nf)$ for individual cases, $O(n+f)$ for microcases, and $O(f)$ for multicases. This simple index model provides a lower bound on access costs; more complex retrieval processes – e.g., inexact matching, choosing among multiple retrieved cases, or features that interact in other than boolean combinations – will have larger decision costs.

The above model of indexing applies to an *optimal* index, that is, one which is (a) balanced, and (b) provides a single path to each case. If the first assumption is violated, the average access cost will be increased. In the second case, storage cost will be significantly increased. For example, a fully redundant generalization hierarchy (such as proposed by MOPs-based systems) involves $F! > (F/3)^F$ nodes[4], a quantity which is much larger than 2^F for any practical value of F.

There are a number of reasons why optimal indexing of the cases may not be practical in all situations: because the features to be indexed must be identifiable in advance for the index to be constructed; because the feature values must be enumerable (for example, features containing variables, features derived through calculations, and continuous features will in general not meet this requirement); and because efficient indexes require significant effort to update as new episodes are acquired. To the extent that indexing falls short, some degree of actual search through cases is required, and the numbers for unindexed decision cost apply.

It might be argued that the above formulas overestimate index costs since no CBR system will ever have available to it more than, say, 2^{15} cases, so that in practice index size is bounded. Let us explore this assumption. First, the relative costs of the indexing schemes are unaffected. Second, for a given $|C|$, individual-case and microcases will have decision effort $E_P = \log_2 |C|$, while multicases will have on the average $E_P = \log_2(|C|/2^n) = \log_2 |C| - n$, a significant improvement. Third, our original estimates for $|C|$ can now be used to estimate the coverage provided per additional example. The larger the space of possible cases, the harder it is to acquire a representative sample of the entire procedure T. Viewed this way, multicases yield the most "knowledge" per example.

Results

The results of this analysis are summarized in Table 1. The formulas derived here are for a complete case-base resulting after all of the possible situation configurations and hence procedure sequences have been observed. Although in general one's case-base is never complete, presumably it must contain a significant percentage of the relevant cases in order to perform adequately, so that the formulas given are expected to be of the correct order of magnitude. The table includes illustrative values for a complete decision tree with $n = 5, f = 2$.

Case-Base Storage. The three alternatives use different amounts of case-base storage. A multicase representation requires the least cases – $O(2^{n+f-1})$ – of the three methods; microcases require 2 times as much, and individual cases require about 2^{nf} times as much.

Effort. Two alternatives were evaluated. If complete indexing is not possible, multicases offer much better

[3]It is important to note that "indexing" is used in the CBR literature in at least three distinct senses: as a *performance* method, to accelerate access to a desired case or cases; as an *organizing* method, for grouping cases observed to have similar features, typically in the service of making generalizations (e.g., CYRUS and IPP); and as a *knowledge-encoding* method, for defining sets of cases with related content. We model the first, we do not model the second, and we assume the third can be encoded as additional features.

[4]Both IPP and CYRUS provide mechanisms for trimming away useless generalizations, but do not evaluate their effectiveness.

per-problem and per-episode retrieval cost than microcases or individual cases ($O(2^f)$ vs. $O(2^{n+f})$ and $O(2^{nf})$ respectively). If complete indexing *is* possible, multicases and individual cases have the same per-episode retrieval cost of $(n-1)f$, but the multicase reduces *per-step* retrieval cost by a factor of $O(n)$. Microcases are the most costly alternative, with $O(n+f)$ retrieval cost per problem and $O(n^2)$ per episode. This is a significant difference when executing a procedure under temporal constraints, where it is desirable to minimize computation per step as well as overall. If all the decision effort is lumped into one large computation, it may become an execution bottleneck.

Index Size. Multicases require less index space than the other two methods, though the difference between multicases and microcases is not large. A more significant advantage is that, if f is small compared to n (e.g, $f \leq 4$) multicases may permit one to avoid indexing entirely, saving not only space but the effort to construct and update them.

Concluding Remarks

A few points deserve drawing out. Representational choices in CBR systems *do have* a significant effect on performance and resource requirements as case-bases increase in size. The multicase embodies two key ideas: exploiting the underlying structure of the problem domain to partition case retrieval into a number of smaller, cheaper retrievals, and keeping the representation as "concrete" (episode-like) as possible (cf. McCartney 1990).

Our formal results depend on our assumption that $f \ll F$ – for procedures, that (a) most choices depend on only a few of the available features, and (b) most steps are only relevant at certain points in a process. Given these, partitioning is very effective. Not all procedures have this property; for example, the Tower-of-Hanoi problem can be described using just one rule which is applied at every step. Issues for future work include the impact of representation on the process of case acquisition, and the complexity and consequences of approximate matching during case retrieval.

The three CBR methods described here can be ordered by increasing constraint on the sequencing of steps. At one extreme, microcases allow any step to be chosen at any time; at the other extreme, individual cases fix entire step sequences. Microcases occupy an intermediate position. Though we have focused on procedure representation, this distinction may be useful in other areas. Consider a case-based design system in the domain of electronic amplifiers. The individual-case approach would correspond to a library of off-the-shelf designs. Microcases would correspond to general design rules for building amplifiers out of smaller functional units. Multicases would correspond to a library of designs plus knowledge of how to adapt them to suit various requirements. In a sense, each multicase would represent the "procedure" for customizing a particular design.

In summary, this paper makes several contributions to the analysis and evaluation of case-based reasoners. It defines an abstract model of CBR to which a variety of architectures can be fit and compared, and defines four criteria on which to evaluate such systems – case-base size, indexed and unindexed retrieval effort, and index size. It provides one of the first detailed complexity analyses of case representation and organization, and uses it to contrast several schemes appearing in the literature.

Acknowledgements

The first author thanks Marc Goodman for sharing his expertise in CBR systems in general and indexing in particular.

References

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

[2] Richard Alterman, Roland Zito-Wolf, and Tamitha Carpenter. Interaction, comprehension, and instruction usage. *Journal of the Learning Sciences*, 1(4):361–398, 1991.

[3] Marc Goodman. A case-based, inductive architecture for natural language processing. In *AAAI Spring Symposium on Machine Learning of Natural Language and Ontology*, 1991.

[4] Kristian J. Hammond. Case-based planning: A framework for planning from experience. *Cognitive Science*, 14:385–443, 1990.

[5] Janet L. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7:281–328, 1983.

[6] Pat Langley and John A. Allen. Learning, memory, and search in planning. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 364–369, Chicago, Illinois, 1991.

[7] Michael Lebowitz. Generalization from natural language text. *Cognitive Science*, 7:1–40, 1983.

[8] Robert McCartney. Reasoning directly from cases in a case-based planner. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 101–108, Hillsdale, NJ, 1990. Lawrence Erlbaum Associates.

[9] Michael Redmond. Distributed cases for case-based reasoning: Facilitating use of multiple cases. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 304–309, 1990.

[10] Edwina Rissland and Kenneth Ashley. Hypotheticals as heuristic device. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1986.

[11] Stephen Robinson and Janet Kolodner. Indexing cases for planning and acting in dynamic environments: Exploiting hierarchical goal structures. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pages 882–886, 1991.

[12] Roger Schank. *Dynamic Memory: A Theory of Reminding and Learning In Computers and People*. Cambridge University Press, Cambridge, 1982.

[13] Craig Stanfill and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1239, 1986.

[14] Roy M. Turner. A schema-based model of adaptive problem solving. Technical Report GIT-ICS-89/42, Georgia Institute of Technology, 1989.

[15] Roland Zito-Wolf and Richard Alterman. Ad-hoc fail-safe plan learning. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, pages 908–914. Lawrence Erlbaum Associates, 1990.