Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases

Michael Redmond*
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
(404) 853-9382
E-mail: redmond@pravda.gatech.edu

Abstract

A case-based reasoner can frequently benefit from using pieces of multiple previous cases in the course of solving a single problem. In our model, case pieces, called *snippets*, are organized around the pursuit of a goal, and there are links between the pieces that preserve the structure of reasoning. The advantages of our representational approach include: 1) The steps taken in a previous case can be followed as long as they are relevant, since the connections between steps are preserved. 2) There is easy access to all parts of previous cases, so they can be directly accessed when appropriate.

Introduction

Case-based Reasoning (CBR) (Kolodner and Simpson 1984) is a method of using previous episodes to suggest solutions to new problems. CBR allows a reasoner to solve problems efficiently when previous similar experiences are available. Problem solving using case-based reasoning usually involves retrieving relevant previous cases, adapting the solution(s) from the previous case(s), if necessary, to solve the new problem, and storing the current episode as a new case to be used in the future.

A case-based reasoner can frequently benefit from using pieces of multiple previous cases in the course of solving a single problem. For example, in protocols taken by Lancaster (Lancaster and Kolodner 1988), mechanics doing a troubleshooting task used pieces of different cases to suggest different hypotheses to consider and tests to perform.

An annotated example from our program CELIA (Cases and Explanations in Learning; an Integrated Approach) (Redmond 1989b), which solves problems in the domain of automobile troubleshooting, illustrates the successful use of multiple cases. In this example, the car is stalling. In forming an initial hypothesis, the reasoner retrieves part of a previous case with the same symptoms that suggests that the idle speed is low and that the mechanic should test whether the engine stalls when it is cold. After carrying out the test the mechanic finds that the engine stalls when warm. Since the results of this test are different than in the previous case, the rest of that case is not useful for further diagnosis.

```
***** Relevant Case Snippet Retrieved ****
   The HYPOTHESIS that Case 101, Snippet Case-Generate-
Hypothesis-316 suggests is:
        (LOW IDLE-SPEED)
**** Continue with Linked Case Snippet ****
   The TEST that Case 101, Snippet Case-Test-Hypothesis-318
suggests is: (TEMPERATURE ENGINE-SYSTEM
                (WHEN (STALLS ENGINE-SYSTEM)) COLD)
   The TEST-RESULT predicted is:
        (TEMPERATURE ENGINE-SYSTEM (COLD))
   Result: (TEMPERATURE ENGINE-SYSTEM (WARM))
***** Abandoning Case 101 *****
```

CELIA recognizes that the first case must be abandoned, and retrieves another case to help it interpret the test result. The new case shares hypotheses and test results with the current situation, rather than just symptoms. It suggests that the problem is a low idle mixture.

```
***** Continuing with Case 105, Snippet Case-Interpret-Test-505
  The RULE-IN that Case 105, Snippet Case-Interpret-Test-505
suggests is: (LOW IDLE-MIXTURE)
  The RULE-OUT that Case 105, Snippet Case-Interpret-Test-505
suggests is: (LOW IDLE-SPEED)
...
*** Continuing with Case 102, Snip. Case-Generate-Hypothesis-289
  The hypothesis that Case 102, Snippet Case-Generate-
Hypothesis-289 suggests is:
        (SMALL (DISTANCE THROTTLE-DASHPOT-STEM
                THROTTLE-LEVER))
  The TEST that Case 102, Snippet Case-Test-Hypothesis-287
suggests is: (DISTANCE THROTTLE-DASHPOT-STEM
                THROTTLE-LEVER)
  The TEST-RESULT predicted is:
     (NOT (SMALL (DISTANCE THROTTLE-DASHPOT-STEM
                THROTTLE-LEVER)))
...
```

It further suggests actions to take to fix the problem (not shown), but after carrying out those actions, the car still stalls. This case, too, is abandoned. Another

*This research has been supported by the Army Research Institute for the Behavioral and Social Sciences under Contract No. MDA-903-86-C-173, and by DARPA contract F49620-88-C-0058 monitored by AFOSR. The author wishes to thank Janet Kolodner for her guidance, Ray Bareiss, and Ashwin Ram for helpful discussions, and Tom Hinrichs, Steve Robinson, Louise Penberthy, and Joel Martin for helpful comments on earlier versions of the paper.

case is retrieved that suggests a new hypothesis: the throttle dashpot is out of place. It suggests checking if the distance between the throttle dashpot and the throttle lever is too short. This hypothesis proves to be correct, the fix it suggests is carried out and the problem is fixed.

The process we are describing is one of using pieces of several cases to solve a problem. We have observed this happening in automobile troubleshooting (Lancaster and Kolodner 1988), medical diagnosis, and meal planning (Hinrichs 1988). We suspect that it is common to any problem solving task that is solved by addressing subgoals individually. It is especially evident when planning and execution are interleaved, where the results of execution are not always as predicted. In order for several cases to be used efficiently in combination with each other, several issues must be addressed.

- How to retrieve a case when some part of it could prove useful.
- How to find and isolate the parts of the previous case that will be useful in the current context.
- How to form generalizations to reflect commonalities in parts of the problem solving experiences.

In this paper we discuss a case representation that enables effective combination of multiple cases by making each part of a case directly accessible, while retaining the links between the parts.

Case Representation

When more than one case is used to solve a problem, frequently only parts of each case will be useful in the synthesis. These parts might be buried within an extended sequence of actions serving many goals. For example, when a reasoner is trying to find a test for a particular hypothesis, the relevant test part of a case is all he needs to focus on. The actions taken and the other hypotheses are not important at that time. Cases must therefore be represented such that their parts can be efficiently accessed.

Traditionally, cases used by case-based reasoners have been treated as monolithic entities [1]. That is, an episode is stored as a single instantiation of a single knowledge structure. Aspects of a case are specified as slots in the representation. Although many of these representations have structured representation within slots and reasoning can be applied to parts of a case, indexing, in general, retrieves the case as a whole.

Treating a situation as a monolithic case and embedding everything in it creates problems for using parts of multiple cases to solve one problem:

- Retrieval of the parts of the case that can be helpful in a given situation has to be a two step process. First, the appropriate case must be accessed, then the currently useful parts hidden inside the case must be found. It can take a lot of effort to find them within the case, even given the right indices to the situation. A one step process allowing direct access to the useful parts of cases would be more efficient.
- Monolithic cases contain too much information for a system to be able to do useful generalization. Since cases have many parts, some of which should be generalized with parts of other cases, but others of which are unique, generalization of commonalities may be delayed.

[1]See Related Work section for some exceptions.

In order for the appropriate part of a case to be accessed when it can be useful, in the middle of problem solving, it is advantageous to divide cases into pieces. In our model,

- Cases are stored in pieces, or *snippets* (Kolodner 1988). This allows the reasoner to use small fragments of cases in its reasoning rather than having to wade through large monolithic cases.
- Each snippet is organized around one particular goal, and contains information pertaining to the pursuit of that goal.
- Each snippet contains the current problem solving context at the time the goal was initiated, including the initial problem description and results of actions taken so far.
- There are links between the snippets that preserve the structure of the diagnosis. Each snippet is linked to the snippet for the goal that suggested it and to the snippets for the goals it suggests.

Content of Snippets

Each snippet can be thought of as a scene of the larger episode. A snippet has three main types of information. First, is the problem solving context at the time of the snippet's occurrence. Second is information related to the goal that the snippet is centered around. Last is information linking the snippet to other related snippets. Figures 1, 2, and 3 together comprise an example of the internal representation of a snippet representing the goal of testing the hypothesis that the carburetor float bowl had too high a fuel level.

Context. Problem solving context includes actions and results of actions taken earlier in the problem solving, as well as features of the problem. *Global context* is the features given for the overall problem situation, *internal context* is the circumstances, state or knowledge established by the actions already taken as part of the problem solving. In the automobile troubleshooting domain, internal context includes tests that have been done and their results; information or hypotheses that have been ruled out or ruled in during problem solving; fixes that have been made during problem solving; and the current hypothesis. The global context includes the chief complaint; other symptoms; how frequently the symptoms occur; how long the problem has been going on; any particular ambient temperature range when failure occurs; any particular weather conditions when failure occurs; the car model; the customer; the mechanics involved; and where and when the problem solving occurred. Global context remains the same across snippets of a case, but internal context changes.

Figure 1 shows an example of the problem solving context part of a snippet.

```
CASE-TEST-HYPOTHESIS-130

CONTEXT

Internal
  Ruled-In    (Lean (Position (Idle-Mixture-Screw)))
  Ruled-Out   (Low (Position (Idle-Speed-Screw)))
              (Lean (Position (Idle-Mixture-Screw)))
              (Incorrect (Position (Throttle-Dashpot)))
  Tests-Done-N-Results
         (Temperature Engine-System
                (When (Stalls Engine-System)) Cold)
            (Hot (Temperature Engine-System))
         (Stalls Engine-System)
            (Stalls Engine-System)
         (Small (Distance Throttle-Dashpot-Stem Throttle-Lever))
            (Not (Small
                (Distance Throttle-Dashpot-Stem Throttle-Lever))
  Fixes-Done   (Increase (Position (Idle-Mixture-Screw)))
  Solution: NIL
  Current-Hypoth   (High (Contains Carburetor-Float-Bowl Fuel))

Global
  Complaint            (Stalls Engine-System)
  Other-Sympt          (Rough (Run Engine-System))
  Frequency            Weekly
  How-Long             2months
  Amb-Temp-@-Fail      Any
  Weath-@-Fail         Rainy
  Car-Type             (1981 Ford Granada)
  Car-Owner            Davis Cable
  Participants         Mark Graves, David Wood
  Location             Mikes-Repair-Shop
  When                 2843569149
```

Figure 1: Example Case Snippet Context.

The problem solving context of a snippet is used for matching during retrieval. As Barletta and Mark (1988) have suggested, both internal and global problem solving context are necessary to maintain coherence and consistency of actions. Since snippets include both internal and global problem solving context, retrieval results in *usefully* similar case pieces.

This form of context creates advantages for combining multiple cases to find a solution. When there is a need to access part of another case, having the internal context available allows matching on results of previous problem solving. Thus a snippet which followed from similar steps and results can be favored. In this way both access issues are addressed: accessing a case that is relevantly similar, and accessing the part of the previous case that will be useful in the current context. An appropriately relevant snippet can be directly accessed. With monolithic cases the internal context at each point in the problem solving would not be available to make accessing parts easy. By saving the context with each piece we are trading space for flexibility. Any method, in order to be as flexible, would have to either represent the internal context at each point, or be able to recompute it at retrieval time, an expensive proposition.

In addition, though not currently implemented, representing the internal context enables analytical reasoning that could determine that the current context is incompatible with something already done prior to the snippet in the previous situation, thus averting failure.

Pursuit of Goal. Each snippet is centered around the pursuit of one goal. It is here that the actions taken in pursuit of a goal and the results of those actions are recorded. When a snippet is retrieved during problem solving, these slots suggest the actions to take and the results to expect if the situation is the same as in the previous case. We have identified 7 types of goals involved in troubleshooting, Table 1 lists those goal types and their associated slots. Figure 2 shows the slots for a test: the test that was done, the method of carrying it out, the tools used, and the result. In general, the goal-related part of a snippet needs to include the actions carried out to achieve the goal and the effects of the actions.

```
CASE-TEST-HYPOTHESIS-130

PURSUIT OF GOAL

Goal     G-TEST-HYPOTHESIS
Test     (High (Contains Carburetor-Float-Bowl Fuel))
Test-Method
         (Turn-Off Engine-System)
         (Remove Carburetor-Air-Horn-Screw)
         (Remove Carburetor-Air-Horn)
         (Ask (Level Fuel Carburetor-Float-Bowl)
                   Scale-On-Carburetor-Float-Bowl)
Test-Tools       Screw-Driver
Test-Result      (High (Level Fuel Carburetor-Float-Bowl))
```

Figure 2: Example Case Snippet Pursuit of Goal.

Goal Type	Slots for Goal Type
Generating a hypothesis	hypothesis generated
Clarifying a complaint	actions taken, information gathered
Verifying a complaint	actions taken
Testing a hypothesis	test that was done, method of carrying it out, tools used, result
Interpreting a test (either of hypotheses or repairs)	hypotheses ruled out, hypotheses suggested or ruled in
Making a fix or replacement	replacement or fix done, method of carrying it out, tools used
Testing a fix or replacement	test that was done, method of carrying it out, tools used, result

Table 1: Goal Types and Their Slots.

Linkage. While it is important to divide cases into snippets so that parts of cases can be easily and directly accessible, it is also important to be able to reconstruct the case. Sometimes a number of steps in the same case can provide useful guidance. A hypothesis suggests a test, the result suggests an interpretation, the interpretation suggests a fix, and the fix is associated with a test of the fix. As long as the expectations from a previous case are upheld in the new situation, the reasoner can benefit by following the sequence of reasoning steps from the recalled case. In order to enable such reconstruction, snippets include links to the snippets for the goals they follow from and the goals that follow from them. Briefly, the idea is, first, retain the links so that in the future use of the snippet, the step that it suggested will be suggested. Second, save the value of the main slot of the preceding snippet to facilitate making generalizations involving the previous step taken. Note that the portion of the snippet shown in Figure 3 has a slot for the previous *hypothesis* because it was preceded by a generate hypothesis snippet, whose main slot was a hypothesis.

```
CASE-TEST-HYPOTHESIS-130
LINKAGE
Link-Down        CASE-INTERPRET-TEST-140
Link-Up          CASE-GENERATE-HYPOTHESIS-125
Prev-Hypothesis  (High (Contains Carburetor-Float-Bowl Fuel))
```

Figure 3: Example Case Snippet Linkage.

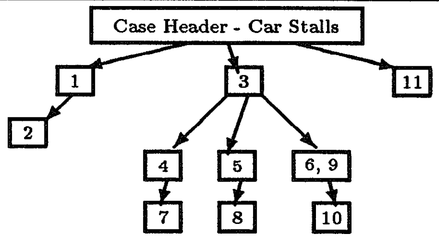
Links Between Snippets

As mentioned above, snippets are linked together in a manner that preserves the underlying structure of the goals pursued in the case. Figure 4 shows the linkages between snippets for a case of a stalling car. This case illustrates the differences that can occur between temporal order and the underlying structure of which snippets follow from which. Each node represents a case snippet for a particular goal pursued in the case. Each link represents a relationship between goals in the case. Arrows point from a snippet to the snippets that it suggests.

Preserving these intra-case links is important for future use of the case. For example, in diagnosis, the case structure preserves which hypothesis suggested a test, and what test result suggested a hypothesis, even if they were not contiguous in the processing. To demonstrate this, in Figure 4, step 6 follows from step 3, since it is a refinement of the previous hypothesis. Step 7 follows from step 4, since it is a test of the hypothesis. Step 11 does not follow from any of the previous hypotheses or tests, it is the start of a new direction after a dead end was reached. This means that the important structure in a case is not the temporal order of the actions taken. Rather, it is which goals follow from which other goals.

For example, in automobile troubleshooting, a hypothesis that the alternator belt is loose could follow from a hypothesis that the battery is not charging since it is a potential root cause. Or a hypothesis that the radiator hose is leaking could follow from a hypothesis that there is a leak in the cooling system, since it is a narrowing of a hypothesis.

Preserving this linkage enables a case to provide guidance as to what to pursue next as long as the results continue to be the same as in the previous case. In troubleshooting, this allows retrieval of a case fragment with a particular hypothesis to yield a connection to the test to do next. The guidance is not affected by any idiosyncratic temporal ordering that does not reflect the underlying structure of the previous case.



Diagnosis Actions (in order presented)

1. Hyp - Loose Connected Spark Plug
2. Test - Loose Connected Spark Plug (Neg.)
3. Hyp - Malfunction Carburetor
4. Hyp - Lean Idle Mixture
5. Hyp - Low Idle Speed
6. Hyp - High Float Level
7. Test - Lean Idle Mixture (Neg.)
8. Test - Low Idle Speed (Neg.)
9. Hyp - High Float Level (restate)
10. Test - High Float Level (Neg.)
11. Hyp - Malfunction Control System

Figure 4: Case Structure.

Snippet Access

Snippets can be accessed two ways:

- Directly, through retrieval, matching the current situation to the snippet's goal and context.
- Sequentially, by following links between snippets.

In CELIA, retrieval via direct access is first restricted to snippets that are centered around the goal type being considered. Then a weighted similarity metric is used, with matching occurring for all features within both the internal and global context. An empirical adjustment of the weight on a feature's importance is made based on the success or failure of prediction during learning (Redmond 1989b). As would be hoped, the similarity metric quickly comes to favor many of the features in the internal context, and give less importance to features in the global context that seem spurious, such as the car owner, the participants, and the location. These, however, are not eliminated, so they can play a part in some unusual situation in which they are important. Further work will investigate using some form of explanation based indexing (Barletta and Mark 1988) in conjunction with this.

Retrieval by sequential access is easy given our case representation. Snippets have links to other snippets that follow from them. When a snippet has been used to suggest an action, if the result is the same as in the snippet's execution, the reasoner can follow the link to the next goal and its actions. Retrieval by sequential access is favored over direct access when it is appropriate. This retains coherence and avoids unnecessary processing.

Advantages of the Approach

Our system, CELIA, learns from observing an expert's actions (Redmond 1989b, 1989a). It uses parts of multiple cases for two tasks

- To predict and explain the instructor's actions as it is learning. The experience is then saved in the distributed form we have described here. Redmond (1989c) explains how this distributed case representation can be constructed from observing raw input.
- To provide guidance during problem solving.

The distributed case representation has advantages for both processes. Here we discuss the advantages for problem solving.

- There is easy access to all parts of previous cases, so they can be directly accessed when a snippet's goal is to be pursued.
- The structure of the case is retained so it can be reconstructed as a whole or in part when necessary. The actions taken in the previous case can continue to provide guidance as long as the situation remains usefully similar to that of the previous case. Operationally, in the current system this means as long as the same results are obtained in the step as in the corresponding step in the previous case.
- Generalizations of case snippets can be formed for the pursuit of a particular goal, but not hindered by the pursuit of other goals in the cases.

While empirical measures of the learning part of the system have been made, our evaluation of the representational approach is based on the fact that it enables, without much cost, flexible problem solving that would otherwise be difficult. With monolithic cases, not only might the reasoner not have the right indices for the case, finding the right part of a case to use for the current task situation is effortful. Even if the case is indexed so that it will be accessed when any of its parts are relevant (*e.g.* by important features of the internal context at each and every point during the case – the tests done and their results, the fixes done ...), once that case is accessed it is necessary to find the point in the case's problem solving where its context best matches the current context. Making this process as simple as CELIA's process would require including all the same information that CELIA's cases have – the context at each point during the case (for indices), directly associated with the step in the case. This case representation content would not be distinguishable from our theory except that our representation preserves the underlying order of the steps.

The ideas discussed raise some issues. First, as snippets become smaller, the distinction between cases and situation/action rules may seem to blur. If problem solving does not benefit from the overall context provided by a whole case, then the case representation might be equivalent to individual decision rules. A related question one might ask is whether problem solving behavior constructed from relatively local decisions can be globally consistent, or will unforeseen interactions between goals creep in. An advantage of traditional case-based reasoning techniques is that a whole case has a coherency that holds together the problem solving behavior. We do not want to lose coherency when we break cases into pieces. We have discussed a number of important ways in which our snippet representation differs from decision rules. First, the links between actions are retained, so a case can be followed as long as the results of actions are as predicted by the previous case. Sequential access of snippets leads to a goal-directed coherence that is not inherent in a set of individual decision rules. Second, snippets include in their context both the initial problem description and the results of actions taken up to that point. This means that when direct access is used the choice of actions to take is directly influenced by the problem solving that has already occurred. Third, the case snippet can provide useful suggestions even when there is only a partial match to the current situation (*i.e.* when a rule may not apply).

Another issue we must address is what size snippet is most useful for problem solving. There is a trade-off between the efficiency of being able to match and use a single case as the solution to the problem, and the generality of matching to the parts that are applicable in the current situation. We have suggested that the appropriate division is for each snippet to concern a single goal. We should clarify that this means that they are concerned with a *leaf* goal. A high level goal such as *explain anomaly* would be broken down into subgoals using knowledge of goals and subgoals. The lowest level goals are the ones that the reasoner would look for guidance in achieving. It is this level of goals that the case snippets are organized around. Applying this to the advantages suggested above, the approach has these advantages:

- Direct access is to the parts of previous cases involved in the pursuit of the types of goals that the problem solver seeks guidance in achieving.
- When generalization of case snippets is added, it can be done for the pursuit of a particular low-level goal for which the reasoner might later want guidance.

It might be argued that the number of indices necessary increases as the cases are divided into smaller chunks and each chunk requires indices. However, if parts of the larger chunks are to be accessible, then equivalent numbers of indices are necessary in order to be able to get an indication that some part of the larger chunk would be of value in the current circumstances. Thus, snippet size and number of indices are independent of each other.

This division of cases into multiple snippets based on goals pursued is important when multiple cases are used to solve different parts of a problem. When changes lead to the need to access a different case to get help pursuing a goal, the part of the case that should be accessed is available such that it can be found in a timely manner. Such division would not be important in a domains in which cases can *only* be considered as a whole (as in *e.g.* HYPO (Ashley & Rissland 1987)).

Related Work and Conclusions

Most CBR approaches have represented cases as single units and reasoned based on one case. MEDIATOR (Simpson 1985) made use of parts of multiple cases in coming to a solution. However, MEDIATOR had to first choose a case, then access the relevant part. More recently, several CBR approaches have separated cases into pieces.

JULIA's (Kolodner 1989; Hinrichs 1988) case pieces represent scenes that are related partonomically and taxonomically. Its snippets, like CELIA's, facilitate synthesizing parts of multiple cases to form a solution, and effective generalization. JULIA does not need the type of links used in our representation, however, because of the relatively low amount of difference in structure of cases, and a relatively static set of goals across problems, with limited need to pursue them in any particular order.

Derivational Analogy (Carbonell 1986) is somewhat similar to our approach in that it saves the problem trace, including generation of subgoal structures and generation of alternatives. A key difference is that our approach provides both direct and sequential access to parts of the problem solving. Derivational Analogy only accesses a trace at the beginning of a problem. If a case can no longer provide guidance a different case must be accessed from the top and the reasoning followed from there. The start of the problem trace is accessed when it shares a subgoal chain with the current situation. Therefore, derivational analogy cannot retrieve a case or part of a case with a similar current subgoal but a different way of getting to it.

Barletta and Mark (1988) group their cases into pieces such that the actions that are used to recover from each of the hypothesized faults are in the same piece. This serves part of the purpose of dividing the cases into pieces, direct access to relevant actions. It appears to be specific to the particular goal of fixing a known fault, however. It is not as flexible as organization by goals, in that it serves a particular goal type, fixing a problem, but does not allow easy direct access to all the goals pursued in the previous case.

Kopeikina, Bandau, and Lemmon (1988a, 1988b) suggested the need for cases that represent how a situation develops over time. Their approach was to divide a case into problem description; action plan; description of effect of implemented action; description of the 'no need for treatment state'; remove controls; and description of a 'no problem state'. Such a case is always accessed at the beginning, and all of the main actions are within the second piece. If the results are not as expected, another case can be used to recover from that *new* problem, however, it is accessed from the beginning as a whole case. They argue against dividing cases up into unconnected individual cases, but do not consider an approach such as ours. Our use of snippets which retain both the internal context, and the internal structure of the case addresses this need.

The use of parts of multiple cases, and the division of cases into linked, goal-centered fragments provides flexibility to recover from changes or unexpected results, while retaining goal-driven processing. The case representation enables direct access to usefully similar parts of previous cases, while retaining the opportunity to follow significant portions of a previous case. When generating a multi-step solution and the solution can be synthesized from multiple cases, our distributed case representation provides significant flexibility advantages. This is important in numerous domains, including automobile troubleshooting, medical diagnosis, many design problems, and we suspect any problem solving task that is solved by addressing subgoals individually.

References

K. Ashley and E. Rissland 1987. Compare and contrast, a test of expertise. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-87)*, San Mateo, CA. Morgan Kaufmann.

R. Barletta and W. Mark 1988. Breaking cases into pieces. In *Proceedings of Case-Based Reasoning Workshop*, St. Paul, MN.

J. Carbonell 1986. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Volume II*. Morgan Kaufmann, Los Altos, CA.

T. R. Hinrichs 1988. Towards an architecture for open world problem solving. In *Proceedings of a Workshop on Case-Based Reasoning*, San Mateo, CA. Morgan Kaufmann.

J. Kolodner and R. Simpson Jr. 1984. A case for case-based reasoning. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

J. Kolodner 1988. Retrieving events from a case memory: a parallel implementation. In *Proceedings of a Workshop on Case-Based Reasoning*, San Mateo, CA. Morgan Kaufmann.

J. Kolodner 1989. Judging which is the "best" case for a case-based reasoner. In *Proceedings of the Second Workshop on Case-Based Reasoning*, San Mateo, CA. Morgan Kaufmann.

L. Kopeikina, R. Bandau, and A. Lemmon 1988a. Case-based reasoning for continuous control. In *Proceedings of a Workshop on Case-Based Reasoning*, San Mateo, CA. Morgan Kaufmann.

L. Kopeikina, R. Bandau, and A. Lemmon 1988b. Extending cases through time. In *Proceedings of Case-Based Reasoning Workshop*, St. Paul, MN.

J. Lancaster and J. Kolodner 1988. Varieties of learning from problem solving experience. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

M. Redmond 1989a. Combining case-based reasoning, explanation-based learning and learning from instruction. In *Proceedings of the Sixth Annual International Workshop on Machine Learning*, San Mateo, CA. Morgan Kaufmann.

M. Redmond 1989b. Combining explanation types for learning by understanding instructional examples. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Hillsdale, NJ. Lawrence Erlbaum Associates.

M. Redmond 1989c. Learning from others' experience: creating cases from examples. In *Proceedings of the Second Workshop on Case-Based Reasoning*, San Mateo, CA. Morgan Kaufmann.

R. L. Jr. Simpson 1985. *A Computer Model of Case-Based Reasoning in Problem Solving*. PhD thesis, Georgia Institute of Technology, Atlanta, GA.