SELECTING THE BEST CASE FOR A CASE-BASED REASONER*

Janet L. Kolodner
School of Information and Computer Science
Georgia Institute of Technology

Abstract

The most important support process a case-based reasoner needs is a memory for cases. Among its functions, the memory for cases must be able to select out the most appropriate cases for the case-based reasoner to use at any time. In this paper, we present the selection processes implemented in PARADYME, a case memory designed to work alongside a case-based reasoner. PARADYME has a two-step retrieval process. In the first step, it retrieves the set of partial matches from the memory. In the second, it selects out a small set of "best" matches. PARADYME chooses "best" cases using a set of six preference heuristics: goal-directed preference, salience, specificity, frequency, recency, and ease of adaptation. PARADYME is novel in two ways. Its use of preferences for choosing a best case means that its principles act as *selectors* rather than restrictors. And its emphasis in choosing best cases is on *usefulness* rather than similarity.

Introduction

A host is planning a meal for a set of people who include, among others, several people who eat no meat or poultry, one of whom is also allergic to milk products, several meat-and-potatoes men, and her friend Anne. Since it is tomato season, she wants to use tomatoes as a major ingredient in the meal. As she is planning the meal, she remembers the following:

> I once served tomato tart (made from mozzerella cheese, tomatoes, dijon mustard, basil, and pepper, all in a pie crust) as the main dish during the summer when I had vegetarians come for dinner. It was delicious and easy to make. But I can't serve that to Elana (the one allergic to milk).
>
> I have adapted recipes for Elana before by substituting tofu products for cheese. I could do that, but I don't know how good the tomato tart will taste that way.

She decides not to serve tomato tart and continues planning. Since it is summer, she decides that grilled fish would be a good main course. But now she remembers something else:

> Last time I tried to serve Anne grilled fish, she wouldn't eat it. I had to put hotdogs on the grill at the last minute.

Considering this, she decides that fish as the main dish would be inappropriate. Having already ruled out meat and poultry as main dishes, she is in a quandry, since it seems that no single main dish will satisfy all the guests. At this point, she comes up with a solution.

> I've had this problem before. ... In that case, what I did was to provide a

*This research was supported in part by NSF under grant No. IST-8608362, and in part by DARPA under contract no. F49620-88-C-0058, monitored by AFOSR. Initial work on this project was begun while the author was on sabbatical at Thinking Machines, Inc., Cambridge, Mass. Thanks to Thinking Machines for providing machine and programming support for the project. Programming was done by Robert Thau.

> choice of main dishes with side dishes that matched all of them. In fact, I usually do that whenever I serve buffet style.

The hypothetical host is employing case-based reasoning (cf., Hammond, 1986, Kolodner, et al., 1985) to plan a meal. In case-based reasoning, a reasoner remembers previous situations similar to the current one and solves a new problem by adapting the solutions to those situations to meet the needs of the new one. In this case, the host is remembering meals she has planned previously to help her generate a plan for her new meal. Some are particular meals (e.g., the tomato-tart meal, the time Anne came for dinner, the last time she served a group of picky eaters), while some are generalized or composite ones, i.e., they've been used over and over with only the variables changed each time (e.g., serving buffet style). The meals she remembers are used to suggest means of solving the new problem (e.g., to suggest a main dish, to suggest serving buffet style, to suggest a means of dealing with all the picky eaters easily), to suggest means of adapting a solution that doesn't quite fit (e.g., substitute a tofu product for cheese), and to warn of possible failures (e.g., Anne won't eat fish).

The most important support process a case-based reasoner needs is a memory for cases. The memory must make cases accessible when retrieval cues are provided to it and it must incorporate new cases into its structures as they are experienced, in the process maintaining accessibility of the items already in the memory. It must be able to handle cases in all of their complexity, and it must be able to manage thousands of cases in its memory. But most importantly, it must be able to select out the most appropriate cases for the case-based reasoner to use at any time.

There are three major ways researchers in the case-based reasoning community are addressing the selection problem. Some people are addressing it by trying to determine how to best choose indexes (e.g., Barletta & Mark, 1988, Hammond, 1986, Kolodner, 1983, Owens, 1988, Schank, 1982) so that only the best cases will be retrieved from the memory. Indexes are used to restrict traversal of memory, and only those cases whose indexed features match the retrieval probe are recalled. One problem with this is that one cannot predict every important feature of an event at the time it happens. Thus, this method is too restrictive. Another problem with this method is that it does not insure that only a small number of cases will be recalled, since many cases might be indexed the same way. While it has worked fine in several implementations, the memories have either been so small that selection was not a problem (as in, e.g., CHEF (Hammond, 1986), MEDIATOR (Simpson, 1985)), or they have had as their goal to recall as much as they could (as in CYRUS (Kolodner, 1983)), where again selection is not a problem.

A second selection method is to filter the problem description before probing memory so that only those features of the problem description relevant to the reasoner's current goal are part of the memory probe (as in, e.g., CHEF (Hammond, 1986)). The problem with this is that some process outside of memory has to choose which features of the problem are the salient ones. It makes more sense to have salience judged in the context of cases already in the memory. Considering the example above, it is a coincidence of circumstances that makes the fact that Anne is a guest an important part of the problem description. That is, the experience already in memory is what tells us that that feature is an important one. We cannot expect an outside process to always know which features or combinations of features are relevant to solving a new problem. It is exactly this task that we want memory to help with.

Other researchers propose filtering methods that are used after retrieval (e.g., Koton, 1988, Riesbeck, 1988, Stanfill, 1987, Rissland & Ashley, 1988). The methods all tend to be special purpose, however, and each has restrictions that keep it from being general. Koton's method, for example, depends on a causal model being available. Rissland's method is specific to adversarial situations. Stanfill's depends on the *memory having large numbers of cases in it in quantities representative of the problem's domain so that an accurate evaluation function can always* be computed based entirely on memory's contents.

And Riesbeck's depends on a static (pre-computed) evaluation function, and thus can't make use of context to decide on a best case.

In the remainder of this paper, we discuss the selection processes used in PARADYME (Kolodner, 1988, Kolodner & Thau, 1988), a case memory designed to be able to select out a small set of best cases from a large case base. PARADYME is designed to work alongside a problem solver. Its cases come from JULIA (Hinrichs, 1988, Kolodner, 1987a,b, Shinn, 1988), a case-based problem solver that plans meals. The problem solver has certain goals to achieve in the context of some problem and a partial solution. The problem, the partial solution, and the goals of the problem solver form the probe to PARADYME's memory (Kolodner, 1988, Kolodner & Thau, 1988).

Upon being probed, PARADYME's first task is to retrieve partial matches from its memory based on the problem description and the partial solution. For the problem described in the introduction, this step would retrieve all meals with vegetarians in attendance, all meals with people allergic to milk products, all meals with meat-and-potatoes men as guests, all meals with Anne as a guest, all meals with tomatoes used as a major ingredient, all summer meals, and all combinations of the above. If the host is someone who entertains or cooks a lot, then clearly, this step will result in retrieval of a lot of cases.

PARADYME's next step, the one we concentrate on in the rest of this paper, is to select out the "best" of those cases. While PARADYME's selection method is based on many of the same principles guiding other case memories, it differs from others in several ways. First, what has been built into previous case memories as restrictors is built into PARADYME as preference heuristics of *selectors*. Thus, PARADYME does not get hurt by the inability to predict every important part of a case at the time it happens. PARADYME prefers cases whose salient features (indexes) match the probe but if no cases with indexed features match, it will recall a case with other matching features. Thus, even if memory update procedures had not indexed the meal where a dish was adapted for Elana by features present in this case, it could still be recalled if no indexed case were found.

PARADYME uses the reasoner's goals similarly. Rather than using reasoning goals to select out a portion of a problem to use as a probe, PARADYME sends the whole problem as a probe and lets previous experience (i.e,. memory) be the guide to which features are the important ones. This way, for example, memory can determine that Anne being a guest is a salient feature at a particular point in the problem solving rather than having the problem solver choose out "guests" as salient features every time an evaluation is done.

Second, PARADYME's emphasis when ranking cases is on usefulness. Using this criterion for ranking means that PARADYME takes the reasoner's goals into account in selecting out a "best" case. Rather than choosing a most similar case, it chooses the most similar of those cases that are first judged most useful. When the hypothetical reasoner above recalls another case where she adapted a recipe with milk for Elana, for example, it is recalled because it predicts how to achieve the goal of adapting a dairy recipe for a non-milk eater. There may be other cases in the memory that are more similar to the situation (perhaps many of the guests match), but this one is most useful for achieving the current goal. Similarly when the case where choice was provided is recalled. There may have been many cases that were more similar than this one, but this one is most useful to the goal of dealing with unsatisfiable food constraints.

Preference Heuristics

PARADYME's selection procedure is based on a set of *preference heuristics*.[1] These heuristics are applied to the set of partially-matching cases to choose a small set of "best" cases. PARADYME uses six different types of preference for this task.

- Goal-Directed Preference

[1] See Rissland & Ashley (1988) for a discussion of why numerical weighting schemes won't work.

- Salient-Feature Preference
- Specificity Preference
- Frequency Preference
- Recency Preference
- Ease-of-Adaptation Preference

The first preference, *goal-directed preference* is based on the principle of utility. That is, since the memory is working in conjunction with a reasoner that has goals, it makes sense to prefer those cases that can help in achieving the problem solver's goals. Thus, when the problem solver is trying to come up with a main dish, those cases that match on main dish constraints will be preferred over others. When it is trying to evaluate the goodness of a solution, those cases that predict success or failure under similar circumstances are preferred. We state this heuristic as follows:

> **Goal-Directed Preference:** Prefer cases that can help address the reasoner's current reasoning goal, and of these, prefer those that share more constraints over those that share fewer.

The second preference heuristic, *salient-feature preference*, is based on the principle that we should use experience to tell us which features of a new situation are the ones to focus on. If memory has done a good job of recording its experiences, they can be used to tell us which features of previous events led to the choice of particular solutions or solution methods and which features of previous events were responsible for success or failure in those cases. These features are the *salient features* of previous cases, and in indexed memories, they form the indexes. When salient features of previous cases exist in a new situation, they can be used to suggest solutions and predict outcomes for the new case. The case where Anne didn't eat fish, for example, has a salient feature set that predicts failure and includes the following facts: Anne was a guest, fish was served, preparation style of the fish was grilled. When all of these features are present in a probe, we can predict that Anne won't eat. PARADYME prefers cases that share full sets of salient features with the new problem over other cases whose full salient feature sets are not in the probe. We state this preference as follows:

> **Salient-Feature Preference:** Prefer cases that match on salient features over those that match on other features, and prefer those that match on a larger subset of salient features over those matching on a smaller subset.

The third preference heuristic is based on the principle that a more specific match can be more predictive than a less specific match. Thus, all other things being equal, cases that match more specifically are preferred over less specific matches. PARADYME has several ways to judge specificity. First, according to PARADYME's definition of specificity, a case is more specific than another if the features that match in the less specific case are a proper subset of the features that match in the more specific case. Thus, a probe is more specifically matched by a case that matches all of its features than one that matches only a subset. Second, a case matches more specifically than one of its ancestors in memory's generalization hierarchy. For example, a particular Italian meal is more specific than a generic Italian meal. Third, a case matches more specifically if the probe matches features in more of its parts. The specificity preference follows:

> **Specificity Preference:** *Prefer cases that match more specifically over less specific matches.*

The fourth and fifth heuristics are based on two principles psychologists have discovered – that items that are referenced more frequently are more likely to be recalled than other similar items and that items that have been referenced more recently are more likely to be recalled than other similar items (all else being equal). This gives rise to two preference heuristics:

> **Frequency Preference:** Prefer cases that have been accessed more

frequently over less frequently-accessed cases.

Recency Preference: Prefer cases that have been accessed more recently over less recently-accessed cases.

A sixth preference heuristic is also based on the principle of utility, and is specific to case-based reasoning. Some adaptations of previous solutions are easier to make than others. This heuristic says to prefer cases whose solutions are easier to adapt than those whose solutions are harder to adapt.

Ease-of-Adaptation Preference: Cases that match on features that are known to be hard to fix should be preferred over those that match on easy-to-fix features.

Application of Preference Heuristics

The application of preference heuristics is complicated. Each preference heuristic attempts to select out a set of better matches. When a heuristic does this, that set is sent on to the next heuristic for pruning. When no subset of cases is better than the rest using some heuristic, however, the entire set it was selecting from is selected. In this way, the preferences act as *selectors* rather than restrictors. We prefer to recall a case that can address the reasoner's current goal but we don't require it. We prefer to recall a case that matches on salient features, but if there are none, the preference heuristics allow recall of a case that matches on a random set of features.

The heuristics are also ordered. Goal-directed preference is applied first, then salience, then specificity, and then frequency and recency. This way, the set of cases that can be used to achieve the reasoner's current goal is selected out first, then any that match on a full set of salient features (of the right kind) are selected from those, the most specific of those are chosen (if some are more specific than others), and then the more frequently or recently recalled cases are selected from those.

There are also other ways the preference heuristics could be applied. For example, some other order might work better. Or, it may be better to run all the preferences on the whole set of partial matches and then to prefer those cases that were selected by more of the preferences. Our current research is focussing on exactly this problem.

Support Processes

While PARADYME chooses best cases by applying its preference heuristics at retrieval time, there are other parts of PARADYME that contribute to making the preference heuristics work. PARADYME has five parts:

1. a hierarchical organization of knowledge and cases
2. a parallel memory retrieval process that chooses out all partially-matching cases from the memory
3. a set of preference heuristics that choose the best matching case from the partial matches activated in step 2
4. a set of transformation rules that transform and elaborate a retrieval probe to get a better "best match" than is possible from the original set of cues
5. a memory update process that marks cases with their salient features and creates generalizations as called for

The hierarchical organization (1) provides a way of determining which partially matching memory structures are more specific than others and gives a way for the retrieval process to determine which partial matches are in the right ballpark. The memory retrieval process (2) chooses the set of cases to focus on in choosing a best match. The transformation rules (4) allow better matches to be found than could be done with only the initial probe. And memory update processes (5) annotate cases with salient feature sets that tell selection processes under what circumstances the case is likely to be relevant. As we stated previously, salient feature sets are similar in function to indexes found in indexed memories. Since they are

so important in allowing the preference heuristics to function, we continue by discussing the types of salient feature sets (indexes) PARADYME assumes its cases will have.

Salient-Feature Sets

We have found three kinds of salient-feature sets (indexes) useful for problem solving. The first contain features that predict the applicability of some method for achieving a goal (*goal-achievement predictor sets*). Second are those that predict the success or failure of a solution (*solution-evaluation predictor sets*). Third are those that describe unusual outcomes (*outcome-achievement descriptor sets*).

Goal-Achievement Predictor Sets are generally conjunctions of goals, constraints on these goals, and problem and environmental features that predict the method or solution for achieving the goal or goal set. If the features of a goal-achievement set are all present in a new situation, and if the problem solver's current goal matches the goal achieved by the salient feature set, then the method of reaching the goal or the solution to the goal can be predicted from the previous case. Cases that match on the basis of goal-achievement predictors are most helpful during problem solving when the problem solver knows what goals it is trying to achieve and knows the environment in which it needs to achieve those goals.

These sets of features may include one or several goals. They include one if the solution that was chosen for that goal did not involve other goals. They include several if solutions to several goals were integrated. Constraints and descriptors on these goals are also included, as are features of the world or features of the problem that determined which of several possible solutions or solution methods was chosen. If all of the features in one of these conjunctive feature sets is designated in a retrieval probe, the solution or solution method used in the previous case can be predicted.

Solution-Evaluation Predictor Sets are conjunctions of features *predicting* unusual outcomes – in general, failures, unexpected successes, and unexpected side effects. If the features of a solution-evaluation prediction set are all present in a new situation, the unexpected result from the previous case can be predicted in the new case. Cases that match on the basis of solution-evaluation sets are most helpful when a reasoner has proposed a solution and needs to evaluate it.

Outcome-Achievement Descriptor Sets are conjunctions of features *describing* unusual outcomes. If the features of an outcome-achievement set are all present in a new situation, the previous case that is recalled can be used to help explain why the unusual outcome arose. In addition, if these features are all present in a new situation and the reasoner is attempting to figure out how to achieve such an outcome, the method by which it was achieved previously can be suggested by the recalled case. These are thus useful in two situations: when the reasoner is trying to explain an anomolous situation and when the reasoner knows the shape of a solution but not how to achieve it.

Any particular case may have several salient feature sets associated with it. For example, it could have one for each goal that was achieved in some unusual way in the course of reasoning about the case. It might also have several associated with outcome and several associated with solution evaluation. When attempting to choose best cases, preference heuristics prefer those cases that have one or more salient feature sets of the right kinds that are fully matched by the new situation. That is, if the reasoner is attempting to evaluate the potential for success of a plan, it prefers cases with fully matching solution-evaluation feature sets. If it is trying to achieve a goal, it prefers cases with fully matching goal-achievement feature sets whose goals match its current goal. If it attempting to explain an anomolous situation, or if it is attempting to find out how to achieve a state of affairs, it prefers cases with fully matching outcome-achievement feature sets.

Discussion

Best cases are chosen in PARADYME by taking into account which features or combinations of features have been found to be most important in the past, and the goodness of fit of a previous case is judged in the context of other possible matches. Preference heuristics select best matches based on what has been relevant in solving previous problems, what the case-based reasoner's current goals are, and the relative specificity of partially-matching cases. This allows the importance of features to be judged in context, where context is provided by the retrieval probe along with the items that are retrieved by partial matching.

While PARADYME's selection method is based on many of the same principles guiding case selection in other case memories, its selection method differs in two ways. First, what has been built into previous case memories as restrictors is built into PARADYME as preference heuristics or *selectors*. Salient feature sets, for example, are equivalent to what others in the case-based reasoning community call *indexes*. That is, they are the features that have been useful previously in making decisions or have been responsible previously for reasoning successes and failures. When salient features from a previous case match features of a new case, they allow predictions or suggestions to be made for the new situation based on the old case. This is the basis of case-based reasoning. Salient feature sets, however, are used differently than indexes have been used. Indexed memories use indexes as restrictors – cases are recalled only when salient features from a previous case match features of the new case. PARADYME's memory, on the other hand, prefers cases that match on salient features over those with no salient features matching, but it also allows recall based on features that were not singled out as salient at memory update time if no cases matching on salient features can be found in memory.

Similarly, PARADYME uses the goals of the reasoner as selectors rather than restrictors. While previous case-based reasoners (e.g., CHEF) used the reasoner's current goal to extract out features from the problem statement to probe memory with, PARADYME sends all the information it has about a situation to memory along with the reasoner's current goals. The preference heuristics use those goals to choose a set of *useful* partially-matching cases. The major advantage to using the reasoner's goal as a selector rather than a restrictor is that it provides a way to let experience (previous cases) designate which features of the new situation are most relevant to consider in achieving a goal.

The second difference between PARADYME and other case memories is in the emphasis on finding a *most useful* set of cases rather than a *most similar* set. Choosing a most similar case means focussing on the correspondences between features of two cases. In aiming to choose a most useful case, on the other hand, we give much attention to what the reasoner needs to do with the case. Correspondences are certainly important (salient-feature preference focusses on these), but they are not sufficient. It only makes sense to focus on the relative goodness of correspondences after we know that the reasoner's goals are being attended to. PARADYME has two preference heuristics that address usefulness: goal-directed preference and ease-of-adaptation preference. Goal-directed preference says that cases that can be used to address the reasoner's current goal should be preferred over others.
Ease-of-adaptation is more specifically aimed at the reasoner's method of achieving its current goal – if it is to use case-based reasoning, and two cases are equally good matches based on other criteria, it should prefer cases that require less work to adapt.

Bibliography

1. Barletta, R. (1988). Explanation-Based Indexing of Cases. *Proceedings of the DARPA Workshop on Case-Based Reasoning.*

2. Hammond, K. J. (1986). *Case-Based Planning: An integrated theory of planning, learning, and memory.* Ph.D. Thesis. Dept. of Computer Science. Yale University.

3. Hinrichs, T. (1988). Towards an architecture for open world problem solving. *Proceedings of the DARPA Workshop on Case-Based Reasoning.*

4. Kolodner, J. L. (1983). Reconstructive Memory: A Computer Model. *Cognitive Science*, vol. 7.

5. Kolodner, J. L. (1987a). Extending problem solver capabilities through case-based inference. *Proceedings of the 1987 International Machine Learning Workshop.*

6. Kolodner, J. L. (1987b). Capitalizing on failure through case-based inference. *Proceedings of the 1987 Conference of the Cognitive Science Society.*

7. Kolodner, J. L. (1988). Retrieving Events from a Case Memory: A Parallel Implementation. *Proceedings of the DARPA Case-Based Reasoning Workshop.* Morgan-Kaufmann, San Mateo, CA.

8. Kolodner, J. L., Simpson, R. L., & Sycara, E. (1985). A Process Model of Case-Based Reasoning in Problem Solving. *Proceedings of IJCAI-85.*

9. Kolodner, J. L. & Thau, R. (1988). *Design and Implementation of a Case Memory.* Technical Report No. GIT-ICS-88/34. School of Information and Compute Science. Georgia Institute of Technology, Atlanta, GA.

10. Koton, P. (1988). Reasoning about evidence in causal explanations. *Proceedings of the DARPA Workshop on Case-Based Reasoning.*

11. Owens, C. (1988). Domain-Independent Prototype Cases for Planning. *Proceedings of the DARPA Workshop on Case-Based Reasoning.*

12. Riesbeck, C. (1988). An Interface for Case-Based Knowledge Acquisition. *Proceedings of the DARPA Workshop on Case-Based Reasoning*

13. Rissland, E. & Ashley, K. (1988). Weighting on weighting. *Proceedings of AAAI-88.*

14. Schank, R. C. (1982) *Dynamic Memory.* Cambridge: Cambridge University Press.

15. Shinn, H. (1988). Abstractional Analogy: A Model of Analogical Reasoning. *Proceedings of the DARPA Workshop on Case-Based Reasoning.*

16. Simpson, R. L. (1985). *A Computer Model of Case-Based Reasoning in Problem Solving.* Ph.D. Thesis. Technical Report No. GIT-ICS/85/18. School of Information and Computer Science. Georgia Inst. of Technology. Atlanta, GA.

17. Stanfill, C. (1987). Memory-Based Reasoning Applied to English Pronunciation. *Proceedings of AAAI-87.*