Research Initiative in Case-Based Reasoning

Edwina L. Rissland

Sept. 1987

COINS Technical Report 88-57

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

Acknowledgements: This document was prepared at the request of Lt. Col. Robert Simpson, DARPA. The author acknowledges the assistance and contributions of the following: Rick Alterman (Brandeis Univ.), Kevin Ashley (Univ. of Mass), Mark Burstein (BB&N), Richard Cullingford (Georgia Tech.), Janet Kolodner (Georgia Tech), Michael Lebowitz (Columbia Univ.), Wendy Lehnert (Univ. of Mass.) and Roger Schank (Yale Univ.). We also wish to acknowledge DARPA Contract N00014-85-K-0017.

Research Initiative in Case-Based Reasoning

Prepared for Lt. Col. Robert L. Simpson,
Program Manager for Machine Intelligence
by

Edwina L. Rissland
Department of Computer & Information Science
University of Massachusetts
Amherst, MA 01003

1. Introduction

The ability to reason with past cases to solve new problems and justify whether or not a past course of action is, or is not, appropriate and hence should, or should not, be taken is central to many problems of central concern to DARPA, such as: provision of powerful reasoning tools to support high level strategic and tactical planning, understanding complex problem solving behavior, aiding the acquisition of expertise, and improving the actual design and manufacture processes for components of military equipment. For instance, a commander faced with a situation requiring development of a course of action needs to relate this new situation with past situations of a similar nature, analyze what was good or bad about the way the past situations were handled, propose ways perhaps modelled on past solutions to handle the new situation, explore the ramifications and uncover potential fatal weaknesses of these proposals, select the best ones, and explain (particularly to those who must carry it out) and justify (particularly to those in higher command) the chosen course of action. Such decision making often occurs in complex domains, under serious time constraints, and with potentially immense penalties for failure. The best decision makers are both thoroughly knowledgeable about the current situation and ever mindful of the lessons of past history; in a word, they are expert case-based reasoners.

Fortunately, many areas of importance to DARPA, such as strategic planning and design and manufacture, are areas where there does exist a corpus of past cases (e.g., the annals of military compaigns, specifications for previously implemented designs) which provides the expert with much of his power; in other words, we have an available case base. Further, in some of these problem domains case-based reasoning techniques provide the expert's primary, if not only, tool since there are few, if any, ironclad theories or rules, and those that do exist are subject to revision, beset with problems of interpretation, and stressed by dynamically changing situations. Thus, we have the inescapable need for case-based techniques. What we do not have, as yet, is an extensive, cohesive, well-developed technological base for using existing case bases to satisfy the need. Through the research contributions of a significant, and growing, community of AI researchers, [1] however, we

[1] E.g., Alterman, Ashley, Hammond, Kolodner, Rissland have all built programs which use case-based reasoning techniques. See the bibliography for a representative sampling of technical papers and the Appendix for brief

do have an excellent foundation for establishing a unified and principled AI approach to case-based reasoning. That is the purpose of this research program.

It is an immediate goal of this research program to develop and explore AI techniques for case-based reasoning, to embrace application domains not doable with other AI methodologies (e.g., expert systems), and to demonstrate these new techniques in a coordinated effort encompassing all of the key ingredients of case-based reasoning. Attainment of these goals requires development of a theory of case-based reasoning, and elucidation of various types and components of CBR, and, most likely, development of generic CBR "shells" to facilitate the building of case-based reasoning systems in various application domains. Another immediate goal is to understand how these newer case-based techniques can be used in problem solving architectures using other AI techniques in a complementary way, for instance, using CBR techniques in concert with learning and knowledge acquisition paradigms to break the "knowledge acquisition bottleneck".

Case-based reasoning ("CBR") can be characterized as the generating, analyzing, or interpreting of courses of actions based on a collection of past and hypothetical cases and the justification and explanation of the conclusions in terms of cases. CBR techniques are used by experts in many domains including the law, mathematics, architectural design, strategic planning, and political policy analysis. For example, Anglo-American common law with its doctrine of *stare decisis*, or reasoning by precedent, is a paradigmatic example of a domain where CBR techniques are used for analysis and interpretation of a new case in terms of old cases and where the only bona fide way of justifying a decision is with cases, where there are no black-and-white rules or predicates, where there are competing answers, where solutions are tested with hypothetical cases, and where the context is deliberately adversarial. Design is a excellent example of a domain where CBR techniques are used for complex problem solving, where new solutions are found through analogical transformations of past ones, where cases are indexed both on success and failure and on both similarities and differences, where problem solutions, and patches to them, are remembered and dynamically indexed, and where there is a crisp performance criterion that the new solution must be implementable and successful.

These two examples, in fact, illustrate the two basic kinds of CBR:

1. **precedent-based CBR** in which past cases, "precedents", are used not only to find a new solution, typically an analysis or interpretation together with its pros, cons and sensitivity to various factors, but also to *justify* it and explain its rationale;

2. **problem-solving CBR** in which past cases are used to find a new solution, typically a plan, detailed problem solution or course of action, but where the new solution is typically offered **without** justifications in terms of the contributing cases.

Both types of CBR share many reasoning subtasks which we outline below. The major differences are: (1) the indispensibility of justification in precedent-based CBR; and (2) the central role of plans in problem-solving CBR. In precedent-based CBR, the relevant precedents, or citations to them,

---

summaries of systems by Rissland & Ashley, Hammond, and Kolodner.

are woven into the solution. In problem-solving CBR, the relevant cases contribute information but are not cited explicitly, even though parts of them might be incorporated "verbatim" into the new solution. Precedent-based CBR typically does not delve into the individual steps of the problem solution, whereas problem-solving CBR does.

Related, but significantly different from CBR, is "**memory-based**" reasoning or **MBR**. MBR also reasons with cases but is fundamentally different from CBR in that the individual cases lose their individual identities and cannot be referrenced or cited in the solution. In MBR it is the contribution of the total ensemble of cases that influences the overall problem solving and evolution of the system. Individual cases, once having contributed to the solution, are no longer available to the reasoner, for instance, for explanation or justification.

In the spectrum of how cases are used in the solution, MBR is at one extreme – individual cases are not available - and precedent-based CBR is at the other extreme – cases must be available and are cited individually. Problem-solving CBR is in between – cases are available but typically not cited. While the thrust of this research program is CBR proper, the related area of MBR should be examined for potential contributions.

Key research issues to be addressed in this DARPA initiative include:

1. Case Memory and Indexing
    - Alternative (e.g., purpose-driven) representations of cases in a Case-Knowledge-Base
    - Abstracting information from cases
    - Management of the Case-Knowledge-Base
    - User-Assisted Acquisition of new cases
    - Automated Acquisition of new cases
    - Indexing Theories & Methodologies
    - Automated Creation of Indices
2. CBR Techniques
    (a) Precedent-Based CBR
        - Hypothetical Reasoning
        - Adversarial Reasoning
        - Argument, Justification & Explanation

    (b) Problem-Solving CBR
        - Problem Understanding
        - Analogical Reasoning
        - Plan Adaptation
3. Interactive CBR Environments

- Generic architecture of a CBR shell
- Intelligent User Interface
- Interactive Explanation & Justification
- Case Summary Generation & Understanding

2. Summaries of Major Research Issues

2.1 Case-Memory & Indexing

Research issues described in this subsection apply to both types of Case-Based Reasoning. Central to both are two issues that go hand in hand: (1) use of indices to retrieve and update case memory; and (2) organization and management of a case memory containing a significant corpus of cases. In order to use previous cases in reasoning, those cases must be made available to the case-based reasoner as needed. This means, first, that a case memory with a significant number of potentially relevant cases, must exist, and second, that the cases be accessible at appropriate times.

With respect to the indexing issue, one way to make sure the right cases are retrieved when needed is to use an analysis of the current case in the search of the case memory. For instance, attributes of the current case can be matched against attributes of cases already in memory and those whose attributes match are retrieved. Key questions concerning indexing are:

1. What features should be used for indexing?
2. How can the feature set used for indexing be updated and changed?

There are several problems that must be solved to make this approach to indexing work:

1. *Attributes to be matched must be chosen well.* One approach is to focus on attributes that are necessary to analyze or resolve the currently open decision. Another is to focus on those that make other domain-related predictions. Yet another is to focus on its purpose or context.
2. *It must be possible to select a best match from the relevant possibilities.* One approach is to use some kind of static similarity metric that is applied through an evaluation function. A better approach is to determine relevancy dynamically and with respect to the current situation and the purpose or point of view of the problem solver.
3. *With experience, there should be learning of which attributes to match on.* One way is to update the set of possible attributes with those that, if present, predict failure or success. A second way is to add those that predict doing something other than what is normally expected. A third is to examine the reasoning in detail and use a technique like explanation-based generalization. A fourth is to use a more statistical or memory-based scheme.

The companion problem to indexing is the problem of organizing case memory. Typically one does not want a flat memory structure (e.g., a list) which as the case base grows will cause access time to grow also. Thus, the challenge is to organize the case knowledge base so that: (1) memory access does not degrade with increased size; and (2) scaling up to a large corpus of cases is possible. Both of these desiderata are constrained, of course, by the requirement that the case base must allow retrieval of relevant cases.

Key issues concerning the organization of a case knowledge base include:

1. What kinds of case structures should be used?
2. How are the case structures linked together?
3. How can different case memory organizations be interfaced?

Current approaches to design of a large case base include:

1. When a case is added to memory, it is indexed within the category corresponding to its main representational concepts by *predictive* features that differentiate it from other cases in the same category. As multiple cases are indexed to the same places, new, more specific generalized descriptions of situations are created, and cases are indexed by these. The net result is a redundant and multiple discrimination net structure in which individual cases and generalized cases reside together. This approach is taken by Kolodner and co-workers.
2. Indexing is not done primarily within categories, as in *1*, but instead it is based on functionality, such as recognition of a potential failure or the use of a novel plan. This approach is taken by Hammond.
3. Indexing is done according to existing "lines" of cases which have addressed a given issue and whose decisions point to similar reasoning (e.g., key features that must be present and resolved) in order to draw a conclusion. This is the approach taken by Ashley and Rissland.

Each approach to case base organization has its pros and cons. For instance, some advantages of the first approach are: general knowledge and individual cases can be found by the same retrieval methods; cases that are similar to each other tend to blend together and the memory does not have to store them all individually; it is unnecessary to store a full representation of each individual case. A disadvantage is that as the case base gets large it tends to become unmanageable because of the large number of remembered differences among cases. Some advantages of the third approach are: it is easy to find cases which resolve a similar issue similarly, which is both the goal and hallmark of precedent-based CBR; individual cases remain individual and are available (in full) for repeated usage; conflicting ways of resolving an issue become apparent; relevancy can be determined dynamically with respect to the current case. A disadvantage is the need to determine lines of cases.

2.1.1 Management of the Case Knowledge Base

Collecting and properly representing a large body of cases will be an important early task in the development of any case-based reasoning system. For this task, adequate tools for developing, editing and browsing through the necessarily large and comprehensive case bases that need to be developed will be vital. If a case-based reasoning system is to be effective, it will also require a background knowledge base of ingredient conceptual terms and ways to generate new ones. [2]

At the minimum, a case management system "shell" must support:

- Effective means to view cases;
- Ability to abstract and summarize cases;
- Ability to peruse case summaries;
- Ability to edit individual cases;
- Access to and control over the types of indices under which cases are stored;
- Ability to create and edit indices;
- Ability to select and control indexing methods;
- Provision for reindexing and global memory reorganizations;
- Consistency maintenance between the representations of related cases;
- Consistency maintenance of use of terms throughout the case base;
- Support for the development of more automatic case acquisition facilities.

2.1.2 Assisted Acquisition of New Cases

In an environment supporting CBR, there is always the problem of acquiring new cases. While the long range goal is to support the automatic acquisition of cases by the environment, in the short haul, the environment must be supportive of user-assisted case acquisition.

User-assisted case acquisition requires:

- an all-encompassing Case-Knowledge-Base manager program, which includes a case editor, browser, etc., as indicated in the previous section;

---

[2] In a related project of the DARPA Strategic Computing Expert Systems program, the Knowledge Acquisition Project of Bolt Beranek and Newman has for the past two years been investigating and developing a collection of state of the art browsing, editing, and consistency maintenance tools for knowledge representation systems. At present, these tools have been demonstrated to work well with frame knowledge bases used for natural language processing and object oriented systems used for modelling physical processes. These tools were designed flexibly, to be used with a variety of knowledge representations, and to support consistency maintenance techniques (akin to truth maintenance) between related types of representations.

- a user-friendly interface, which supports multiple modes of interaction including graphics, menu-driven, and natural language techniques, and which allows the interface to model tasks and purposes for which the environment is used and those of its users;
- tools to support input from existing case bases, libraries, and other sources.

2.1.3 Automated Acquisition of New Cases

When competent performance is dependent on a large knowledge base of specific cases, one must consider the problems involved in creating that case base as seriously as one considers the viablity of case-based reasoning techniques themselves. This places a high priority on the knowledge acquisition aspect of case-based reasoning. In any domain where hundreds or thousands of individual cases must influence the problem solving process, automated knowledge acquisition is, in fact, the only viable strategy.

Thus the long range goal is to have the CBR environment be able to take responsibility for acquiring new cases on its own. Sources of new cases include: (1) results of the CBR environment's own case-based reasoning and problem solving (e.g., system-posed hypotheticals, new problem solutions); (2) new updates from external case bases and libraries; and, of course, (3) the users.

There is a range of problems for automated acquisition. One is the integration of a new case into the case base without any change to the representation or indexing schemes and their ingredient primitives. A more difficult problem is the generation of new primitives and schemas to allow the the structure and description of the case base to evolve.

Integration of a new case into memory is related to the process of finding relevant cases in retrieval since the same indices used to retrieve related cases can be used to store the new case in memory. The new case should, of course, also be indexed by relevant features it has that no other encountered case has ever had. Furthermore, integration of the new case should include making generalizations based on both similarities and differences between the new case and other cases already in memory. Integration should also reflect instances of success and failure of a method or analysis in order to provide shortcuts or warnings in future cases. Problems occuring in integration include the remembering of a good deal of trivial, coincidental correlations and the missing of certain novel ones.

Several approaches are available for case acquisition. For instance, in problem solving CBR, predictions made from a previous solution can form the basis for case acquisition. If a prediction holds up in a new case, the features responsible for the prediction can be used in remembering the new case. If predictions don't hold up, then an explanation of why they didn't could be the way to remember the new case. Thus, one approach to case integration focusses on success and failure. To learn from mistakes and triumphs, the reasoner must keep track of why it made each of the decisions it did and the effects these had on the solution; this is tantamount to dealing with the "credit assignment" problem well-known to workers in machine learning. This suggests that work from machine learning (e.g., explanation-based generalization) will have an impact on the case acquisition problem. Related difficulties in automated case acquisition are: recognizing and explaining a failure of the existing index set and making recommendations for a new index. These

are related to the "new term" and "bias" problems, other classics from machine learning, in which the problem is to learn new terms to add to the problem solver's concept description language to aid the problem solving, is at the heart of the harder case acquisition problem concerning development of new representation and indexing schemes.

At the very least, automated acquisition of a new case requires:

1. Selection of memory structures to be used to store the case;
2. Computation of indices for the case;
3. Positioning the case appropriately in the Case-Knowledge-Base;
4. Accomodation of competing indexing and positioning of the case;

Major difficulties in case acquisition are:

2.1.4 Automated Creation of Indices

Clearly, the effectiveness of any case acquisition tool – indeed the effectiveness of the whole CBR method – depends on the proper indexing of cases and an ability to search memory for cases that are potentially relevant, but whose primary indices did not anticipate relevance in some new situation. Unless all cases are indexed by every possible facet of their description, there will always be the potential for cases to be missed. As new cases are described to the CBR system, either in the course of developing the case base or in active reasoning with a new CBR problem, it is going to be desirable, and, in fact, necessary, to create new indices for old cases automatically, so that the case based reasoner does not have to fall back on extended memory searches repeatedly during normal operations. Automated creation of indices, and hence automatic memory reorganization, is a vital part of any effective CBR environment.

To date many CBR systems have been able to work around this issue for the simple reason that they have never had a large number of cases in memory (i.e., more than 1000). Obviously this will not be acceptable as larger systems and more ambitious CBR decision aids are developed. Many systems will continue to do relatively superficial similarity-based indexing in order to organize cases, but new approaches will also be needed. Candidate methodologies contributing to solution of this problem include: (1) learning techniques like inductive inferencing and explanation-based generalization; (2) adaptive planning; (3) success-driven and failure-driven problem-solving techniques; (4) statistical and memory-based reasoning techniques.

2.2 CBR Methods

2.2.1 Precedent-Based CBR

In precedent-based CBR the key is to extrapolate from cases similar to the current case to decide the current case and to justify this decision in terms of the past cases. A large part of the effort is on selecting and arguing about the relevancy of cases: showing similarity with supporting cases and distinguishing contrary cases. The primary elements of precedent-based CBR are:

1. **statement** of the current fact situation;
2. **analysis** of the current situation;
3. **retrieval** of relevant existing cases from a Case-Knowledge-Base;
4. "**positioning**" of the current situation with respect to retrieved cases;
5. **heuristic (hypothetical) variation** of the current facts and attendant analyses;
6. **selection** of most on point cases, most troublesome cases, etc.;
7. **argument** formulation, experimentation, evaluation and revision;
8. **explanation/justification** of analysis and argument in terms of cases.

Three particularly special aspects of precedent-based CBR are: (1) hypothetical reasoning; (2) adversarial reasoning; and (3) argument, justification and explanation.

2.2.1a Hypothetical Reasoning

One key aspect of case-based reasoning is the ability to generate hypotheticals and use them to test the merits of an analysis. A reasoner must always be alert to the possibility of "signing up" for overly-rosy analyses. Well-designed and artful *what if's* – i.e., hypotheticals – are a powerful way to preclude such decisional blunders. Hypotheticals help the reasoner to fathom the ramifications, soft spots, and side effects of an analysis or proposed course of action. They also can provide cases to supplement a case base sparse in relevant cases; this is especially important when a reasoner must deal with a novel situation.

One approach to using hypotheticals ("hypos") is to generate a "constellation" of them that are conceptually close to the current case and then use these to help assess robustness or vulnerability of proposed analyses. For instance, if a potential analysis or solution, which had looked good in the current case, looks bad in many of the "nearby" hypos, this suggests that that approach is vulnerable. Other approaches are (1) to examine multiple "ply" arguments about a case or (2) to sort and score supporting and contrary most-on-point cases. The first approach is typically a heuristic method, for instance, using one or more of the following: (1) Making a case extreme; (2) Making a case weaker/stronger; (3) Enabling a near-miss analysis; (4) Dis-abling a near-win analysis; (5) Strengthening a near-win; and (6) Adding a closely-coupled aspect; (7) Adding potentially conflicting aspects.

Work on hypothetical aspects of CBR has the potential for significant cross-fertilization with research on machine learning, in particular, on the intelligent selection of training instances. The same sort of heuristics that indicate what sort of hypothetical to consider can be used to indicate what sort of training examples to use. Up to now, very little empirical work has be done to understand the use of even the best known rules of thumb, like "Use near misses" or "Use extreme cases." Fruitful results on the problem of *intelligent example selection* could help alleviate the knowledge

acquisition bottleneck that plagues expert systems as well as any CBR system.

2.2.1b Adversarial Reasoning

Precedent-based CBR is inherently adversarial since there are typically competing, contradictory ways of looking at a case. The facts that a prior case (i.e., a "precedent") had a certain cluster of features, and that the decision of the prior case was made because of some of those features and in spite of others, are treated as a *precedential justification* in an argument that a new case with a similar combination of features should be decided in the same way as the precedent. The decision of the precedent: (1) *Selects* certain features that are important enough for purposes of credit assignment; (2) *Clusters* the selected features; and (3) *Ranks* them, ranking the features in the cluster that favor the decision higher than those that cut against it, at least in that case.

Precedent-based CBR seeks to find those precedents in the CKB which are as close as possible to the same combination of features as the new case (i.e., the most on point cases or **mopc's**) and uses them to construct the skeleton of an argument about how to decide the new case. In analyzing the new case, the importance of features is thus not determined by some *a priori* ranking but *dynamically* in light of the particular combination of features that appear in the new case.

There may be, and often is, no one right answer in Precedent-Based CBR. If the emerging precedents all favor one decision, then the decision is clear. If the precedents lead to conflicting conclusions, however, then the case-based reasoner critically compares them, for example, by distinguishing them, that is, finding factual differences between them and the new case that justify treating them differently, or by determining hypothetically the effect that deciding the new case according to one precedent would have on other well-entrenched precedents. This adversarial process of comparing precedents yields the strongest precedent-based justifications for deciding the new case.

2.2.1c Argument, Justification & Explanation

In generating a skeletal argument of justifications based on precedents, a case-based reasoner provides a framework for explaining a decision and its alternatives. In general, comparisons to other examples make good explanations. A case-based reasoner explains its analyses by citing the precedent cases as examples. By "replaying" the process of comparing and distinguishing the cases in the form of an explicit argument with points, responses and rebuttals, the reasoner lays out the comparative strengths and weaknesses of the alternatives. The case-based reasoner also poses hypothetical variations of the new case or precedents to demonstrate critical features, which if different, would lead to different conclusions. A case-based reasoner's ability to illustrate the consequences of and alternatives to a given course of action by posing hypothetical scenarios (worst, best, most recent, most likely cases, etc.) is vital for planning of an argument.

2.2.2 Problem-Solving CBR

In certain problem solving domains, reasoning on the basis of a previous case can help a problem solver avoid previously-made mistakes (even if a full explanation of why the previous failure hap-

pened cannot be derived) and direct the problem solver in appropriate problem solving directions, such as constraining the search space, aiding the problem solver in dealing with relatively novel situations, or in deriving problem solving shortcuts.

Problem-solving CBR follows the basic cycle: (1) Recall a previous case by probing case memory; (2) Focus on appropriate parts of that case; and (3) Use parts of the previous case to derive an appropriate solution for the new case.

This results in two separate processes running in parallel. A memory process watches the problem solver's focus, uses it to derive search keys, and queries the memory with those search keys. At the same time, the problem solver depends on the memory to return to it whatever it finds that is relevant to the problem solver's focus. The memory process might return generalized knowledge for the solver to use or it might return a previous case which is similar to what the problem solver is currently dealing with. The problem solver might also request necessary information from the memory process when there is something specific it is looking for that is not being provided automatically.

In problem-solving CBR an entire case is rarely used since all of its details is too cumbersome to work with. Rather, the parts of the case that have relevance to the new case are the ones to focus on, where relevance is determined by the reasoner's goals. These include the set of things the reasoning is trying to make conclusions about. Given the current goals of the reasoner, focus is then directed to those parts of the previous case that are relevant to fulfilling the goals. The goals in turn might come from the problem-solving activities themselves, as in means-ends analysis problem solving. Sometimes the goal set is known a priori and the reasoner goes through the set of goals sequentially.

Once the problem solver has an old case, a goal, and has focused on a part of the old case that is to be used in achieving that goal, it attempts to achieve the goal for the new case based on the old one. The process for doing this involves many considerations: Was the previous case a success or a failure? Was the part we are focussing on responsible for the failure or not? Did it change as a result of re-evaluation? Is there a value that, when derived, will achieve the goal, and if so, is that value available in the old case? do we know how that value was derived for the old case? Was it by an "easy" or a "complex" set of reasoning steps? Do we know why the value from the previous case was appropriate? Do we know why the method of deriving that value previously was appropriate? If achievement of the goal is not done by simple derivation of a value, do we have a generalized schema that explains how the goal was achieved previously? If no schema, do we have the set of steps? Is our goal to derive a plan or is it to derive a feature value?

Three particularly critical aspects of problem-solving CBR are: (1) problem understanding; (2) analogy; and (3) plan/solution adaptation.

2.2.2a Problem Understanding

If CBR problem solvers are to use previous experience in making decisions, then they must also be able to evaluate results of reasoning they have previously done. Since one cannot assume that problems for CBR are so well-defined as to admit a black-and-white specifications for success or failure, CBR problem solvers must be able to accept feedback from their users as well as provide

explanations and justifications. In particular, CBR problem-solvers will be expected to become more proficient and this necessitates analysis of both failures and successes. Thus, problem-solving CBR is closely connected with learning and explanation/justification. Because a reasoning failure may be due to misrepresentation or misunderstanding of a problem, problem understanding is a critical part of the standard problem solving cycle, where by understanding is meant derivation of appropriate problem representations and elaboration of the problem statement to fill in missing details.

Understanding a new case includes finding the best knowledge in memory that can be used to make predictions from it. Finding this knowledge is equivalent to integrating the new case with what is already in the memory. As reasoning is going on, memory is constantly being probed and updated, and the case is becoming better integrated and understood, and hence, better knowledge to use in making predictions about the case is being derived.

2.2.2b Analogy

An important component of case-based reasoning is the process of analogizing relevant related cases to suggest solutions and hypotheses for new situations. This potentially requires several forms of analogical reasoning, such as: (1) within-domain adjustments or transformations; (2) across-domain transformations; and (3) use of the same case for different purposes. Such reasoning requires methods for mapping and merging alternative structures and the capacity to integrate several partially-correct analogies.

In general, analogical reasoning requires:

- The ability to retrieve and map multiple related causal scenarios for the current case, possibly from different base domains;
- The ability to determine analogical correspondences based both on the *roles* of objects in comparable situations and also their similarity in terms of static properties;
- The ability to *incrementally* extend an analogy to new target situations, taking into account adjustments made by the target domain reasoner following an initial mapping;
- The ability to *merge* the results of several analogies in reaching a satisfactory solution to the problem at hand.

There are several different analogical methods available. *Transformational analogy* is a method whereby a value or frame (schema) is transferred from a previous case and transformed to fit the new case. In *derivational analogy*, the conditions under which a previous decision was made are taken into account and transfer tends to be of a method for making a decision rather than a value. In *schema-based analogy* the current and previous cases are compared and a schema describing the similarities of the problem statements is described. The schema must be such that it can be used to describe both problem statements. It is then broadened to describe the solution to the previous problem and the new problem is solved by application of the schema.

While these three methods are the ones that are applicable when the previous case resulted in success, additional reasoning must go on when the previous case resulted in failure. In this case, the conditions under which previous values were computed and the set of steps used to to make decisions are checked against the new case to see if the same potential for failure exists. The previous case may also provide suggestions to the problem solver of how to proceed. In order for failure-driven and derivational analogy, in general, to work, the problem solver must keep track of the justifications for its decisions, including the reasoning steps used, the conditions under which they were chosen, and the set of other possible choices, must be maintained, as well as the dependencies between problem solving decisions, in a similar way to what a truth maintenance system does.

2.2.2c Plan Adaptation

When the problem solver's goal is to derive a plan, there are a set of special-purpose case-based inference methods that are used in addition to the methods just described. The problem in this case is "plan modification", modifying a previously-used plan to fit a new situation. The previously-used plan might be a plan schema that is not completely applicable to the new situation, or it could be the particular instantiation of a plan used in a particular case.

One way for a problem solver to modify a plan is to store knowledge with the plan's preconditions about the results of proceeding without fulfilling the precondition, alternate ways the precondition might be fulfilled, and ways in which to change the plan so that the precondition is no longer a problem.

Adaptive planning focuses on techniques needed for re-using old plans in new situations. Because the old plans represent the habitualized activities of the planner they tend to be fairly specific. Adaptive planning requires:

- The explication of background knowledge that is associated with the old plans, in particular, (a) causal knowledge and (b) categorization knowledge.
- Developed heuristics that exploit the background knowledge in the service of re-using the old plan.
- Understanding types of situation difference that can occur between the old plan and the current situation and heuristics that are geared to each type of situation difference.
- Situation matching techniques for replacing individual (sub)steps by others, in memory, that are more appropriate to the current situation.

Work on adaptive planning is directly relevant to work on adversarial methods, when adaptive planning techniques are extended: (1) to handle the interactions among several plans and thus could coordinate plans among allies and adjustments due to the recognition of an adversary's plan; (2) to the problem of recognizing an adversary's plan; and (3) to handle the interaction between strategic and tactical concerns.

Work on adaptive planning also impacts on indexing because a number of the adaptive techniques exploit the categorization of plans. It also relates to interactive justification and explanation because the trace of the adaptive planner can be read like an explanation and justification of its reasoning.

2.2.3 Related Work on Memory-Based Reasoning & Machine Learning

It is one thing to propose theories and solutions regarding memory and indexing that work for a limited prototype operating with a relatively small case base (less than 100 cases). It is quite another to develop techniques that remain effective when the case base is large (thousands of cases). We do not have enough experience with traditional, typically heuristic, AI methods under these circumstances to know whether or not they can maintain acceptable performance levels under large memory conditions. Memory-based reasoning methods, on the other hand, address this situation explicitly since MBR is designed with large case bases in mind. It is therefore imperative to understand the relationship between MBR and CBR.

Two particularly relevant areas are: (1) case-driven training; and (2) generalization-based learning methods.

2.2.3a Case-Driven Training

For any domain where a large number of specific cases are available to a case-based reasoner, there is an opportunity to use the case-base to train the system. This training process may be straightforward or difficult depending on the form of available cases and the types of knowledge representation required. Some problem areas are by their very nature amenable to memory-based techniques, as has been shown by recent work on word pronounciation; this is so because knowledge structures needed to represent the elements of the case are easy to extract. On the other hand, the amount of effort required to train a system in an area such as political history is substantial since all available cases are encoded in natural language descriptions; for such domains, all of the problems associated with knowledge-based natural language processing are relevant since we cannot automate the analysis of a case without automating the process of understanding the text describing it.

If we can make progress on design principles for case-based training, we will have a powerful strategy for handling the "knowledge acquisition bottleneck" problem. With a competent training module in place, the problem of building up knowledge to support a case-based reasoner becomes a problem in designing effective training sessions which entails the intelligent selection or generation of example cases, mentioned above in regard to hypothetical reasoning.

2.2.3b Generalization-Based Learning Methods

In some domains, learning depends on a careful, detailed examination of a few example learning episodes rather than on a more superficial alteration of a learning system's behavior on the basis of how the examples were labelled (e.g., as positive or negative examples). In problem-solving CBR, the system inspects its own problem-solving behavior. Feedback to the system on its performance

is then used to shape the system's evolution as it gains more experience with specific problems. This makes certain problem-solving methods similar to certain machine learning techniques like explanation-based generalization.

Generalization-based methods provide a kind of indexing that is important when handling large numbers of case examples. While other methods of indexing (such as around planning failures) will be needed, it is impractical to expect such complex methods to be applied to large numbers of cases in the near future. It seems reasonable to strike a balance between superficial methods and "deep" problem-solving methods. Generalization-based methods inspect an intermediate level of detail and thus should be useful to work on CBR. Such approaches should complement CBR techniques, which examine small numbers of case in great detail.

2.3 Interactive CBR Environments

There are several aspects of building a high-powered CBR environment. This section briefly addresses some of those that have not been expressly covered in other sections of this document.

Generic Components of CBR Environments

First of all, there is a need to determine the generic subtasks in CBR and a processing architecture that embraces them in a unified way, so that it will be possible to build future CBR systems more efficiently and principledly. A major component of such a generic CBR environment, of course, would be a facility or "toolkit" for managing the Case-Knowledge-Base; desiderata for such a management shell have been addressed in Section 2.1.1. Other generic components of CBR environments are: (1) an intelligent interface that supports graceful interaction in the environment, in particular (2) interactive explanation and justification of the case-based reasoner's conclusions, recommendations, etc.; and (3) case summary generation and understanding.

Intelligent User Interface

In designing and building a sophisticated CBR environment, it is imperative that the needs, styles, and expectations of the users be not forgotten. At the very least, precepts of good intelligent user interfaces should be followed: that the system degrades gracefully, is tolerant of user-error, does not pedantically ask for information that is readily inferable, offers a variety of interaction modalities, and is able to model the users and their tasks. Further work will be needed to elucidate principles and requirements peculiar for CBR environments. These of course will be partially dictated by the application area[s] chosen.

Interactive Explanation & Justification

A sophisticated CBR environment will also need to address the problem of interactively explaning and justifying a case analysis. This requires that the interface not just be a conduit for information between the user and the CBR modules performing explanation and justification. There will need to be models of the user, what information he might want or be satisfied with, and

the purposes to which the information will be applied. It will require intelligent discourse between the environment and user and, as is well-known, this will require conceptual understanding of the user's input.

Case Summary Generation & Understanding

Users of a sophisticated CBR environment, particularly high level decision makers, will demand to see summarizations and abstractions of the case at hand and those cases and hypotheticals that the CBR system used in its reasoning. Further, such a user might well want to present a case to the system in a natural language summarization. Thus, while the focus of this research is case-based reasoning and not natural language, it is important for the success of this research with its intended users that CBR systems have some rudimentary natural language understanding ability. (This is also related to the desiderata of case abstraction, listed in Section 2.1.1, and the central task of problem understanding, considered in Section 2.2.2a.) At the very least, the ability to present and understand short case summaries will help the researchers and developers of CBR systems in their work.

Appendix 1 – Examples of Existing CBR Systems

Example 1 - CHEF, a Problem-solving CBR System in Planning

CHEF is a problem-solving type of CBR system that creates and debugs recipe plans from its memory of old ones in the domain of Szechuan cooking. It is the research project of Kris Hammond of the University of Chicago. CHEF's input is a set of gastronomical goals and its output is a single recipe (i.e., plan) for satisfying them. CHEF accomplishes this by finding a past plan that satisfies as many of the most important goals as possible and then modifying it to satisfy the remaining goals.

Before searching for a plan to modify, CHEF examines the goals in its input and predicts any failures that might arise out of the interactions between the plans satisfying them. If a failure is predicted, CHEF adds a goal to avoid the failure to its goal list. It then makes use of a CHEF-generated explanantion of the failures to index abstract repair strategies appropriate to the problem at hand. In so doing, CHEF is able to avoid failures it has encountered before.

CHEF consists of six modules:

ANTICIPATOR predicts planning problems on the basis of failures caused by past interaction of goals similar to those in the current problem;

RETRIEVER searches CHEF's plan memory for a plan that satisfies as many of the current goals as possible while avoiding problems predicted by the ANTICIPATOR;

MODIFIER alters the plan found by the RETRIEVER to achieve goals from the input that it does not satistfy;

REPAIRER is called if a plan fails. It builds up a causal explanation of the failure, repairs the plan and the hands it to the STORER for indexing;

ASSIGNER uses the causal explanation built by the REPAIRER to determine the features which will predict this failure in the future;

STORER places new plans in memory, indexed by the goals that they satisfy and the problems they avoid.

References

Hammond, K. 1986. "CHEF: A Model of cased-based planning". *Proceedings AAAI-86*, pp. 267-271.

Hammond, K. 1986. *Case-based Planning: An integrated theory of planning, learning and memory.* Ph. D. Thesis, Yale University.

HYPO, a Precedent-Based CBR system in the Law

HYPO is a precedent-based CBR system which analyzes a new case, interprets it in light of legal issues, and creates a skeletal argument and a justification of it in terms of existing cases and hyotheticals. HYPO, developed at the University of Massachusetts/Amherst, operates in the domain of trade secret law. Based upon its knowledge of case law and accepted ways of approaching a trade secret case (called "dimensions"), HYPO is able not only to retrieve relevant cases from its Case-Knowledge-Base but also to dynamically assess their relevancy to the case at hand and therefore select most-on-point, most troublesome, etc. cases (organized in a "claim lattice") to use as the foundations of an argument and an explanation of its rationale.

HYPO consists of the following modules:

CASE-ANALYZER runs through the library of dimensions and produces a case-analysis which includes a list of applicable and near-miss dimensions and cases;

FACT-GATHERER which on the basis of the case-analysis can request additional information from the user about the current fact situation;

CASE-POSITIONER which on the basis of the facts and corresponding case-analysis organizes the retrieved cases in a "claim lattice", which structures the cases according to how relevant they are;

BEST-CASE-SELECTOR selects from the claim lattice special cases such as "most-on-point" cases for each of the sides;

HYPO-GENERATOR uses the case-analysis and the claim lattice to heuristically generate legally meaningful hypotheticals (e.g., borderline or extreme cases, ones that present two opposing ways of looking at the case) to pose to the user for consideration as testing "what if" situations;

3-PLY-ARGUMENT-GENERATOR which on the basis of available best cases for each of the sides, generates an argument for side 1, a rebuttal by side 2, and a response by side 1;

JUSTIFICATION module fleshes out an argument skeleton complete with case citations as required by the legal profession to support, compare, refute, etc. various points of argument;

EXPLANATION module presents the rationale of the argument in terms of cited cases and illustrative hypotheticals.

References

Ashley, K. D., & Rissland, E. L. 1987. "Compare and Contrast, A Test of Expertise". To appear *Proceedings AAAI-87*, Seattle.

Rissland, E. L. & Ashley, K. D. 1987. "A Case-Based System for Trade Secrets Law". *Proceedings First International Conference on AI & Law*, Boston.

Rissland, E. L. & Ashley, K. D. 1986. "Hypothetical as Heuristic Device". *AAAI-86.*

PERSUADER, a Problem-solving CBR system in Labor Mediation

PERSUADER is a problem-solving type of CBR system which works in the domain of labor mediation. It is one of the most recent CBR systems to come out of Kolodner's lab at Georgia Tech. Given a labor dispute, PERSUADER derives a compromise solution, presents it to the disputants, receives their comments, and then either attempts to persuade the party that doesn't agree to the compromise to accept it or derives a new solution based on the old one and feedback from the disputants. PERSUADER does this by finding an important labor/mamagement contract from its memory to use as a "ballpark" solution and then refining it to solve the current problem by considering differences between the current situation and the retrieved case.

PERSUADER uses domain knowledge about the changes implied by different features of a dispute (e.g., if the company in a retrieved situation was very wealthy with respect to the rest of the industry and the company in the new situation is a poorer one, PERSUADER might modify the previous contract with respect to economic demands so that the current company can handle the contract without going bankrupt.) PERSUADER refines the ballpark solutions in a case-based manner, for instance, by using a difference to index a previous reasoning experience and examining the way the difference had been handled in the previous case to come up with a refinement in the current one.

PERSUADER also uses case-based reasoning to find "creative" solutions to problems when a typical remembered solution will not adequately resolve the situation. To do this, PERSUADER uses knowledge structures called SAPs (Situation Assessment Packets) that allow it to recognize the kinds of problem situations that arise under various kinds of goal-conflict situations. SAPs represent strategic information about dealing with such situations and also index cases in which particular solutions were used. These sitations may have nothing to do with labor/management relations, yet because their overall structure in terms of goals and relationships of people is the same, the solutions used in them can be used to solve labor/management problems.

Lastly, PERSUADER takes a case-based approach to argumentation. After proposing a compromise solution, PERSUADER attempts to persuade a party that is not in agreement with the solution to accept it. It uses previous cases in two ways: (1) as a basis for arguments (e.g., you think x will happen as a result of this situation, but in situation y, the opposite of x happened, so you don't have to worry) and (2) as a source of argumentation strategies.

References

Sycara, K. 1985. "Persuasive argumentation in resolution of collective bargaining impasses." *Proceedings of the Seventh Annual Conference of the Cognitive Science Society.*

Sycara, K. 1987. "Finding Creative Solutions in Adversarial Impasses." Submitted to the 1987 Conference of the Cognitive Science Society.

Selected Bibliography

Alterman, R. 1986. "An Adaptive Planner". *Proceedings of AAAI-86*, pp. 65-69.

Alterman, R. 1985. "Adaptive Planning: Refitting old plans to new situations". *Proceedings of the Seventh Annual Conference of the Cognitive Science Society.*

Ashley, K. 1986. "Knowing What to Ask Next and Why: Asking Pertinent Questions Using Cases and Hypotheticals." *Proceedings of the Eighth Annual Conference of the Cognitive Science Society.*

Ashley, K. 1985. Reasoning by Analogy: A Survey of Selected A.I. Research with Implications for Legal Expert Systems. In *Computing Power and Legal Reasoning*, Charles Walter (Ed), West Publishing Co., St. Paul, MN.

Ashley, K. D. and Rissland, E. L. 1987. "Creating Neighborhoods of Cases: Projections Through a Case Space." Submitted to *IJCAI-87.*

Ashley, K. D., & Rissland, E. L. "Compare and Contrast, A Test of Expertise." To appear Proceedings AAAI-87, Seattle.

Burstein, M. 1986. "Concept Formation by Incremental Analogical Reasoning and Debugging". In *Machine Learning*, Vol. 2. Michalski, Carbonell and Mitchell (Eds). pp. 371-393. Morgan Kaufmann.

Carbonell, J.G. 1986. "Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition." In *Machine Learning*, Vol. 2. Michalski, Carbonell and Mitchell (Eds). pp. 371-392. Morgan Kaufmann.

Carbonell, J.G. 1983. "Learning by Analogy: Formulating and Generalizing Plans from Past Experience. In *Machine Learning: An Artificial Intelligence Approach.* Michalski, Carbonell and Mitchell (Eds), Tioga Pub., CA.

Carbonell, J. G. 1982. "Experiential Learning in Analogical Problem Solving." *Proceedings AAAI-82*, Pittsburgh, PA.

Carbonell, J. G. 1981. "A computational model of analogical problem solving." *Proceedings IJCAI-81.*

Carbonell, J. G. 1979. "Planning through adversity: the counterplanning process". *Proceedings IJCAI-79*, Tokyo, Japan. pp. 124-130.

DeJong, G. 1981. "Generalizations based on explanations." *Proceedings IJCAI-81*, Vancouver, B.C., Canada.

Gentner. D. 1983. "Structure-Mapping: A Theoretical Framework for Analogy." *Cognitive Science* 7(2):155-170.

Hammond, K. 1986. *Case-based Planning: An integrated theory of planning, learning and memory.* Ph.D. Thesis, Yale University, 1986.

Hammond, K. 1986. "CHEF: A model of case-based planning." *Proceedings AAAI-86.*

Hammond, K. 1986. "Learning to Anticipate and Avoid Planning Problems through the Explanation of Failures." *Proceedings AAAI-86.*

Hammond, K. 1986. "The Use of Remindings in Planning." *Proceedings of the Eighth Annual Conference of the Cognitive Science Society,* 1986.

Hayes-Roth, B. 1980. Projecting the future for situation assessment and planning. Note N-1600-AF Rand Corp., Santa Monica, CA.

Kolodner, J. L. 1987. "Capitalizing on Failure Through Case-Based Inference." Submitted to AAAI-87.

Kolodner, J. L. 1987. "Coordinating Case-Based and From-Scratch Reasoning." Submitted to Ninth Annual Meeting of Cognitive Science Society.

Kolodner, J. L. 1983. "Reconstructive Memory: A Computer Model." *Cognitive Science,* 7(4): 281-328.

Kolodner, J. L. 1983. "Maintaining Organization in a Dynamic Long-Term Memory." *Cognitive Science,* 7(4): 243-280.

Kolodner, J. L. & Cullingford, R. E. 1986. Toward a Memory Architecture that Supports Reminding. *Proceedings of the Eighth Conference of the Cognitive Science Society.*

Kolodner, J.L. and Simpson, R.L. 1984. "Experience and Problem Solving: A Framework." *Proceedings of the Sixth Annual Conference of the Cognitive Science Society.*

Kolodner, J. L., Simpson, R. L., & Sycara-Cyranski, K. 1985. "A Process Model of Case-Based Reasoning in Problem Solving." *Proceedings IJCAI-85.*

Lebowitz, M. (1986). "Concept learning in a rich input domain: generalization-based memory." In *Machine Learning: An Artificial Intelligence Approach.* Vol. 2. Michalski, Carbonell, and Mitchell (Eds.) pp. 193-214.

Lehnert, W.G. 1987. "Case-Based problem solving with a large knowledge base of learned cases". To appear in *Proceedings AAAI-87*, Seattle.

Lehnert, W. G. 1987. "Word Pronunciation as a Problem in Case-Based Reasoning". To appear in *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, in press.

Riesbeck, C. K. 1981. "Failure-driven reminding for incremental learning." *Proceedings IJCAI-81*, Vancouver, B.C., pp. 115-20.

Rissland, E. L. 1983. "Examples in Legal Reasoning: Legal Hypotheticals." *Proceedings IJCAI-83*, Karlsruhe, Germany.

Rissland, E. L. & Ashley, K. 1987. "HYPO: A case-based reasoning system."

Rissland, E. L., & Ashley, K. D. 1986. "Hypotheticals as Heuristic Device." *Proceedings AAAI-86.*

Rissland, E. L. & Ashley, K. D. 1987 "A Case-Based System for Trade Secrets Law". *Proceedings First International Conference on AI & Law.*

Rissland, E. L. & Collins, R. T. 1986. "The Law as Learning System." *Proceedings of the Eighth Annual Conference of the Cognitive Science*

Simpson, R.L. 1985. *A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation.* Ph.D. Thesis. Technical Report No. GIT-ICS-85/18. School of Information and Computer Science, Georgia Institute of Technology.

Stanfill, C., and Waltz, D. 1986. "Toward memory-based reasoning." *Communications of the ACM*, vol. 29, no.12. pp.1213-28.

Wall, R. and Rissland, E. 1982. "Scenarios as an aid to planning." *Proceedings AAAI-82*, Pittsburgh.

Winston, P. H. 1983. "Learning Physical Descriptions from Functional Definitions, Examples and Precedents." *Proceedings AAAI-83.*

Winston, P. H. 1984. "Learning New Principles from Precedents and Exercises." *Artificial Intelligence* 19:321-350.