Representation and Management Issues for Case-Based Reasoning Systems

Igor Jurišica juris@ai.utoronto.ca
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 1A4, Canada

September 15, 1993

Contents

1 Introduction

In many areas, the reasoning process is based on individual expertise, rather than on general knowledge (e.g., legal reasoning, medical diagnosis, etc.). Case-based reasoning is a paradigm based on remembering previous experience (cases). In general, a case consists of a problem description and an outcome. Case-based problem solving is performed by retrieving similar cases from a case knowledge base and reusing these cases either with or without adaptation.

A case-based reasoning system (CBRS) can be seen as a special type of a knowledge-based system. Case-based systems deal with cases or episodes, which represent instances of concepts (or their generic form). They are special in the sense of different knowledge they use and consequently different representation and management requirements they have. Case-based reasoning also shares similar features with similarity-based reasoning (Tversky, 1977; Gaschnig, 1979; Lebowitz, 1986; Bailey, Thompson and Feinstein, 1987; Gentner, 1989; Ross, 1989; Vosniadou and Ortony, 1989; Gentner and Forbus, 1991; Heckerman, 1991), instance-based reasoning (Aha, Kibler and Albert, 1991; Aha, 1992; Scott and Sage, 1992; Broos and Branting, 1993), analogy-based reasoning (Carbonell, 1981; Carbonell, 1983a; Gentner, 1983; Burstein, 1986; Carbonell and Veloso, 1988; Adelson et al., 1988; Keane, 1988; Burstein, 1989; Vosniadou and Ortony, 1989; Indurkhya, 1990; Cook, 1991; Dierbach and Chester, 1991; Deraedt and Bruynooghe, 1992; Bhansali and Harandi, 1993), and reinforcement learning (Sutton, 1992; Ram and Santamaría, 1993).

The performance of case-based reasoning system directly depends on the quality and quantity of its encoded knowledge: on the one hand, if there is not enough knowledge, the problem-solving capabilities are limited; on the other hand, the system can become useless because of unmanageable amounts of knowledge.

Case-based reasoning systems can also be viewed as a continuous knowledge acquisition and learning systems. Knowledge acquisition, together with other types of learning, are essential characteristics of an intelligent system. Even though learning is common for humans, it is difficult to build learning machines. Despite the differences between existing machine learning algorithms, they all allow for the changes in the system that are adaptive in the sense that the system's efficiency improves over time (Simon, 1983). This requires that the knowledge is represented in structures that can be modified when new knowledge arrives (McCarthy, 1958; Buchanan and Wilkins, 1993). Thus, the knowledge organization often is the source of power of the learning method.

Even though the fields of machine learning and knowledge acquisition are closely related, research communities of these two fields are largely separate. Traditionally, machine learning researchers have emphasized concept learning from a library of preclassified training examples. In contrast, knowledge acquisition researchers emphasize a broad range of expert tasks with semi-automatic approaches that have a human expert in the acquisition loop.

There has not been sufficient progress in the area of knowledge acquisition, mainly because the knowledge acquisition problem can be defined only after the issues of knowledge organization, representation, and inference are settled (Buchanan and Wilkins, 1993).

Knowledge-based system should satisfy three main concerns of related areas, namely:

1. The program should have a model that is well understood. Moreover, if the program captures general principles of data- and hypothesis-directed reasoning, then it can be used as the basis for a user model - *a psychological issue.*

2. The control knowledge should be separated from the factual knowledge - *a knowledge engineering issue.*

3. Findings, hypotheses, evidence, justifications, structure, and strategy should be distinguished

- *an epistemology issue.*

Above mentioned issues are also related to case-based reasoning systems; however, because of the uniqueness of this approach, some specific problems have been diminished and others appeared. Previous work in case-based reasoning has been oriented mainly toward specific domains. This produced numerous prototypes and insight into the difficult issues and structure of the paradigm. However, there is almost no notion of general principles and domain-independent systems yet. It is also difficult to compare different systems since there are not precise statements of the basic underlying algorithms, the systems operate in different areas, and the case bases are small.

1.1 Motivation

The representation and management of large case knowledge bases are not trivial tasks and have not been sufficiently elaborated so far. These tasks can be characterized as a process of representation, visualization, and application of episodic knowledge: specifying the content of the case (episode), describing a retrieval algorithm and precisely defining matching criteria, characterizing strategies to make past cases fit the new problems (adaptation), and case base evolution through application of machine learning.

The representation and management issues for case-based reasoning are challenging because many applications require large case bases with space- and time-efficient processing. Even though the representation and management of cases can be handled similarly as in knowledge-based systems, there are particular differences which need to be considered, namely:

- Cases are suitable for capturing problem-solving processes by representing episodes of these processes, and thus are suitable not only in easily formalizable areas (as are traditional knowledge-based systems) but in less formalizable areas as well. Because knowledge acquisition is supported by learning from past experience, CBRSs are able to supplement weak domain models.

- A case can represent a complete solution (plan, design) which can be retrieved, adapted, and reused later, eliminating the need for decomposition and recomposition.

- In contrast to earlier AI systems, case-based system use specific past experience in solving problems (however, accumulated cases can be processed and generalized later.).

1.2 Outline

The mainstream theme of this document is to characterize problems in case-based reasoning systems and present available solutions to it. The document is organized as follows. Section 2 defines the case-based reasoning approach and identifies main components of such systems as well as basic requirements for them. Section 3 reviews diverse representations of episodic knowledge. Section 4 discusses different approaches to case classification. Section 5 presents retrieval mechanisms, identifying various similarity measures and classification strategies. Section 6 presents adaptation methods used in CBRSs. Section 7 characterizes relations of CBR to other areas in artificial intelligence. Attention is focussed on common aspects and applied ideas. Section 8 discusses both evaluation methods which are unique to the CBRSs and methods that are shared with other knowledge-based systems. Section 9 concludes the study with a summary and characterization of open research issues in the presented areas.

2 Case-Based Systems

2.1 Case-Based Reasoning

Case-based reasoning is usually viewed as a process of remembering and recalling one or a small set of actual instances or generic cases. The decision making is based on comparisons between the new situation and the old experience; thus, the quality of reasoning depends on the quality and quantity of cases in a case base. With increased number of unique cases (instances or generic cases), problem-solving capabilities of CBRSs improve while the effectiveness of the system (at least in terms of efficiency) may decrease.

On the one hand, if the cases are not unique enough, i.e., the cases are highly correlated, then the system will be limited in the diversity of solutions it can generate (O'Leary, 1993). On the other hand, if the case base is small, then the possible solutions are limited. However, based on the experience in machine learning (Minton, 1988), and more recently in the case-based reasoning community (Ram and Santamaría, 1993; Anthony G. Francis and Ram, 1993), the system must either restrict the number of learned cases or an extra effort must be put towards developing of storage and retrieval mechanisms in order to cope with the utility problem.

The major processes shared by the reasoner that does case-based reasoning are as follows (see also figure 1):

- **Case storage** - representing all important features of the episode (actual or hypothetical; an instance or generic) in a case base. Representation of cases can be flat or can be hierarchically organized.
- **Case retrieval** - retrieving relevant cases from the case base. In addition, the reasoner must evaluate the solution (store the feedback information), in order to make sure that pure solutions are not repeated along with the good ones (this is also referred to as validated retrieval (Simoudis and Miller, 1990)).

2.2 Case-Based Reasoning Systems

Case-based reasoning systems differ in decision making methods. Based on this classification, the following systems are usually recognized:

1. **Interpretative systems** compare new situation to recalled experience. The interpretation is used when the problem is not well understood and when there is a need to criticize a solution. Cases are used to provide justification for solutions, evaluation of solutions when no clear-cut methods are available, and interpretation of situations when definitions of the situation's boundaries are not well-defined. Interpretative systems can be further distinguished into the following:
    - **Justification** is the way of making persuasive arguments to convince others (Rissland and Ashley, 1987a; Berman and Hafner, 1993).
    - **Interpretation** is performed by classification of a concept, based on a fuzzy definition of borders. If, however, no case is found to be similar enough to a new problem, hypothetical situations can be considered (Ashley, 1988; Bareiss, 1988).
    - **Prediction** (or projection) is one of the most important parts of the evaluative component of any planning or decision making system. If everything about the situation is known, projection is just running known inferences forward from a solution. However,

in many cases, all effects cannot be predicted. Here, cases provide a way to projecting effects based on what has been true in the past (Goodman, 1989).

A natural application for interpretative systems is the American legal system: relevant cases form an argument for or against a case. There is no need to modify the past case, the problem is solved by finding and presenting the relevant cases.

2. **Problem-solving systems** derive solutions to a new problems by adapting an old solution for the current situation. Case-based reasoning system is considered to be a problem solver if:

   - it can learn from mistakes, thus, it will not repeat the same mistake twice;
   - it will keep trying to find better solution, even though one is already available;
   - it will not consider available results immediately; it will first modify them to fit the current needs.

   A problem-solving CBRS is applied to problems, where simple presentation of past cases does not suffice for finding a solution (e.g., planning, diagnosis, design); a case needs to be modified to fit a new problem: some parts may be deleted because they are not needed for the new problem, some parts may be added since they are missing in the old case, and some parts may be changed in a specific way (e.g., in a software design system, the adaptation process may looks like: $put\_on(green\ box)$ is changed to $put\_on(blue\ box)$, or $for(i == 0; i < 100; i++)$ is changed to $for(i == 0; i < 3; i++)$). This process may also rely on generalizations, specializations, or on analogical problem-solving (different strategies for modification, usually called adaptation strategies, are discussed in more detail in section 6).

The case-based reasoning cycle (figure 1) is in general almost the same for both interpretative and problem-solving case-based system; it starts with retrieval which results in ballpark solution proposal. The main distinction between interpretative and problem-solving CBRSs is that while the former group of systems does not use adaptation at all the latter group rely heavily on adaptation. After this intermediate step, both approaches criticize and evaluate the solution and based on the result, either the final solution is proposed or an additional adaptation is performed. If the retrieved solution does not suffice (either because of small relevance of it or because additional solutions are necessary) the system may alter the original query and retrieve additional cases.

Considering different methods used in case-based reasoning systems, one can classify existing CBRSs into the following categories:

- **Statistically oriented** - conditional probabilities that the problem should be solved as the previous case are computed (Stanfill and Waltz, 1986).

- **Model based** - explain the cases in terms of a theoretical model. Determine if past explanation applies to a new situation (Koton, 1988b; Féret and Glasgow, 1993).

- **Planning/design oriented** - cases are instantiated plans that satisfy goals or avoid goal conflicts; they are used as a solution templates (Sycara, 1987; Hammond, 1989a; Goodman, 1989; Navinchandra, Sycara and Narasimhan, 1991; Pankakoski et al., 1991; Tanaka, Hattori and Sueda, 1992; Hanks and Weld, 1992; Berger, 1993; Kolodner and Wills, 1993).

- **Exemplar based** - cases are exemplars of concepts; they represent examples (prototypes) of simple instances of cases (Bareiss, 1988; Fertig and Gelertner, 1991).

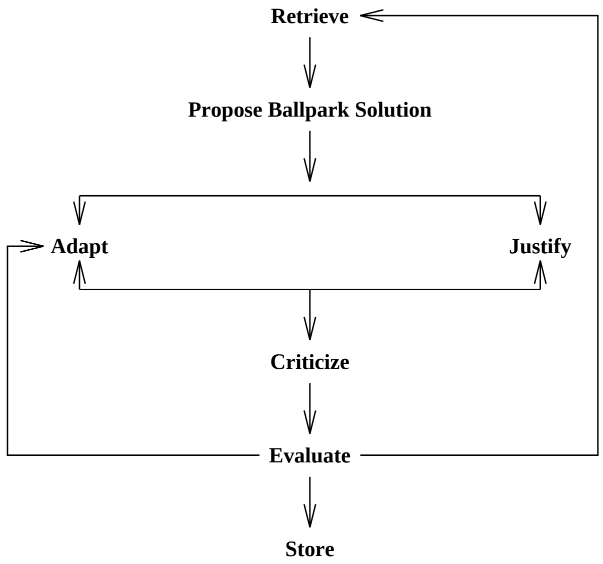

Figure 1: The case-based reasoning cycle

- **Adversarial or precedent based** - cases (precedents) justify arguments either for or against the problem (Ashley, 1988; Gonzalez and Laureano-Ortiz, 1992; Berman and Hafner, 1993).

2.3 Application Areas of Case-Based Systems

2.3.1 Case-Based Design

In case-based design, problems are defined as a set of constraints; the problem solver is required to provide a concrete artifact that solves the constraint problem (Kolodner, 1992). The constraints can either *underspecify* the problem, i.e., there are many possible solutions or *overconstrain* the problem, i.e., there is no solution if all constraints are fulfilled. In solving either of these cases, the design specifications must be re-specified while solving the problem.

Case-based reasoning systems are especially useful for the problems that are hard to solve both in one chunk and in a decomposed form because of the strong interaction between individual pieces. Solving the problem by adapting an old solution allows the problem solver to avoid dealing with many constraints, and keeps it from having to break the problem into pieces needing recomposition.

Another interesting contribution of CBR paradigm for design is pointing out problems with proposed solution. In almost all design problems, more than one case is necessary to solve the problem. The reason for that is the complexity of problems and the fact that design problems tend to be large. Within the CBR approach, decomposition and recomposition are avoided, as are large constraint satisfaction and relaxation problems. A brief description of some design-oriented case-based systems follows:

- JULIA (Kolodner, 1987; Hinrichs, 1988; Hinrichs, 1989) plans meals. The emphasize is put on the final product, which has to meet certain constraints rather than on individual steps towards the main goal, as it is in planning. Cases are organized according to their features in a discrimination net.

- KRITIK (Goel, 1989; Goel and Chandrasekaran, 1989) is a combination of case-based and model-based reasoning for design of small mechanical assemblies. Cases are used to propose solutions; the model is used to verify these solutions, to point out where adaptation is needed, and to suggest specific adaptation.

- CLAVIER (Barletta and Hennessy, 1989) is being used to lay out pieces made of composite materials in an oven to bake. There is no complete causal model of what works and why; however, the system's goal is to avoid placing pieces in the wrong places. Almost always, several cases are used to do the design. One case provides an overall layout, which is adapted appropriately, the others are used to fill in holes in the layout. This system is distinct from others because it is used on a daily basis in Lockheed Inc. since September 1990.
- PERSUADER (Sycara, 1987) solves labor management disputes by adapting similar contracts. If no possible contract can be retrieved, the system generates a new contract from scratch. The emphasize is on meeting different constraints simultaneously.
- XBE (Pankakoski et al., 1991) is applied to manufacturing systems design. The system allows the user to represent and use different types of knowledge; it also uses hypothetical cases (similarly as HYPO system (Ashley, 1990));
- ARCHIE (Pearce et al., 1992) is used to help architects in the high-level task of conceptual design. The cases are used both for proposing and critiquing. A case consists of design goals, design plans, outcomes, and lessons to be learned.
- DéjàVu (Smyth and Cunningham, 1992) is a hierarchical CBR system for the development of plant control software for controlling autonomous vehicles in loading and unloading metal coils in a steel milling process.

2.3.2 Case-Based Planning

Planning is the process of coming up with a sequence of steps or a schedule for achieving a particular state of the world. There are several problems that must be dealt with in planning, namely:

1. **problem of protections** - good plans are sequenced and a planner has to ensure that the effects of individual steps are projected into the future; and
2. **problem of preconditions** - a planner has to ensure that all preconditions of individual steps are fulfilled before their scheduling.

In addition to the above mentioned problems, even more complex problems can arise if the number of goals competing for achievement at any time is quite high. In general, there are two possible ways to tackle this problem:

1. **Achieve each goal independently of the others**. Then, planning and execution time are at least the sum of achieving each goal (because of interactions, planning and execution can be even more complex). This requires decomposition and recomposition of plans.
2. **Achieve several goals simultaneously**. Solving the goals in conjunction makes the problem significantly harder.

Case-based planning and case-based design systems are quite similar to each other. Both types of systems are useful for the problems that are hard to solve in one chunk, but cannot be easily decomposed, and solved separately, because of the strong interaction between individual pieces. Here, individual steps imply different constraints which have to be satisfied.

Because of these similarities between case-based planning and case-based design systems, it is sometimes difficult to judge if a particular system belongs to the former or latter category. The only distinctive factor in such cases is the emphasis, either on the planning or on the design part of the system.

Case-based planning uses previously successful plans, classified according to the conjunction of goals they achieve. Case-based planning inherently diminishes the problem of combining planning and execution. A brief description of some case-based planning systems follows:

- CHEF (Hammond, 1989a) is an early case-based planner, which works in the domain of Chinese recipes. CHEF address the problem of anticipating problems before execution time by learning from its problematic experiences.

- PLEXUS (Alterman, 1988) is a program to demonstrate the skills required to ride a subway. PLEXUS is able to do execution-time repairs by adapting and substituting semantically-similar steps for those that have failed.

- TRUCKER (Hammond, 1989b; Hammond et al., 1993) keeps track of pending goals and takes advantage of opportunities that arise and allow it to achieve these goals earlier than expected.

- MEDIC (Turner, 1989) is a diagnosis program, which is able to reuse previous plans for diagnosis; it is flexible enough in its reuse to be able to follow up on unexpected turns of events. MEDIC also compares the importance of goals and decides which plan to use to achieve the most important goal. Because of its emphasize on planning, the system is considered to be a case-based planning system, rather than case-based diagnostic system.

- CSI BATTLE PLANNER (Goodman, 1989) is an example where cases are used to criticize and repair plans before they are executed.

- SPA (Hanks and Weld, 1992) is a domain-independent case-based planning system. A plan is represented as a set of steps, various links, and constraints describing interrelationships among the steps.

2.3.3 Case-Based Diagnosis

In case-based diagnosis, a solver is given a set of symptoms and is asked to explain them. On the one hand, diagnosis is similar to the classification problem in cases when only a small number of possible explanations exist. On the other hand, if the space of possible explanations is significant, one can view diagnosis as the problem of creating an explanation. Similarly to the comparison of design and planning case-based systems, case-based diagnosis system can rely on planning and thus can be classified as a planning system (Turner, 1989). However, there are also diagnosis systems which are model-based (Koton, 1988b; Goel, 1989; Goel and Chandrasekaran, 1989; Birnbaum et al., 1990; Féret and Glasgow, 1993). In these cases, categorization is easier since the distinctive features are easily perceivable. A brief description of some case-based diagnosis systems follows:

- SHRINK (Kolodner and Kolodner, 1987) is an early case-based reasoning system, designed to be a psychiatric diagnostician. Here, the diagnosis from a previous case is used to generate a hypothesis about the diagnosis in the new case.

- CASEY (Koton, 1988b) is a system applied for diagnosis of heart problems; it adapts the diagnosis of previous heart patients to new patients. CASEY is built on top of an existing model-based diagnostic program. Because adaptation is based on a valid causal model, its diagnoses are as accurate as those made from scratch based on the same causal model and they also tend to be more time efficient.

- PROTOS (Bareiss, 1989a), a hearing disorders diagnostic system, is designed to ensure that cases are used in pointing the way out of previously experienced reasoning predicaments. In its application domain, many of the diagnoses manifest themselves in similar ways, and only subtle differences differentiate them. By using feedback from the user, PROTOS learns these subtle differences.

- EAD (Explanation-Aided Diagnosis) (Féret and Glasgow, 1993) is a model-based diagnostic system with a case-based reasoning module, applied for diagnosis of complex devices. The system uses hierarchical knowledge representation and the model-based reasoner is used to identify relevant experience. The case-based module critiques the results obtained by the model-based reasoner, and also helps to consider unexplored alternative hypotheses.

2.3.4 Case-Based Explanation

Explanation is introduced to the CBR terminology in the connection with the plan or solution failure. The system may try to construct an explanation of the reason for the failure in order that the plan or solution may be repaired and the case base modified to avoid such a failure in the future.

Explaining anomalies is prevalent in all of our problem solving and understanding activities. Explanation has been called the credit assignment problem and the blame assignment problem, depending whether it explains a success or a failure.

A case-based explanation (Schank, 1986; Ram, 1993b; Ram, 1993a) explains a phenomenon by remembering a similar phenomenon, borrowing its explanation, and adapting it to fit. In this context, CBR fits very naturally and will improve the explanation process. A case-based explanation requires the following mechanisms:

- a retrieval mechanism that can retrieve similar cases;

- an adaptation mechanism that must be quite creative;

- a validation mechanism that can decide if a proposed explanation has any merit.

An interesting application of case-based explanation is presented in (Louis, McGraw and Wyckoff, 1992). Here, a case-based approach is used to explain solutions generated by a genetic algorithm. This task is quite difficult because the solutions cannot be easily explained. An explanation is later used to tune the genetic algorithm. Thus, case-based explanation helps in developing an adaptive genetic algorithm.

3 Case Representation

There is still a lack of consensus as to exactly what information should be represented in a case. A case is an arbitrary set of features, used for describing a particular concept. Considering its form, a case can be a simple concept or a connected set of subcases; it can be a specific instance or a generalization. Many different case representations have been proposed so far. There are two main reasons for this:

1. the motivation for CBR came from different directions, namely a desire to model human performance of knowledge-intensive tasks, and an attempt to use existing database technology to store and access knowledge; and

2. different application areas required different functional requirements on the case representation.

3.1 Approaches to Case Representation

Even though most of the work on representation in CBR is related to particular applications, there have also been made theoretical contributions. A theoretical analysis (in terms of default logic) of a case representation in early CBR systems is presented in (Koton and Chase, 1989). Cases are considered to be a collection of features (attribute-value pairs) which are organized in a generalization structures. In this study, gens are structures which represent groups of similar cases. Each gen stores a list of common features (norms) for all cases it organizes. A case base, usually referred to as a case memory, is a structure which represents and relates either individual cases or gens. The case base is usually represented as a discrimination net. The features that differentiate cases or gens from the parent gen are called diffs. These links are labeled by a particular feature which differentiates the more specific node from the more generic one.

Above mentioned reasons resulted in case representation ranging from simple flat lists of features (Bareiss, 1988; Gonzalez and Laureano-Ortiz, 1992; Kitano, Shibata and Shimazu, 1993) through complex descriptions of episodic experience (Bradtke and Lehnert, 1988a; Hammond, 1989a; Brandau, Lemmon and Lafond, 1991; Hennessy and Hinkle, 1992; Branting, 1992), to vectors of analog values (Ram and Santamaría, 1993). Some of these differences are captured next:

- Bradtke and Lehnert's CBS system (Case-Based Search system) (Bradtke and Lehnert, 1988a) represents cases as complete problem solutions. A case is a list of problem states from a start state to a goal state. Each case implicitly contains many others as sub-cases.

- Hammond's CHEF (Hammond, 1989a) represents cases as plans, which are either positive (they lead to a successful solution) or negative (they represent failures). Failures are used to predict potential problems. For adaptation process, failure explanation, and credit assignment in CHEF, a strong domain model is used.

- Kambhampati's PRIAR (Kambhampati, 1989) uses plans augmented with causal representation. The problem of refitting of old plans to a new situations is guided by the protection annotations left by a nonlinear planner.

- Carbonell's work on derivational analogy (Carbonell, 1983a; Carbonell, 1983b; Carbonell and Veloso, 1988; Veloso and Carbonell, 1989) is more elaborated, compared to CHEF and PRIAR, in the sense that plans include the rationale for decisions as well as the results.

- Kolodner (Kolodner, 1989) and Zito-Wolf (Zito-Wolf, 1993), suggest to use more vague definition of a case. According to them, a case is simply a portion of richly interconnected memory, allowing the application of multiple cases to a single problem.

- Alterman (Alterman, 1989; Alterman and Wentworth, 1989) considers cases (plans) to be content-dependent in that much of the meaning associated with the case is derived from its relative position in an event concept. Because cases include background knowledge about the goals, PLEXUS does not have a significant case base; instead, the mechanism for adapting old plans is used.

- Branting's GREBE system (Branting, 1991) uses a representation language in which arbitrary causal, temporal, and intensional relations in a legal domain could be stated explicitly through propositions (an average case consists of 89 propositions).

- The NETTRAC system (Brandau, Lemmon and Lafond, 1991) represents cases as a complete history of a single network problem and all the control actions that alleviated the problem. It is not possible to divide the case into a subcases since it would sacrifice the context of the complex case.

- In Jones's BRAINSTORMER (Jones, 1992), cases represent abstract planning advice. This allows using of the same cases in different planning situations, which is usually not possible in other CBR systems. Moreover, fewer number of abstract cases are required for the same functionality compared to the system with instances of cases.

- In the SQUAD system (Kitano, Shibata and Shimazu, 1993), cases represent analysis of problems under the status quo, possible causes, counter-measures taken, and effects of the counter-measures. The case structure is dynamically changing as new cases are acquired. This system is different from others because it is the real-world application [1] and relational database is used for case storage.

If we summarize presented approaches, the following options for case representation can be characterized:

- Stored cases represent either problem description and solution, or traces.

- All cases are stored, even though some of them are quite similar. An alternative is to store only "less" similar cases (paradigm cases) or to store valid generalizations of cases.

- Cases are stored either as monolithic structures, used individually, or as a connected sets of events that are constructed at retrieval time or during the learning process.

- Cases are stored with additional information (e.g., causal links) which is acquired during case use, case modification, or storage.

As was presented, case representation depends (at least partially) on the task for which a CBR system is used. In (Branting, 1992) it is argued that for a general-purpose knowledge base, intended to support a variety of different tasks, a featural representation language, capable of expressing only case attributes relevant to a single task, is not sufficient. Instead, a relational representation is required. This should form a common ground which serves all applications.

[1] In (Kitano, Shibata and Shimazu, 1993) it is reported that the system uses 25,000 cases with 3,000 cases added every year, 1,500 employees is responsible for case accumulation, and the utilization of the system in the NEC Corporation saves $100 million per year.

3.2 Case Base Construction

An important issue, related to representation, is case base construction. As was already mentioned, it is not a trivial task to decide how to represent a case in a particular domain and what cases are worth storing (because of the utility problem (Minton, 1988; Hunter, 1989; Anthony G. Francis and Ram, 1993)). Moreover, the representation can be extremely complex which makes it difficult to acquire a sufficient number of cases for a particular application. These problems are the main reasons why many CBR systems have relatively small case bases.

The process of a case base construction can be simplified as follows:

- A case base is created automatically from an existing database. An example of this approach is the FGP Machine (Fetch, Generalize, and Project) (Fertig and Gelertner, 1991). The FGP Machine is a software architecture that processes large databases [2] and makes the knowledge from them explicit.

- A special intelligent system is used as an assistance to case acquisition. A representative of this approach is the CBF system (Case-Based Formulation) (Branting, 1992). The system is based on the use of previous cases as a model and guide for expressing new cases. A copy-and-edit approach is used, where a frame is added to a case base by finding a similar frame, copying it, and modifying the copy. This process can use either single case or multiple cases. The system allows a goal specification in order to constrain the matching process; it is an interactive method, automatically insuring consistency with existing collection of related facts with a growing case base.

4 Case Classification

The basic assumption of the case-based reasoning system is that all relevant cases will be retrieved efficiently. In order to guarantee efficient retrieval, case storage requires special techniques to be applied. Several general approaches have been proposed for this problem, namely:

1. Classification of all cases into a hierarchy, grouping relevant cases together (in CBR literature this process is usually referred to as the *indexing problem.*

2. Implementation of a CBR system on a parallel computer without using case classification (Stanfill and Waltz, 1986; Waltz, 1989; Rubin, 1992).

3. Classification is supported by parallel implementation (Kolodner, 1989).

4. Neither classification nor the parallel implementation is used (Thagard et al., 1990; Hennessy and Hinkle, 1991; Gentner and Forbus, 1991). However, this approach is only usable for small case bases.

4.1 Classification Methods

Different systems use different classification methods. There are several possibilities to categorize these methods. Porter (Porter, 1989) categorize the methods based on the source of knowledge for classification. He identifies two possibilities: an explicit representation using a fixed vocabulary

[2] However, the current implementation is inadequate to handle case databases of more than 1,000 cases assuming 10-20 features per case. The largest test-case-base consists of 643 cases, each having 3.6 features on average.

(the vocabulary is predefined which makes this approach static), and a dynamic representation which stores cases in hierarchies (because there is no predefined structure, this approach is more flexible and adaptable). An obvious advantage of the latter approaches is the adaptability and flexibility which makes it suitable for large case bases and especially for inter-domain reasoning. The other categorization is based on the algorithm used, namely:

- **Nearest-neighbor** retrieves cases based on a weighted sum of features in the input case that match cases in memory (Hennessy and Hinkle, 1992; Pearce et al., 1992; Gonzalez and Laureano-Ortiz, 1992; Cardie, 1993a). The method is efficient if the retrieval goal is not well-defined or if few cases are available. The problem with the method is that the feature weights are context dependent and thus they must be either predefined (and consequently fixed) or a special method must be used (e.g., inductive learning) to specify them in more dynamic way (not necessarily after each case acquisition).

- The **inductive** approach (e.g., ID3 (Quinlan, 1986), UNIMEM (Lebowitz, 1987), COBWEB (Fisher, 1987)) automatically extracts relevant features for retrieval or creates hierarchies of cases by comparing the classification and discrimination abilities (the ability to characterize and distinguish a particular case from the others) of individual features (Goodman, 1989; Simoudis, 1992). This method is applicable only if sufficient number of cases are available and the goal of retrieval is well-defined. However, the learning process is quite time consuming. This is the reason why in some applications the classification is done only as a batch process. In some implementations, only surface features (easy to observe) are classified using the inductive method while deep features (difficult to observe) are classified by a human expert.

- **Discrimination net** uses both previous approaches (Kolodner, 1983; Bradtke and Lehnert, 1988a; Hammond, 1989a; Cardie, 1993a; Rissland et al., 1993).

- An **explanation-based** approach uses past experience to alter the classification (DeJong and Mooney, 1986; Barletta and Mark, 1988; Ram, 1993b). The main advantage of the method is the possibility to automate the case classificaton. This process helps to incrementally refine the system's understanding of the case. The EBI system (Explanation-Based Indexing) (Barletta and Mark, 1988) can use any automatically selected feature for a classification; however, the system may select features which are hard to observe (or compute) or which does not provide sufficient discrimination (e.g., many cases would be retrieved using this feature). The AQUA system (Asking Questions and Understanding Answers) (Ram, 1993b; Ram, 1993a) overcome this drawback by selecting only easy to observe or easy to compute features. But even this approach has some drawbacks: only a predefined types of classifications may be learned and in some cases, a classification may be created for a one-time only event.

- A **knowledge-based** approach uses existing knowledge of each case in the case library to determine the importance of individual features for retrieving each case (Lauzon and Rose, 1993; Kitano, Shibata and Shimazu, 1993). Even though all approaches up to date rely on domain expertise, this approach is the most interesting due to its adaptability and flexibility, which makes it suitable for inter-domain problem-solving.

4.2 Representational Requirements for Classification

The problem of organizing cases so that the retrieval of relevant experience is efficient is one of the most important issues in case-based reasoning. In (Schank et al., 1990), an attempt is made to unify different approaches to the indexing problem. The authors present ideas about the representational issues required for efficient retrieval. More theoretical work on this problem was presented in (Domeshek, 1992). However, in both cases only certain domains (story understanding) were considered and no attempts were made to consider other domains.

Following presented differences in approaches to knowledge retrieval the following categorization can be defined:

- the algorithm for knowledge classification either uses a certain (predefined) class of features which are used in accessing cases (Kolodner, 1983; Birnbaum and Collins, 1989; Hunter, 1989; Owens, 1989; Hammond, 1989a) or is based on the representation hierarchy (Porter et al., 1988; Lauzon and Rose, 1993);
- knowledge can be stored as is or a generalization hierarchy may be created (Shinn, 1988; Branting, 1989; Thompson and Langley, 1989; Collins and Birnbaum, 1990; Fertig and Gelertner, 1991; Jones, 1991; Yang, Robertson and Lee, 1993; Ram, 1993b);
- the algorithm can depend heavily on a special computer architecture (Stanfill and Waltz, 1986; Kolodner, 1989; Waltz, 1989; Rubin, 1992), or it can be implemented on a standard computer;
- the classification algorithm assumes large, real-world case bases (Kitano, Shibata and Shimazu, 1993) or it can assume only small ones;
- in addition to functionality, the algorithm may or may not be psychologically plausible.

As was already mentioned, the main task of case-based reasoning system is to retrieve the most relevant cases from a case base when required. The process of classification is crucial for the system's performance. If the system uses a predefined set of features for a classification, then its problem-solving abilities are constrained since the system becomes a single context system (Porter, 1989). Several strategies for solving the problem of predefined indexing vocabularies were identified, namely:

1. All features are used for classification. This naive approach suggests an extension that probably not all features are equally useful. Thus, next approaches try to (somehow) select appropriate features for indexing.
2. Only less computationally expensive features are considered for classification. The problem here is that these features are efficient, but they may result in less than perfect retrieval.
3. Use only features given in the initial description of the problem situation ("low-level" features). This approach is simple and straightforward, but can be too simple to be useful in some domains.
4. More abstract properties of the problem description are generated and only these are used for classification. The main problem of this approach is with derivation of more abstract features because the need for a deep knowledge is required and in particular application this knowledge may not be available.
5. A combination of 3 and 4. This approach can overcome the drawbacks of both approaches.

Presented approaches clearly show that we can make either the initial process (case acquisition and case base construction) more difficult or the reasoning process slower. If the initial knowledge is shallow, then the knowledge acquisition is going to be simpler than in the case when the initial knowledge has also deep structures.

Another important issue is that regardless of where the most work is done, the indexing vocabulary construction is a difficult problem; thus, there is a necessity to employ machine learning approaches to help in this process. Another reason for this is that static classification is in many domains useless since it does not take into account the task and context that may change over time (Suzuki, Ohnishi and Shigemasu, 1992; Berman and Hafner, 1993; Kitano, Shibata and Shimazu, 1993; Lauzon and Rose, 1993).

5 Case Retrieval

Retrieval of cases from memory can be characterized as remembering of relevant past experience. Retrieval consists of the following subtasks:

1. **Recall previous cases** - the goal is to retrieve cases relevant to the current problem, e.g., retrieve similar cases. Cases are found according to classification which is created according to features in the current case.

2. **Select the best subset** - only the most promising case (or cases) retrieved in the initial step are selected. For some application only one case is required, for other, small set of relevant cases is important (for adaptation and/or justification).

5.1 Retrieval Strategies

The objective is to retrieve the most relevant cases and to retrieve only a useful number of them. There are three possible selection methods used to retrieve past experience, namely:

1. All cases are pre-classified according to pre-selected set of features (Kolodner, 1983; Birnbaum and Collins, 1989; Hunter, 1989; Owens, 1989; Hammond, 1989a). The problem is that one cannot predict every important feature for all future problems in advance.

2. The problem description is filtered prior to memory retrieval. During this stage, only relevant features are used. The problem with this approach is that the salient features have to be somehow selected for each problem: in (Hammond, 1989a) they are predefined and in (Lauzon and Rose, 1993) they are dynamically specified by the user.

3. Retrieved cases are filtered after retrieval (Rissland and Ashley, 1986; Koton, 1988b; Kolodner, 1989; Simoudis and Miller, 1990). All these methods are domain-dependent and they have restrictions keeping them from being general.

Based on the different representations, retrieval algorithms can be classified as follows:

- **Associative** retrieval classifies any or all features independently of all the other features (e.g., nearest-neighbor classification technique (Gonzalez and Laureano-Ortiz, 1992; Hennessy and Hinkle, 1992; Pearce et al., 1992; Cardie, 1993a)). During the retrieval, the presence or absence of the features (either all or selected subset of them) is counted (in some cases a weighted sum is computed) and the the result is used to judge the overall relevance of the case to the current problem.

- **Hierarchical** retrieval uses feature highly organized into a general-to-specific concept structure (e.g., explanation-based learning or inductive learning used for hierarchy construction);

- **Mixed** retrieval is characterized by use of both previous paradigms (e.g., discrimination net (Kolodner, 1983; Bradtke and Lehnert, 1988a; Hammond, 1989a; Cardie, 1993a; Rissland et al., 1993)). A somewhat different approach is used in the FRANK system (Rissland et al., 1993); if the initial query is unsuccessful, then the CBR-task mechanism alters the particular initial values and resubmits the query. This is similar to the principle how task-oriented retrieval works in the system described in (Lauzon and Rose, 1993); however, this system is more flexible than FRANK system thanks to the knowledge representation language TELOS (Mylopoulos et al., 1990).

From the description of above mentioned retrieval algorithms follows that the associative retrieval is desirable if the flexibility is required. The hierarchical approach is advantageous if the retrieval task is well-defined. The mixed approach shares the advantages of individual approaches and thus can be applied in both cases.

The most problematic parts of the retrieval process are assessing similarity and retrieving similar cases. Psychological experience tells us that cases can be retrieved strictly on the basis of the surface features (superficial, perceivable, intrinsic), on the basis of abstract features (derived, deep), or based on both. In many studies (e.g., (Tversky, 1977; Gentner, 1983; Birnbaum and Collins, 1988; Keane, 1988; Vosniadou and Ortony, 1989; Collins and Birnbaum, 1990; Simoudis and Miller, 1990; Thagard et al., 1990; Gentner and Forbus, 1991; McDougal, Hammond and Seifert, 1991; Kolodner, 1992; Owens, 1993)) researchers are aware of the existence of all three types of remindings; however, there is a disagreement on what features are the most useful and important for retrieval (these features are referred to as salient and are context and task dependent). Despite the different opinions on this issue, it is obvious that the goal is to retrieve the most useful cases and an appropriate number of them. The usefulness of cases depends on the task which has to be fulfilled while the number of required cases depends on the reasoning strategy.

Consequently, it is not a trivial task to decide which of the infinite number of surface features should be represented and used for retrieval, and what kind of abstract features should be generated. There is a tradeoff between an efficient retrieval and a quality of retrieval: the system should represent sufficient number of features while guarantee efficient processing for large case base. In order to control the number of retrieved cases the following order of preference heuristic was suggested (Kolodner, 1989):

1. **goal directed preference** prefers cases that address the current reasoning goal; it orders them according to number of constraints, preferring more constraints;

2. **salient-feature preference** prefers matching on features which are more important in a particular situation over the matching on other features;

3. **specificity preference** prefers more specifically matched cases over less specific matches;

4. **frequency preference** prefers more frequently-accessed cases over less frequently-accessed cases;

5. **recency preference** prefers more recently-accessed over less recently-accessed cases; and

6. **ease-of-adaptation preference** prefers cases that match on harder-to-fix features over those that are easy to adapt.

5.2 Proposing a Ballpark Solution

In this part of the retrieval process, relevant portions of the selected case are extracted to form a ballpark solution to the current problem. For interpretative case-based systems, proposing a ballpark solution involves partitioning the retrieved cases according to the interpretations or solutions they predict. For problem-solving CBR systems, proposing a ballpark solution involves presenting the most similar past experience, either with respect to symptoms (surface features) or to goals (abstract features).

There are several options in creating a ballpark solution. A system can propose a single case or a set of applicable cases. A set of cases can be used for justification, argumentation, or one solution can be constructed using all possible cases. What follows is a brief description of methods used in different systems:

- JULIA (Kolodner, 1987) presents a piece of an old plan as a ballpark solution if it is attempting to derive some piece of a meal plan; it proposes a similar meal plan as a ballpark solution, if the task is to derive a meal plan.
- PROTOS (Bareiss, 1988) uses a coarse evaluation function to distinguish which of the many possible interpretations proposed by the recalled cases is closest to its new case.
- PARADYME (Kolodner, 1989) uses preference heuristic after retrieval to choose the most useful cases from the set of cases returned by memory access functions. Previous cases are used to provide context to the case selection process.
- HYPO (Ashley, 1990) does not start to work in this step since it has necessary information already given in the form of dimensions.
- CLAVIER (Barletta and Hennessy, 1989; Hennessy and Hinkle, 1992) proposes several relevant cases; one case is presented as an overall layout and the other cases are used to complete the layout where required.
- CBA (Case-Based Appraisal) (Gonzalez and Laureano-Ortiz, 1992) retrieves ten most similar cases which are used to compute the similarity match score. The most similar case is used as a basis for a new appraisal, while the others serve as a guideline for an adaptation of it.
- Protein secondary structure prediction system (Leng, Buchanan and Nicholas, 1993) retrieves first k similar cases (protein structures) from the case base (the number of selected cases is variable and depends on the precision of the similarity metric). The most similar protein is considered as a reference case. In the second step, the whole sequence is decomposed into pieces and the structures are analyzed.

Based on the experience from the above mentioned systems, the following issues form a general guideline for proposing a ballpark solution:

- The goals of the reasoner determine where to focus. The internal structure of the old case, especially the dependencies between different parts of the case, tell the reasoner how to expand focus in relevant ways.
- Do as little work as possible - simple adaptations should be considered before more sophisticated ones take place.
- The system should minimize the need for external input during adaptation; existing cases should be used instead.

5.3 Similarity Assessment

Similarity plays a central role in theories of human problem solving and thus is important for the research in artificial intelligence. Even though there are different approaches to similarity, the underlying idea is to classify cases according to some features which allows us to use these cases in similar situations. Usually, an AI system assesses similarity by comparing symbolic representations which describe situations. Intuitively, the goal is to compare those parts of knowledge so that the most similar cases will also be the most useful ones.

The definition of similarity in cognitive science and initial work in this area can be found in (Tversky, 1977; Gick and Holyoak, 1980; Gentner, 1983; Holyoak, 1985; Vosniadou and Ortony, 1989). However, it is believed that similarity has not been well defined up to date. There is some agreement between the CBR community and the cognitive psychology researchers that similarity judgments can be highly goal-dependent or that they can be highly automatic (based on surface similarity). As was mentioned in section 4, in different situation different features will be the most important for retrieval. For fast, easy to use, intra-domain remindings, surface features will be the most useful (quick responses to the environment). For more complex remindings (inter-domain remindings, analogic, or thematic remindings) abstract features may be required, even without the use of any surface features. It was observed, that the frequency of relational remindings increases with expertise in a domain; however, even experts use surface similarity in some situations. A case-based reasoning system should use both types of remindings (Thagard et al., 1990; Gentner and Forbus, 1991; McDougal, Hammond and Seifert, 1991; Lauzon and Rose, 1993).

5.3.1 Similarity Assessment in CBRSs

Case-based reasoning systems use specific past experience in problem solving. When a new problem is posed, the system tries to find the most similar past problem (i.e., a case) and modify its solution so it fits the new problem. Similarity in CBR systems affects all aspects of such reasoning, namely:

- similarities of salient features of a new case to features predictive of past cases (this suggests relevant cases for retrieval);
- similarities of remaining case features to features predicted by a potentially relevant case (this confirms the case's relevance);
- dissimilarities involving features relevant to the solution of the known case guide adaptation of the solution to the new situation;
- dissimilarities which lead to problem solving failures (or non optimal solutions) trigger learning processes which result in case retention, representation refinement, and the acquisition of additional domain knowledge;
- varying similarity of two analogues affects complexity of analogical mapping.

Reviewing the literature dealing with similarity assessment, one notice that there are basically two main camps studying this problem: psychologists, and AI researchers. The first group tries to model human problem solving in its all deepness and to build systems psychologically valid. In contrast, artificial intelligence community attempts to build useful systems, even if they function differently from the way humans do.

The paper (Kibler and Aha, 1987) presents some initial baseline for evaluating the performance of complex similarity assessment strategies. The simplest systems are based on the assumptions that uniform featural description exists and that the exhaustive search is tractable. When given a new problem, the program compares features between a new problem and all stored cases. Based on matching features, the overall similarity score is computed.

More advanced approach is presented in (Kolodner, 1983, CYRUS), (Kolodner and Kolodner, 1987, SHRINK), and (Kolodner, 1987, JULIA). Here, the similarity assessment is combined with the classification process. All similar cases are stored in the hierarchy under the same classification. If, however, the retrieved case proves not to be useful, the system make use of secondary classification based on failures. When a case is retrieved via such a link, it is assumed to have greater similarity to the new case being processed because of its past involvement in a similar failure.

The MEDIATOR system (Simpson, 1985) works similarly. All possible classification paths are used and generally multiple cases are retrieved. Retrieved cases are ranked based on similarity and heuristic is used to prune this set further. First of all, cases for which the most important features are not identical (i.e., goals) are eliminated; remaining cases are ordered using the fixed preference scheme (background knowledge). MEDIATOR also uses a secondary classification scheme based on problem solving failures.

The aim of the HYPO system (Ashley and Rissland, 1988; Ashley, 1988) is different from systems presented so far. The system tries to retrieve all similar cases and they are ranked as being in support of an argument or in opposition to it.

In the case of CASEY (Koton, 1988a; Koton, 1988b) the causal model is used to infer featural equivalence and importance. The system abstracts from the actual symptoms (in medical diagnosis) to the underlying pathological states for matching.

Another approach is adopted in PROTOS (Bareiss, 1989a; Porter, Bareiss and R.C.Holte, 1989; Porter, Bareiss and R.C.Holte, 1990) where similarity of new and recalled case is assessed by explaining how their features provide an equivalent evidence for a classification. However, this approach requires a domain knowledge for explanation of featural equivalence. The overall similarity is based on heuristic evaluation of explanation quality and the importance of unmatched features. Because the explanation is highly dependent on a domain knowledge, the system has a single context similarity assessment.

In GREBE (Branting, 1988; Branting, 1991) an explanation is used to assess similarity of a new case (GREBE works in legal domain). A new case and precedent are believed to be similar to the degree to which explanations of key legal relationships in the precedent are applicable to the facts of the new case. Explanations are represented as cases and CBR is recursively invoked to map facts to legal relationships. Similarity assessment is simplified in the similar way as in PROTOS - single context similarity.

The most referenced systems that use both types of features for retrieval are (Thagard et al., 1990, ARCS) and (Gentner and Forbus, 1991, MAC/FAC). Both systems use two stage retrieval. The second stage is used for filtering out retrieved cases based on both surface and abstract features.

As a summary of presented systems, the following issues need to be stressed:

- Case retrieval and similarity are inter-related.
- Stored cases and problem solving situations are represented in terms of surface and abstract features. These features need to be acquired from domain expert or automatically from domain knowledge.
- Procedure to assess similarity relies on domain knowledge in addition to case features. Domain knowledge enables inferences of featural equivalence and evaluation of the importances of unmatched features in assessing similarity.

- Retrieved cases are ordered based on heuristic or numeric similarity functions. Usually, only featural similarities are used, systems are restricted only to a single problem-solving context.

Because there is a difference in understanding similarity among AI researchers, cognitive psychologists, and information retrieval experts an effort was made which tried to unify different views on similarity (King et al., 1988). Some of the most important issues are the following ones:

- Similarity of one situation to another is highly domain dependent. However, there is a sign that some general principles can be found.
- Similarity is used both to guide case storage and indexing, and as a conflict resolution method to refine or order the retrieved cases. It is arguable, whether the bulk of computational effort is concentrated in representing cases or in problem solving.
- If not all cases are uniformly represented there are two possibilities to resolve the similarity assessment: to use fixed context sensitive equivalence of features, or to resolve featural equivalence during problem solving. Because there is no predefined equivalence of features in the latter approach, this approach is more flexible and thus less domain dependent.
- Domain knowledge is useful for similarity assessment, however, it is not clear how to efficiently acquire such knowledge, what type of knowledge is most appropriate, how to represent it, and what amount is reasonable. The long-term goal is to build domain-independent systems.

5.3.2 Similarity Assessment in Other Areas

Similarity measures are not only important in CBR systems. In conceptual clustering similarity measure must be defined in order to cluster objects properly. If no similarity measure is used then we get an arbitrary clustering (domain structure). It has already been recognized in this area that defining a similarity measure is not an easy task. A straightforward approach is to compare the proportion between the number of attributes which share the value by two objects to the number of all attributes defined for those objects (Tversky, 1977). It was shown (Watanabe, 1985) that if we take this approach and we use all available primary (observed) and derived attributes, then any object is equally similar to any other ("Ugly Duckling Effect").

In database community, namely in spatial databases, the problem of similarity has also been studied (Jagadish, 1991). The author proposes an organization for a database of objects that permits an efficient retrieval of objects with a shape similar to an input shape. For similarity judgments an area-based similarity is used: two shapes are similar if the area where they do not match (error area) is smaller than error margin when one shape is placed on top of the other. The error margin is not constant and is used to control the amount of retrieved cases.

There is a relationship between CBR and the uncertainty reasoning. In the latter one, a certainty factor is used to estimate the quality of the result of the reasoning. In CBR a *similarity measure* is used and retrieved cases are ordered. A number of retrieved cases depends on a retrieval strategy and in some systems can be dynamically controlled by relaxing or constraining the initial query (Lauzon and Rose, 1993; Rissland et al., 1993) (this works as a similarity threshold).

As was mentioned in section 3, representation and retrieval are inter-related. Thus, different similarity assessment strategies require different representations. Analogical similarity is more efficient when hierarchical representation is used. Flat representation of set of features is sufficient for surface-feature similarity.

Many of the presented systems use explicit similarity assessment because they operate on weak domain theories. In a tractable and perfect domains the similarity can be computed if an appropriate knowledge representation scheme is used (Porter, 1989; Thagard and Holyoak, 1989; Lauzon and Rose, 1993; Rissland et al., 1993). The advantage of computed similarity is that the system can compute similarity while taking into consideration different contexts and different tasks.

6 Case Adaptation

There are only a few examples where the old solution can be reused without any change: either if the case base is sufficiently large, or if cases are represented with different granularities (fine-grain cases comprise a coarse-grain cases; thus, the solution can be composed using fine-grain cases). More often, one must modify either a solution or strategy to get a solution for the current problem. The process of modifying the old example to fit a new problem is called adaptation. At the beginning, the system has to decide what should be adapted in the ballpark solution. Later, the adaptation rules are applied to take the differences between the retrieved case and the input case into account. The main tasks in the adaptation process are as follows:

1. decide what needs to be adapted;
2. estimate if it is worthwhile to continue in adaptation;
3. specify how the inconsistencies between old solutions and the new needs can be diminished; and
4. perform adaptation according to step one, considering constraints set up by step two.

Most of the systems use domain- and task-specific adaptation rules. For each type of adaptation strategy, one has to designate the knowledge required for its application. Even straightforward application of adaptation strategies results in competent answers, but using them in novel ways ensure creative answers.

Based on diversity of CBR systems (planning, explanation, diagnosis, ...) one can differentiate adaptation strategies as follows:

Task differences put constraints on the time when the adaptation must take place.

- In the planning domain, adaptation usually occurs during run time since the evaluation of the plan is done during execution.
- In design domains, the evaluation has to take place in the design phase. Thus, adaptation will also occur in this phase.
- Adaptation of an explanation/diagnosis must be driven by an evaluation of how well the current explanation/diagnosis fits the existing set of constraints.

Knowledge differences specify the character and amount of domain knowledge required for adaptation.

- System uses deep understanding of the dependencies in a domain. An example is a system presented in (Goel and Chandrasekaran, 1989), where design modification techniques are guided by the internal dependency structure of the plan.

- System uses only surface level techniques to adapt cases. As an example, Kass's system (Kass, 1989) relies on external evaluation to ensure that no impossibilities are introduced to explanation during modification.

Domain independence characterizes the dependence of the system on a particular domain.

- The system uses only domain dependent rules for adaptation (Kass, 1989; Goel and Chandrasekaran, 1989; Alterman, 1988; Carbonell, 1986).
- The initial set of techniques is domain independent; later, some domain specific rules are incorporated through the explanation of expectation failures (Converse, Hammond and Marks, 1989).
- Modification techniques are domain independent (Hinrichs, 1989; Hanks and Weld, 1992).
- The system has a set of meta-goals that span across domains and above them specific transformations are constructed to optimize plans (Collins, 1989).

6.1 Adaptation Strategies

Based on the decision what needs to be adapted (see the adaptation algorithm, page 21), the adaptation algorithms can be categorized into the following categories:

- **Structural adaptation** applies the adaptation rules directly to the solution stored in a case. As an example, CHEF (Hammond, 1989a) modifies particular recipes, and JULIA (Kolodner, 1987) modifies prior criminal sentences. The CLAVIER system (Barletta, 1989) uses pieces of other cases to adapt the current case. This approach is more suitable than the rule-based approach since extremely large number of rules would be required to represent contextual information (in the case of CLAVIER, this information include material, shape, and position constraints of individual parts). Similar approach is also used as a basis in structure mapping analogical problem-solving (Gentner, 1983; Falkenhainer, Forbus and Gentner, 1989; Gentner and Forbus, 1991).
- **Derivational adaptation** uses rules that generated the original solution to generate the new solution by their re-application (Carbonell, 1983a; Veloso and Carbonell, 1993). MEDIATOR (Simpson, 1985) uses this approach and takes advantage of storing not only a solution with a case, but the planning sequence that contributed to the solution.

Even though both kinds of adaptation techniques are useful for CBRSs, the derivational adaptation has the following advantages over the structural adaptation:

1. it needs fewer ad hoc rules; and
2. it can be used across domains.

Even though the adaptation strategies are usually domain dependent, there are some common characteristics among them. The following approaches have been suggested:

- Use of a set of domain-dependent adaptation rules (Hammond, 1989a; Kass, 1991). In addition to domain dependency, the rules in CHEF (Hammond, 1989a) are usable only for local changes; they cannot be used to make global changes to the whole recipe. Difficulty in obtaining reliable adaptation rules, together with an impossibility of using these specific rules in other domains, is an obvious problem of domain-dependent adaptation. The adaptation in ABE (Kass, 1990) is based on tweaking strategies which are used to make small changes to explanation patterns. There are three strategies available: replace slot fillers with causal equivalent, generalize over-specific explanations, and elaborate incomplete explanations. Used strategies encode general knowledge about planning, not the knowledge about the specific planning domain. Thus, ABE is less domain-dependant than CHEF.

- Use of a domain model (Koton, 1988b; Jones, 1992). The model is used to adapt the retrieved diagnosis, taking into account differences in symptoms between retrieved and input cases (Koton, 1988b, CASEY). In the case of BRAINSTORMER (Jones, 1992), the adaptation converts generic knowledge into specific knowledge. The model-based adaptation suffers from the same problems as a set of an adaptation rules in CHEF.

- Use of pieces of existing cases (Hennessy and Hinkle, 1992; Gonzalez and Laureano-Ortiz, 1992; Tanaka, Hattori and Sueda, 1992). This is the most promising approach since it is the most general one and requires no additional domain knowledge other than cases. However, the problem of solution validation is more severe here, when compared to previous methods.

6.2 Adaptation Methods

In general, adaptation is the process of fixing up an old solution to meet the demands of the new situation. This can be achieved by: inserting something new into an old solution (put-in adaptation), deleting something from the solution (cut-off adaptation), and making a substitution in the retrieved solution. The following adaptation methods have been identified:

1. **Null adaptation** is the direct application of the retrieved solution to the new situation. The psychological evidence for this adaptation is in tasks where the solution itself is simple, even though the reasoning to a solution may be complex.

2. **Parametrized solutions**, a structural adaptation, is probably best understood. It is based on the comparison of the retrieved and input problem descriptions along the specified parameters. Parametrized adaptation is used in HYPO (Ashley, 1990), PERSUADER (Sycara, 1987), and PLEXUS (Alterman, 1986).

3. **Abstraction and respecialization** is a general structural adaptation technique that abstracts the piece of the retrieved solution, and respecializes it later. Respecialization results in applying other specializations of the abstractions to the current situation. Thus, it results in analogical problem solving. This technique is used in PLEXUS (Alterman, 1986) as a second step in adaptation process (first, a null adaptation is used, and only if something fails, the system replans it using abstraction and respecialization). A similar approach is used in PERSUADER (Sycara, 1987).

4. **Critic-based adaptation** is a structural adaptation based on using critics to debug almost correct solutions (Simmons, 1988; Hammond, 1989a; Gonzalez and Laureano-Ortiz, 1992). A critic checks if a particular combination of features can cause a problem in a plan. If such a feature is found, a specific repair strategy is applied for repair. In CHEF, several critic-based adaptation rules are used, e.g., deletion of unnecessary steps in a meal plan, insertion of missing steps to the plan, and separation of interfering steps. Similarly, in PERSUADER (Sycara, 1987) a critic checks if the goals of the various actors would be satisfied by the particular solution, e.g., if there is no interference among different goals.

5. **Reinstantiation** is a derivational adaptation method since it operates on the plan that was used to generate a particular solution. This imposes a limitation on reinstantiation, based on the problem solver's planning capabilities, because reinstantiation of a plan is planning. In MEDIATOR (Simpson, 1985) reinstantiation is used to generate resolutions between two disputants, which allows analogical reasoning.

There are domains, where problems are difficult to solve in one chunk while they are also difficult to decompose and solve separately because of the strong interaction between pieces. Here, the CBR paradigm provides the possibility to solve the problems at once and to adapt only those parts of the solution which do not fit the new situation (Hennessy and Hinkle, 1992). Solving the problem by adapting an old solution allows the problem solver to avoid dealing with many constraints, and keeps it from having to break the problem in to pieces which would require subsequent recomposition (Kolodner, 1992).

It is beneficial to keep track of near-miss cases and to use hypothetical cases for better adaptation process. In the HYPO system (Ashley, 1988; Rissland and Ashley, 1987b), this information helps the reasoner to focus on important features, which, if they were present, would yield a better solution. In the case of XBE (Pankakoski et al., 1991) hypothetical cases are theories formalized to the situation-means-ends forms (basic structure of a case in XBE). Hypothetical cases are incomplete; thus, they cannot be used in matching to the same extent as actual cases. However, they are used to introduce new design choices to case-based system and to integrate case-based and rule-based reasoning.

In the case of PERSUADER (Sycara, 1987) and CBA (Gonzalez and Laureano-Ortiz, 1992) the adaptation is performed using adaptation critics, which uses similarity metrics. This allows for meaningful adaptation but it is a domain-dependent approach.

Even though it is desirable to adapt an old experience to a current problem, sometimes it is either not possible to do so or retrieval of cases alone suffices for a particular task. This is the case for complex and ill-formed domains. In such cases, the system only presents relevant experience to the user as an advice and it is up to the user to adapt presented cases as necessary. Because of this fact, it is desirable that the case base is larger than in other systems (Kitano, Shibata and Shimazu, 1993). Such systems are referred to as co-operative case retrieval systems (repositories) since the task of storage and retrieval is done by machine and the adaptation and feedback is done by a human expert.

7 Relations of CBR to Other Areas in AI

7.1 Learning

A case-based reasoning system is a problem-solving system which inherently incorporates learning. Much of the learning is achieved in the following ways:

1. **Accumulation of new cases** - learning by remembering, knowledge acquisition, or rote learning. New cases give the reasoner more context for solving problems or evaluating situations. Learning can also help with estimation of which cases are worth storing and which should not be included in the case base. Learning algorithms can also help in converting an existing databases into case bases.

2. **Automatic case generation** by analogical reasoning, explanation-based learning, and by using genetic algorithms (construction of hypothetical cases).

3. **(Re)structuring the knowledge** in a case base based on learned knowledge. This involves classification and changes of hierarchy of knowledge base concepts. New classification allows for fine-tuning system's recall strategy, so it remembers cases at more appropriate times. Hierarchical changes affect system's deep knowledge.

4. **Updating the existing knowledge** in the converging way (reinforcement learning).

5. **Generalization** of already possessed knowledge. This involves creating more general concepts and case modification (construction of generic cases, rule generation, domain theories revision and development). When several cases are represented as similar to each other, and all predict the same solution, the reasoner can form a useful generalization. On the one hand, there are systems which use only instances of cases; on the other hand, there are systems which use both instances of cases and their generalizations. There are also systems (Boström and Idestam-Almquist, 1989) which use the generalization process as a way of evolving useful rules from cases. Rules could be used when they match cases exactly, while cases would be used when rules were not immediately applicable. In (Jones, 1992), a system which uses only abstract cases is presented (see also page 11).

Usually, the learning in CBRSs is opportunistic, which means that it is invoked only when some opportunities are recognized and it is not planned in advance. An example of such learning is failure-driven learning where goals and situations that are likely to cause problems in the future are identified (Hammond, 1989a; Hunter, 1989; Hammond et al., 1993).

7.2 Analogy

Analogical problem solving has been used in case-based reasoning systems in various ways. The CYRUS (Kolodner, 1983) is the first computer implementation of many of the themes expressed in Schank's work (Schank, 1982). It is a story understanding program which uses analogical problem solving if it cannot find direct answer to a query. Another example of analogical problem solving using a case-based approach is the SWALE system (Kass and Leake, 1988), a case-based creative explainer. When SWALE can not explain an anomalous event, it starts to look for other explanations in other contexts. The analogical reasoner in the planning system BRAINSTORMER (Jones, 1989) is similar to TWEAKER (a descendant of the SWALE system, which focuses on the tasks of explanation application and adaptation (Kass and Leake, 1988)); it is a failure-driven process and is invoked to answer requests from the planner.

Analogical problem solving has also been studied as a psychological model. In SME (Structure Mapping Engine, (Falkenhainer, Forbus and Gentner, 1989)) structural and semantic constraints (Gentner, 1983; Keane, Ledgeway and Duff, 1991) are used to generate all possible matches between two domains. Later, the resulting set is processed using structural criteria and the best match set is identified. In the ARCS system (Analog Retrieval by Constraint Satisfaction) (Thagard et al., 1990), analogs are retrieved using multiple constraints (the system retrieves superficially and analogically similar analogs). In the MAC/FAC system (Gentner and Forbus, 1991), a two stage retrieval is used as in ARCS. First, analogs are retrieved based on superficial similarities. Second, a retrieved set of analogs is refined based on structural similarities (second stage is similar to the SME approach).

Even though the previous work on analogy is reused to some extent in case-based systems, one can identify some differences in these two approaches:

- Case-based systems aim to find useful situations while analogical systems tends to find similar situations.

- Systems using analogy are usually inter-domain, whereas CBR systems are intra-domain.
- Most of CBR systems retrieve specific cases from a case base as opposed to generalized concepts or domain principles.
- An exact match is considered to be a perfect case in CBR; however, is not considered as an analogical relationship.
- Case-based systems usually pay less attention to the overall structural consistency of the retrieved case and the new problem as long as the new case is "sufficiently" similar in a way that it makes it useful to the process of solving a given problem. Models of analogy have typically been more concerned with overall systematicity.
- Case-based systems are pragmatic, problem-solving, and task oriented, whereas analogical systems have broader scope of reasoning in learning and generalization.

According to CBR research to date, the following can be concluded concerning analogy:

- According to psychological studies on analogy, both goal-based features and surface features are important for retrieval of related cases from a case base. The decision which of the two extreme approaches is more important is domain dependent. It was pointed out, however, that what is a surface feature in one domain can easily be a structural feature in another (Keane, 1988; Vosniadou and Ortony, 1989). An interesting issue is to examine which features are more important in a particular domain and to adapt the system accordingly. It is because the value of case features affect not only the retrieval stage of analogy, but also the transfer stage.
- When a similar case is identified in memory it could be unnecessarily complex for a given task, or it could be incomplete. So, it is suggested that a more sophisticated process of reconstruction should be used before retrieved information can be utilized in a transfer.
- Cases need to be "instructed" to be remembered, otherwise reminding does not occur. Thus, analogical reminding is not a deterministic procedure that can be evoked without strategic effort. It is also recognized that cognitive goals form a context within which retrieval operates, and the nature of this context determines what remindings occur.
- It was shown on various experimental tests that deeper understanding and encoding will aid analogical transfer (Keane, 1988; Vosniadou and Ortony, 1989). It is also believed that a great deal of inference is required to fully understand an example containing abstract relations as well as content features. Building an initial representation that contains both the abstract and content features is critical for any later analogical use based upon them.
- There is a problem that many times the system will not scale up properly for larger case bases. It is necessary that the system has representative cases, since the similarity and thus reminding is context sensitive. It should also be guaranteed that the search space of exemplars includes many kinds of overlapping knowledge and features. The next question is when to use cases to reason. Most of the systems (unless with hybrid representation) use case-based reasoning for all new problems. But psychological tests seem to support an idea that cases are not used so often and that problem solving relies on generalized principles most of the time.

- It is believed that CBR systems have the strong pragmatic, problem-solving task orientation, as opposed to the broader role of reasoning by analogy in learning and generalization. Many CBR systems to date rely on pre-existing generalizations, domain and system-specific procedures to determine what might be useful in a new case. However, it is also believed that CBR should be useful in other kinds of complex tasks that are not exclusively problem-solving. This brings in an interesting issue: systems usually use AI models which allow them to work only in a particular domain because the role of the process is implicitly encoded in the model, the memory base and representational format. The issues to be addressed by CBR systems are analogical transfer between domains and abstract reminding.

8 Evaluating Case-Based Systems

Only recently, some attention has been paid to evaluation of CBR systems (in some cases even the size of a case base is not reported). There are two main reasons why most of the systems are not sufficiently evaluated:

- It is hard to compare different systems since there is no underlying methodology; moreover, it is not always easy to convert a system from one domain to another. The usual solution is to compare the system to the performance of the human expert (Bareiss, 1988; Ashley, 1988; Gonzalez and Laureano-Ortiz, 1992; Tanaka, Hattori and Sueda, 1992; Leng, Buchanan and Nicholas, 1993), to the performance of the underlying system (Koton, 1988b; Golding and Rosenbloom, 1991; Hanks and Weld, 1992; Féret and Glasgow, 1993), to the performance of other systems (Bradtke and Lehnert, 1988a; Bradtke and Lehnert, 1988b; Bareiss, 1989b; Hanks and Weld, 1992; Cardie, 1993c; Krovvidy and Wee, 1993; Skalak, 1993), or to evaluate the system's performance before and after learning in novel environments (Ram and Santamaría, 1993). It should be noted that even in many of these cases the evaluation is only limited.

- Presented systems have only a small case base and thus the problem of scaling up is impossible to evaluate. Noted exception is the SQUAD system for software quality control (Kitano et al., 1992; Kitano, Shibata and Shimazu, 1993) with 25,000 cases, the ANAPRON system for the task of pronouncing names (Golding and Rosenbloom, 1991) with 5,000 cases, and a case-based system for knowledge acquisition (Cardie, 1993b) with 3,060 cases.[3]

Because CBR systems are studied as both psychological and computational systems, one should evaluate such systems accordingly. A psychological model should prove to be useful for explanation of psychological data, whereas computational system should prove to be useful in the real-world applications. A general guideline for evaluation of case-based reasoning systems is presented in (Koton, 1989; Cohen, 1989; O'Leary, 1993). The following issues are the most important ones:

- to present case-based problem-solving as an improvement over existing technologies, it is useful to compare a system's performance to the performance of an underlying system or to other systems, applied to the same domain;

- different systems can use different metrics for evaluating the method: (1) cognitive validity, (2) the amount of time required to solve the problem, (3) the quality of the solution;

[3]It should be noted that in some problem domains even small case bases suffice.

- evaluation should prove the system's ability to scale up;
- evaluation should present and discuss circumstances and assumptions under which the method works, when it fails to provide the solution, and what are its limitations;
- evaluation should characterize if the system relies on other methods and what is the dependency between the CBR system and the underlying method.

System evaluation can include not only performance measure issues but also verification and validation. Verification is performed to ensure consistency, completeness, and correctness of the system. Even though most of the systems rely on the ensured consistency (Kolodner and Riesbeck, 1986; Boström and Idestam-Almquist, 1989; Branting, 1992; Gonzalez and Laureano-Ortiz, 1992; Yang, Robertson and Lee, 1993) there are also systems which do not require consistency checking (Fertig and Gelertner, 1991; Kitano, Shibata and Shimazu, 1993). Validation ensures corectness of the final system with respect to the user needs and requirements. On the one hand, verification of the CBR system is virtually non-existent. On the other hand, validation is presented in some systems, namely:

- In PROTOS (Bareiss, 1988), the accuracy serve as a validation method. It is reported, that the PROTOS is correct 100% of the time (while COBWEB only 58% of the time and ID3 only 29% of the time).
- In HYPO (Ashley, 1990), the validation was performed by the comparative analysis of four (out of thirty) legal cases by a system.
- In CSI Battle Planner (Goodman, 1989), a set of 10% of the cases were put aside before classification. Later tests revealed 81.3% accuracy in predicting these situations.
- In PSSP (Protein Secondary Structure Prediction) system (Leng, Buchanan and Nicholas, 1993) authors used varying reference proteins number (1 to 55) and varying window size (20 to 24) to comparatively evaluate the system's performance. The best results were obtained with 55 reference proteins and the window size of 22. The accuracy on a new data set was 67.3% and predictive accuracy 69.3%. Moreover, if the class type of a protein was known, then secondary structure prediction achieved 90% accuracy. The system's performance is compared to the actual protein structures and several similarity matrices and different methods are compared as well.

9 Conclusions and Directions for Further Research

We have reviewed research in case-based reasoning and presented its connection to other areas in cognitive science. We have argued for the use of case-based reasoning systems by presenting several applications which gain by using cases to capture knowledge. During this process, we made two main observations:

1. Efforts in case representation and reasoning are driven by the requirements of specific applications. Similarity metric and classification strategy are usually predefined (hard-coded) and thus are not flexible. The case representation is strongly related to the application area, varying from simple structures to more complex hierarchies of cases. Consequently, retrieval and reasoning is also hard-coded into the system and is not (easily) transferable between various systems and adaptable to different domains. The learning strategies are in many

systems simple or are not implemented at all (except rote learning). As a result, there is no case-based system that provides adequate functionality for a wide range of applications.

2. The case base management in the current systems is not adequate for larger amount of cases. In many systems, this issue was ignored. Consequently, because only small number of cases were used, systems assumed that all cases are in memory. A better solution to this problem is to use more suitable representation and management tools to handle the complexity and needs of various applications even if the number of cases rises. Even though there are simple applications where small case bases are sufficient, there are also applications where a large number of cases is required.

This study presents different requirements and the drawbacks of existing systems in aim to propose further research in this area. What follows are some important topics for further research, including proposal of a general framework for case knowledge base management (CKBM).

Representation: up till now in many systems: (1) only surface features are represented in cases; (2) there is only one type of case in a particular application; (3) cases have flat organization; and (4) similarity assessment and classification strategy are often predefined and thus making contexts and tasks unchangeable which makes the system less flexible.

Therefore, more emphasis must be put towards characterizing useful contents of cases, such as context, task, and feedback information. Alternatively, more adaptive ways of similarity assessment and classification strategy need to be introduced. This should be based on the structure of case knowledge. Formalization of case representation is the goal of research as well. This will also allow for easier case knowledge interchange and case base reuse by various systems.

Reasoning: the requirements for individual components of case-based reasoning systems were identified in section 2. The most important requirements are the treatment of uncertainty and incompleteness of knowledge and the independence of application domain. This is required if the systems are to be useful in the real-world applications.

Therefore, the reasoning component should serve as a general purpose reasoning kernel, where application specific knowledge and methods can be implemented on top. Moreover, it is desirable that the system can draw not only inter-domain but intra-domain analogies as well.

An adaptation of previous cases to a new problem is a related issue. Most of the earlier systems used domain-specific adaptation rules to adapt solutions. Recent work is directed to build less domain-dependent systems; thus, adaptation rules need to be domain-independent. However, because of inherent complexity of this task, the systems are beeing converted from a stand-alone problem solvers to knowledge-based advice systems.

Learning: the most used learning method in current systems is rote learning (accumulation of cases). There are some exceptions where explanation-based learning or conceptual clustering is used for case classification and where some cases are generalized for future use. As was already mentioned the utility problem was mostly ignored; thus, all new experience was stored, regardless of the usability potential.

Current learning algorithms must be enhanced to work with proposed representation schemas and reasoning. It would also be beneficial to incorporate other learning strategies such as reinforcement learning and genetic algorithms. This would make the system more adaptive and allow for knowledge mining; thus, it will support knowledge evolution. Thanks to better evaluation during the learning process not all new cases will be stored. The decision will be made based on potential usability and cost of storing a new case.

Retrieval Optimization: retrieval optimization in case knowledge bases is almost an untouched problem. All current systems deal with it by providing classification strategy which allows retrieving relevant cases. However, these cases are relevant only to a predefined context (due to predefined indexing vocabularies and similarity measures). Moreover, accessibility is not sufficient if system performance is considered. Therefore, an open issue is to allow context and task specification for efficient retrieval both inter- and intra-domain.

Management: the database component is the heart of a large case knowledge base management system. It must efficiently integrate with representation and reasoning components to allow storing different types of cases.

The selection of the data models primitives and operations is a critical issue. Object oriented knowledge representation seems to be the right technology to use. This suggests considering Telos, a structural knowledge representation language, as a possible candidate. The current Telos framework can be extended for the needs of case knowledge bases.

Pursuing these issues allows for a definition of a **Case Knowledge Base Management System** (CKBMS). A CKBMS is a system which provides support for case representation, reasoning and management for a wide range of applications (see observation 1.), and make an efficient use of large amounts of cases (see observation 2.). A CKBMS consists of four components as identified above: a case representation component, a reasoning component, a learning component, and a knowledge management component. Even though the components are separated, they are strongly interconnected to support knowledge visualization and knowledge evolution - the main reasons for efficient CKBMSs.

References

Adelson, B., Burstein, M. H., Gentner, D., Hammond, K. J., Holyoak, K. J., and Thagard, P. R. (1988). The role of analogy in a theory of problem-solving. In *Proc. of the 10th Annual Conference of the Cognitive Science Society*, pages 298–304, Montreal, Quebec.

Aha, D. (1992). Tolerating noisy, irrelevant and novel attributes in instance- based learning algorithms. *International Journal of Man - Machine Studies*, 36(2):267–287.

Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

Alterman, R. (1986). An adaptive planner. In *Proc. of AAAI-86*, pages 65–69, Philadelphia, PA.

Alterman, R. (1988). Adaptive planning. *Cognitive Science*, 12:393–422.

Alterman, R. (1989). A concept space for reasoning about cases involving event structure. In *Proc. of the 2nd Workshop on Case-Based Reasoning*, pages 16–19, Pensacola Beach, FL.

Alterman, R. and Wentworth, M. (1989). Determining the important features of a case. In *Proc. of the 2nd Workshop on Case-Based Reasoning*, pages 193–196, Pensacola Beach, FL.

Anthony G. Francis, J. and Ram, A. (1993). Computational models of the utility problem and their application to a utility analysis of case-based reasoning. In *Proc. of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC.

Ashley, K. D. (1988). *Modelling legal argument: Reasoning with cases and hypotheticals.* PhD thesis, Department of Computer and Information Science University of Massachusetts, Amherst. Also available as a COINS Technical Report No. 88–01.

Ashley, K. D. (1990). *Modeling Legal Argument: Reasoning with Case and Hypotheticals.* MIT Press/Bradford Books, Cambridge, MA.

Ashley, K. D. and Rissland, E. L. (1988). Waiting on weighting: A symbolic least commitment approach. In *Proc. of AAAI-88*, pages 239–244, St. Paul, MN.

Bailey, D., Thompson, D., and Feinstein, J. (1987). Similarity networks as a knowledge representation for space application. In *Proc. of 3rd Conference on Artificial Intelligence for Space Application*, pages 279–284, Huntsville, AL.

Bareiss, E. R. (1988). *A unified approach to concept representation, classification, and learning.* PhD thesis, Department of Computer Science, University of Texas. Also published as a Technical Report CS 88-10, Dept. of Computer Science, Vanderbilt University, Nashville, TN.

Bareiss, E. R. (1989a). *Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning.* Academic Press, Boston, MA.

Bareiss, E. R. (1989b). The experimental evaluation of a case-based learning apprentice. In *Proc. of the 2nd Workshop on Case-Based Reasoning*, pages 162–167, Pensacola Beach, FL.

Barletta, R. (1989). Case adaptation in autoclave layout design. In *Proc. of the 2nd Workshop on Case-Based Reasoning*, pages 203–207, Pensacola Beach, FL.

Barletta, R. and Hennessy, D. (1989). Case adaptation in autoclave layout design. In *Proc. of the 2nd Workshop on Case-Based Reasoning*, Pensacola Beach, FL.

Barletta, R. and Mark, W. (1988). Explanation-based indexing of cases. In *Proc. of AAAI-88*, pages 541–546, St. Paul, MN.

Berger, J. (1993). Knowledge acquisition and design support in a medical domain. In *Proc. of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC.

Berman, D. H. and Hafner, C. D. (1993). Representing teleological structure in case-based legal reasoning: The missing link. In *Proc. of the 4rd International Conference on Artificial Intelligence and Law*, Amsterdam, The Netherlands.

Bhansali, S. and Harandi, M. (1993). Synthesis of UNIX programs using derivational analogy. *Machine Learning*, 10(1):7–55.

Birnbaum, L. and Collins, G. C. (1988). The transfer of experience across planning domains through the acquisition of abstract strategies. In *Proc. of the 1st Workshop on Case-Based Reasoning*, pages 61–79, Clearwater Beach, FL.

Birnbaum, L. and Collins, G. C. (1989). Remindings and engineering design themes: A case study in indexing vocabulary. In *Proc. of the 2nd Workshop on Case-Based Reasoning*, pages 47–51, Pensacola Beach, FL.

Birnbaum, L., Collins, G. C., Freed, M., and Krulwich, B. (1990). Model-based diagnosis of planning failures. In *Proc. of AAAI-90*, pages 318–323, Boston, MA.

Boström, H. and Idestam-Almquist, P. (1989). APPRENTICE88 - A case-based refiner for heuristic classification systems. Technical Report 157, Department of Computer and Systems Sciences, University of Stokholm, Stokholm, Sweden.

Bradtke, S. and Lehnert, W. G. (1988a). Some experiments with case-based search. In *Proc. of AAAI-88*, pages 80–93, St. Paul, MN.

Bradtke, S. and Lehnert, W. G. (1988b). Some experiments with case-based search. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 80–93, Clearwater Beach, FL.

Brandau, R., Lemmon, A., and Lafond, C. (1991). Experience with extended episodes: Cases with complex temporal structure. In *Proc. of the 4 th Workshop on Case-Based Reasoning*, pages 1–12, Washington, D.C.

Branting, L. K. (1988). The role of explanation in reasoning from legal precedent. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 94–103, Clearwater Beach, FL.

Branting, L. K. (1989). Integrating generalizations with exemplar-based reasoning. In *Proc. of the 11th Annual Conference of the Cognitive Science Society*, pages 139–146, Ann Arbor, MI.

Branting, L. K. (1991). *Integrating rules and precedents for classification and explanation: Automating legal analysis.* PhD thesis, University of Texas, Austin.

Branting, L. K. (1992). A case-based approach to problem formulation. In *Proc. of the 14th Annual Conference of the Cognitive Science Society*, pages 726–731, Bloomington, IN.

Broos, P. and Branting, L. K. (1993). Compositional instance-based acquisition of preference predicates. In *Proc. of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC.

Buchanan, B. G. and Wilkins, D. C. (1993). *Readings in Knowledge Acquisition and Learning.* Morgan Kaufmann Publishers, San Mateo, CA.

Burstein, M. H. (1986). Concept formation by incremental analogical reasoning and debugging. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2, Los Altos, CA. Morgan Kaufmann.

Burstein, M. H. (1989). Analogy vs. CBR: The purpose of mapping. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 133–136, Pensacola Beach, FL.

Carbonell, J. G. (1981). A computational model of analogical problem solving. In *Proc. of the 7 th IJCAI*, pages 147–152, Vancouver, Canada.

Carbonell, J. G. (1983a). Derivational analogy and its role in problem solving. In *Proc. of AAAI-83*, pages 64–69, Washington, DC.

Carbonell, J. G. (1983b). Derivational analogy in problem solving and knowledge acquisition. In *Proc. of the 2nd International Workshop on Machine Learning*, Monticello, IL.

Carbonell, J. G. (1986). Learning by analogy: Formulating and generalizing plans from past experience. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, volume 2.

Carbonell, J. G. and Veloso, M. M. (1988). Integrating derivational analogy into a general problem solving architecture. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 104–123, Clearwater Beach, FL.

Cardie, C. (1993a). A case-based approach to knowledge acquisition for domain-specific sentence analysis. In *Proc. of AAAI-93*, Washington, DC.

Cardie, C. (1993b). A case-based approach to knowledge acquisition for domain-specific sentence analysis. Personal communication.

Cardie, C. (1993c). Using decision trees to improve case-based learning. In *Proc. of the 10 th International Conference on Machine Learning*, Amherst, MA.

Cohen, P. R. (1989). Evaluation and case-based reasoning. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 168–172, Pensacola Beach, FL.

Collins, G. C. (1989). Plan adaptation: A transformational approach. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 90–93, Pensacola Beach, FL.

Collins, G. C. and Birnbaum, L. (1990). Problem-solver state descriptions as abstract indices for case retrieval. In *AAAI Spring Symposium on Case-Based Reasoning*, pages 32–35.

Converse, T., Hammond, K. J., and Marks, M. (1989). Learning modification rules from expectation failure. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 110–114, Pensacola Beach, FL.

Cook, D. J. (1991). The base selection task in analogical planning. In *Proc. of the 12 th IJCAI*, pages 790–795, Sidney, Australia.

DeJong, G. F. and Mooney, R. J. (1986). Explanation-based learning: An alternative view. *Machine Learning*, 1(2):145–176.

Deraedt, L. and Bruynooghe, M. (1992). Interactive concept-learning and constructive induction by analogy. *Machine Learning*, 8(2):107–150.

Dierbach, C. and Chester, D. L. (1991). A formal basis for analogical reasoning. In *Proc. of 2rd International Conference on Principles of Knowledge Representation and Reasoning.*

Domeshek, E. A. (1992). *Do the right thing: A component theory for indexing stories as social advice.* PhD thesis, Yale University. Also available as a Technical Report 92-26 from the Northwestern University.

Falkenhainer, B., Forbus, K. D., and Gentner, D. (1989). Structure-maping engine. *Artificial Intelligence*, 41:1–63.

Féret, M. P. and Glasgow, J. I. (1993). Hybrid case-based reasoning for the diagnosis of complex devices. In *Proc. of AAAI-93*, Washington, DC.

Fertig, S. and Gelertner, D. H. (1991). FGP: A virtual machine for acquiring knowledge from cases. In *Proc. of the 12 th IJCAI*, pages 796–802, Sidney, Australia.

Fisher, D. (1987). Knowledge acquisition via incremental concept clustering. *Machine Learning*, 2(1):39–172.

Gaschnig, J. (1979). A problem similarity approach to devising heuristics: First results. In *Proc. of AAAI-79*, pages 301–307.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.

Gentner, D. (1989). The mechanisms of analogical learning. In Vosniadou, S. and Ortony, A., editors, *Similarity and Analogical Reasoning*, pages 199–241, New York, NY. Cambridge University Press.

Gentner, D. and Forbus, K. D. (1991). MAC/FAC: A model of similarity-based retrieval. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 504–509, Chicago, IL.

Gick, M. and Holyoak, K. J. (1980). Analogical problem solving. *Cognitive Psychology*, 12:306–355.

Goel, A. K. (1989). *Integration of case-based reasoning and model-based reasoning for adaptive design problem solving.* PhD thesis, Department of Computer and Information Science. The Ohio State University.

Goel, A. K. and Chandrasekaran, B. (1989). Use of device models in adaptation of design cases. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 100–109, Pensacola Beach, FL.

Golding, A. R. and Rosenbloom, P. S. (1991). Improving rule-based systems through case-based reasoning. In *Proc. of AAAI-91*, pages 22–27, Anaheim, CA.

Gonzalez, A. J. and Laureano-Ortiz, R. (1992). A case-based reasoning approach to real estate property appraisal. *Expert Systems with Applications*, 4:229–246.

Goodman, M. (1989). CBR in battle planning. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 264–269, Pensacola Beach, FL.

Hammond, K. J. (1989a). *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, Boston, MA.

Hammond, K. J. (1989b). Opportunistic memory. In *Proc. of the 11 th IJCAI*, Detroit, MI.

Hammond, K. J., Converse, T., Marks, M., and Seifert, C. M. (1993). Opportunism and learning. *Machine Learning*, 10(3):279–309.

Hanks, S. and Weld, D. S. (1992). The systematic plan adaptor: A formal foundation for case-based planning. Technical Report 92-09-04, University of Washington, Washington, DC. Submitted to CACM.

Heckerman, D. (1991). Similarity networks for the construction of multiple-fault belief networks. *Uncertainty in Artificial Intelligence 6*, 12:51–64.

Hennessy, D. and Hinkle, D. (1991). Initial results from Clavier: A case-based autoclave loading assistant. In *Proc. of the 4 th Workshop on Case-Based Reasoning*, pages 225–232, Washington, D.C.

Hennessy, D. and Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert*, 7:21–26.

Hinrichs, T. R. (1988). Towards an architecture for open world problem solving. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 182–189, Clearwater Beach, FL.

Hinrichs, T. R. (1989). Strategies for adaptation and recovery in a design problem solver. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 115–118, Pensacola Beach, FL.

Holyoak, K. J. (1985). The pragmatics of analogical transfer. In Bower, G., editor, *The Psychology of Learning and Motivation*, New York, NY. Academic Press.

Hunter, L. (1989). Finding paradigm cases, or when is a case worth remembering? In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 57–61, Pensacola Beach, FL.

Indurkhya, B. (1990). On the role of interpretive analogy in learning. *New Generation Computing*, 8(4):385–402.

Jagadish, H. (1991). A retrieval technique for similar shapes. In *Proc. of the 10th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 208–217, Denver, CO.

Jones, E. K. (1989). Case-based analogical reasoning using proverbs. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 275–279, Pensacola Beach, FL.

Jones, E. K. (1991). Adapting abstract knowledge. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 155–160, Chicago, IL.

Jones, E. K. (1992). Model-based case adaptation. In *Proc. of AAAI-92*, pages 673–678, San Jose, CA.

Kambhampati, S. (1989). Representational requirements for plan reuse. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 20–23, Pensacola Beach, FL.

Kass, A. (1990). *Developing creative hypotheses by adapting explanations.* PhD thesis, Yale University. Also available as an ILS technical report No. 6, Northwestern University.

Kass, A. M. (1989). Strategies for adapting explanations. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 119–123, Pensacola Beach, FL.

Kass, A. M. (1991). Adaptation strategies for case-based plan recognition. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 161–166, Chicago, IL.

Kass, A. M. and Leake, D. B. (1988). Case-based reasoning applied to constructing explanations. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 190–208, Clearwater Beach, FL.

Keane, M. T. (1988). *Analogical Problem Solving.* Ellis Horwood, Chichester, UK.

Keane, M. T., Ledgeway, T., and Duff, S. (1991). Constraint on analogical mapping: The effects of similarity & order. In *Proc. of the 13th Annual Conference of the Cognitive Science Society*, pages 275–280, Chicago, IL.

Kibler, D. and Aha, D. (1987). Learning representative exemplars of concepts: An initial case study. In *Proc. of the 4th International Workshop on Machine Learning*, pages 24–30, Irvine, CA.

King, J., Klein, G., Whitaker, L., and Wiggins, S. (1988). SURVER III: An application of case-based reasoning. In *Proceedings of the 4th Annual Aerospace Applications of AI Conference*, Dayton, OH.

Kitano, H., Shibata, A., and Shimazu, H. (1993). Case-method: A methodology for building large-scale case-based systems. In *Proc. of AAAI-93*, Washington, DC.

Kitano, H., Shibata, A., Shimazu, H., Kajihara, J., and Sato, A. (1992). Building large-scale and corporate-wide case-based systems: Integration of organizational and machine executable algorithms. In *Proc. of AAAI-92*, pages 843–849, San Jose, CA.

Kolodner, J. L. (1983). Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7(4):281–328.

Kolodner, J. L. (1987). Capitalizing on failure through case-based inference. In *Proc. of the 9th Annual Conference of the Cognitive Science Society*, pages 715–726, Seatle, WA.

Kolodner, J. L. (1989). Selecting the best case for a case-based reasoner. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 77–81, Pensacola Beach, FL.

Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34.

Kolodner, J. L. and Kolodner, R. (1987). Using experience in clinical problem solving: Introduction and framework. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):420–431. Also published as a TR GIT-ICS-85/21.

Kolodner, J. L. and Riesbeck, C. K. (1986). *Experience, Memory, and Reasoning.* Lawrence Erlbaum Assoc., Inc., Hillsdale, NJ.

Kolodner, J. L. and Wills, L. W. (1993). Paying attention to the right thing: Issues of focus in case-based creative design. In *Proc. of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC.

Koton, P. (1988a). Reasoning about evidence in causal explanation. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 260–270, Clearwater Beach, FL.

Koton, P. (1988b). *Using experience in learning and problem solving.* PhD thesis, Computer Science Department, MIT.

Koton, P. (1989). Evaluating case-based problem solving. In *Proc. of the 2^{nd} Workshop on Case-Based Reasoning*, pages 173–175, Pensacola Beach, FL.

Koton, P. and Chase, M. P. (1989). Knowledge representation in a case-based reasoning system: Defaults and exceptions. In *Proc. of 1^{st} International Conference on Principles of Knowledge Representation and Reasoning*, pages 203–211, Toronto, Ontario.

Krovvidy, S. and Wee, W. G. (1993). Wastewater treatment systems from case-based reasoning. *Machine Learning*, 10(3):341–363.

Lauzon, D. and Rose, T. (1993). Task-oriented and similarity-based retrieval. In *Proc. of the 12^{th} International Conference on Entity-Relationship Approach*, Dallas, TX. Submitted.

Lebowitz, M. (1986). Not the path to perdition: The utility of similarity-based learning. In *Proc. of AAAI-86*, pages 533–537, Philadelphia, PA.

Lebowitz, M. (1987). Experiments with incremental concept formation. *Machine Learning*, 2(1):103–138.

Leng, B., Buchanan, B. G., and Nicholas, H. B. (1993). Protein secondary structure prediction using two-level case-based reasoning. In Searls, D. and Shavlik, J., editors, *Proc. of the 1^{st} International Conference on Intelligent Systems for Molecular Biology*, Washington, DC.

Louis, S., McGraw, G., and Wyckoff, R. O. (1992). CBR assisted explanation of GA results. Technical Report 361, Department of Computer Science, Indiana University, Bloomington, IN. Also in the Journal of Theoretical and Experimental Artificial Intelligence, 5, 1993, pages 21-37.

McCarthy, J. (1958). Programs with common sense. In *Proc. of the Symposium on the Mechanisation of Thought Precesses*, pages 77–84, National Physical Laboratory.

McDougal, T. F., Hammond, K. J., and Seifert, C. M. (1991). A functional perspective on reminding. In *Proc. of the 13^{th} Annual Conference of the Cognitive Science Society*, pages 510–515, Chicago, IL.

Minton, S. N. (1988). *Learning search control knowledge: An explanation-based approach.* Kluwer Academic Publishers, Hingham, MA.

Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M. (1990). Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems*, 8(4):325–362.

Navinchandra, D., Sycara, K. P., and Narasimhan, S. (1991). Behavioral synthesis in CADET, a case-based design tool. In *Proc. of the 7^{th} IEEE Computer Society Conference on Artificial Intelligence Applications*, pages 217–221, Miami, CA.

O'Leary, D. E. (1993). Verification and validation of case-based systems. *Expert Systems with Applications*, 6(1):57–66.

Owens, C. (1989). Plan transformations as abstract indices. In *Proc. of the 2^{nd} Workshop on Case-Based Reasoning*, pages 62–65, Pensacola Beach, FL.

Owens, C. (1993). Integrating feature extraction and memory search. *Machine Learning*, 10(3):311–339.

Pankakoski, J., Eloranta, E., Luhtala, M., and Nikkola, J. (1991). Applying case-based reasoning to manufacturing systems design. *Computer Integrated Manufacturing Systems*, 4(4):212–220.

Pearce, M., Goel, A. K., Kolodner, J., Zimring, C., Santosa, L., and Billington, R. (1992). Case-based design support - A case study in architectural design. *IEEE Expert*, 7(5):14–20.

Porter, B. W. (1989). Similarity assessment: Computation *vs.* representation. In *Proc. of the 2nd Workshop on Case-Based Reasoning*, pages 82–84, Pensacola Beach, FL.

Porter, B. W., Bareiss, E., and R.C.Holte (1989). Knowledge acquisition and heuristic classification in weak-theory domains. Technical Report AI89-96, Artificial Intelligence Laboratory, University of Texas, Austin.

Porter, B. W., Bareiss, E., and R.C.Holte (1990). Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45:229–263.

Porter, B. W., Lester, J. C., Murray, K., Pittman, K. E., Souther, A., Acker, L., and Jones, T. (1988). AI research in the context of multifunctional knowledge base: The botany knowledge base project. Technical Report AI-TR-88-88, Department of Computer Sciences, University of Texas, Austin, TX.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Ram, A. (1993a). AQUA: Questions that drive the explanation process. In *Inside Computer Explanation*. Lawrence Erlbaum. Also available as the Technical Report GIT-CC-93/47.

Ram, A. (1993b). Indexing, elaboration and refinement: Incremental learning of explanatory cases. *Machine Learning*, 10(3):201–248.

Ram, A. and Santamaría, J. C. (1993). A multistrategy case-based and reinforcement learning approach to self-improving reactive control systems for autonomous robotic navigation. In *Proc. of the 2nd International Workshop on Machine Learning*, Harpers Ferry, WV.

Rissland, E. L. and Ashley, K. D. (1986). Hypotheticals as heuristic device. In *Proc. of AAAI-86*, pages 289–297, Philadelphia, PA.

Rissland, E. L. and Ashley, K. D. (1987a). A case-based system for trade secrets law. In *Proc. of the 1st International Conference on Artificial Intelligence and Law.*

Rissland, E. L. and Ashley, K. D. (1987b). HYPO: A case-based reasoning system. In *Proc. of the 10th IJCAI*, Milan, Italy.

Rissland, E. L., Daniels, J. J., Rubinstein, Z. B., and Skalak, D. B. (1993). Case-based diagnostic analysis in a blackboard architecture. In *Proc. of AAAI-93*, Washington, DC.

Ross, B. H. (1989). Remindings in learning: Objects and tools. In Vosniadou, S. and Ortony, A., editors, *Similarity and Analogical Reasoning*, pages 438–469, New York, NY. Cambridge University Press.

Rubin, S. H. (1992). Case-based learning - A new paradigm for automated knowledge acquisition. *ISA Transactions*, 31(2):181–209.

Schank, R. C. (1982). *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge University Press, New York, NY.

Schank, R. C. (1986). *Explanation Patterns: Understanding Mechanically and Creatively.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Schank, R. C., Brand, M., Burke, R., Domeshek, E., Edelson, D. J., Ferguson, W., Freed, M., Jona, M., Krulwich, B., Ohmaye, E., Osgood, R., and Pryor, L. (1990). Towards a general content theory of indices. In *AAAI Spring Symposium on Case-Based Reasoning*, pages 36–40.

Scott, P. and Sage, K. (1992). Why generalize? Hybrid representations and instance-based learning. In *Proc. of the 10 th ECAI*, pages 484–486, Vienna, Austria.

Shinn, H. S. (1988). Abstractional analogy: A model of analogical reasoning. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 370–387, Clearwater Beach, FL.

Simmons, R. G. (1988). A theory of debugging. In *Proc. of the 1 st Workshop on Case-Based Reasoning*, pages 388–401.

Simon, H. (1983). Why should machines learn? In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine learning. An artificial intelligence approach*, volume 1, pages 25–38, San Mateo, CA. Morgan Kaufmann.

Simoudis, E. (1992). Using case-based retrieval for customer technical support. *IEEE Expert*, 7(5):7–12.

Simoudis, E. and Miller, J. (1990). Validated retrieval in case-based reasoning. In *Proc. of AAAI-90*, pages 310–315, Boston, MA.

Simpson, R. (1985). *A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation.* PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA. Also published as a Technical Report GIT-ICS-85/18.

Skalak, D. B. (1993). Using genetic algorithm to learn prototypes for case retreival and classification. In *Proc. of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC.

Smyth, B. and Cunningham, P. (1992). Déjà Vu: A hierarchical case-based reasoning system for software design. In *Proc. of the 10 th ECAI*, pages 587–589, Vienna, Austria.

Stanfill, C. and Waltz, D. L. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228.

Sutton, R., editor (1992). *Machine Learning*, volume 8. A special issue on reinforcement learning.

Suzuki, H., Ohnishi, H., and Shigemasu, K. (1992). Goal-directed processes in similarity judgement. In *Proc. of the 14th Annual Conference of the Cognitive Science Society*, pages 343–348, Bloomington, IN.

Sycara, K. P. (1987). *Resolving adversarial conflicts: An approach to integrating case-based and analytic methods.* PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA. Also published as a Technical Report No. GIT-ICS-87/26.

Tanaka, T., Hattori, M., and Sueda, N. (1992). Use of multiple cases in case-based design. In *Proc. of the 8th IEEE Computer Society Conference on Artificial Intelligence Applications*, pages 233–239, Monterey, CA.

Thagard, P. R. and Holyoak, K. J. (1989). Why indexing is the wrong way to think about analog retrieval. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 36–40, Pensacola Beach, FL.

Thagard, P. R., Holyoak, K. J., Nelson, G., and Gotchfeld, D. (1990). Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46:259–310.

Thompson, K. and Langley, P. (1989). Organization and retrieval of composite concepts. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 329–333, Pensacola Beach, FL.

Turner, R. M. (1989). *A schema-based model of adaptive problem solving.* PhD thesis, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA. Also published as Technical Report GIT-ICS-89/42.

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

Veloso, M. M. and Carbonell, J. G. (1989). Learning analogies by analogy - The closed loop of memory organizationand problem solving. In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 153–158, Pensacola Beach, FL.

Veloso, M. M. and Carbonell, J. G. (1993). Derivational analogy in PRODIGY: Automating case acquisition, storage,and utilization. *Machine Learning*, 10(3):249–278.

Vosniadou, S. and Ortony, A. (1989). *Similarity and Analogical Reasoning.* Cambridge University Press, New York, NY.

Waltz, D. (1989). Is indexing used for retrieval? In *Proc. of the 2 nd Workshop on Case-Based Reasoning*, pages 41–44, Pensacola Beach, FL.

Watanabe, S. (1985). *Pattern Recognition. Human and Mechanical.* John Wiley, New York.

Yang, S.-A., Robertson, D., and Lee, J. (1993). KICS: A knowledge-intensive case-based reasoning system for building regulations and case histories. In *Proc. of the 4rd International Conference on Artificial Intelligence and Law*, Amsterdam, The Netherlands.

Zito-Wolf, R. (1993). Case-based organization using multicases. In *Proc. of the AAAI-93 Workshop on Case-Based Reasoning*, Washington, DC.