A. Aamodt, E. Plaza (1994); Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications. IOS Press, Vol. 7: 1, pp. 39-59.

Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches

Agnar Aamodt
University of Trondheim,
College of Arts and Science,
Department of Informatics,
N-7055 Dragvoll, Norway.
Phone: +47 73 591838;
fax: +47 73 591733;
e-mail: agnar@ifi.unit.no

Enric Plaza
Institut d'Investigació en Intel·ligència Artificial, CSIC,
Camí de Santa Bàrbara,
17300 Blanes, Catalonia, Spain.
Phone: +34 72 336101;
fax: +34 72 337806;
e-mail: plaza@ceab.es

Abstract

Case-based reasoning is a recent approach to problem solving and learning that has got a lot of attention over the last few years. Originating in the US, the basic idea and underlying theories have spread to other continents, and we are now within a period of highly active research in case-based reasoning in Europe, as well. This paper gives an overview of the foundational issues related to case-based reasoning, describes some of the leading methodological approaches within the field, and exemplifies the current state through pointers to some systems. Initially, a general framework is defined, to which the subsequent descriptions and discussions will refer. The framework is influenced by recent methodologies for knowledge level descriptions of intelligent systems. The methods for case retrieval, reuse, solution testing, and learning are summarized, and their actual realization is discussed in the light of a few example systems that represent different CBR approaches. We also discuss the role of case-based methods as one type of reasoning and learning method within an integrated system architecture.

1. Introduction

Over the last few years, case-based reasoning (CBR) has grown from a rather specific and isolated research area to a field of widespread interest. Activities are rapidly growing - as seen by the increased rate of research papers, availability of commercial products, and also reports on applications in regular use. In Europe, researchers and application developers recently met at the First European Workshop on Case-Based Reasoning, which took place in Germany, November 1993. It gathered around 120 people and more than 80 papers on scientific and application-oriented research were presented.

1.1. Background and motivation.

Case-based reasoning is a problem solving paradigm that in many respects is fundamentally different from other major AI approaches. Instead of relying solely on general knowledge of a problem domain, or making associations along generalized relationships between problem descriptors and conclusions, CBR is able to utilize the *specific* knowledge of previously experienced, concrete problem situations (cases). A new problem is solved by finding a similar past case, and reusing it in the new problem situation. A second important difference is that CBR also is an approach to incremental, sustained learning, since a new experience is retained each time a problem has been solved, making it immediately available for future problems. The CBR field has grown rapidly over the last few years, as seen by its increased share of papers at major conferences, available commercial tools, and successful applications in daily use.

This paper presents an overview of the field, in terms of its underlying foundation, its current state-of-the-art, and future trends. The description of CBR principles, methods, and systems is made within a general analytic scheme. Other authors have recently given overviews of case-based reasoning (Ch. 1 in [Riesbeck-89], Introductory section of [DARPA-89], [Slade-91], [Kolodner-92]). Our overview differs in four major ways from these accounts: First, we initially specify a general descriptive framework to which the subsequent method descriptions will refer. Second, we put a strong emphasis on the methodological issues of case-based reasoning, and less on a discussion of suitable application types and on the advantages of CBR over rule-based systems. (This has been taken very well care of in the documents cited above). Third, we strive to maintain a neutral view of existing CBR approaches, unbiased by a particular 'school'[1]. And finally, we include results from the European CBR arena, which unfortunately have been missing in American CBR reports.

What is case-based reasoning? Basically: To solve a new problem by remembering a previous similar situation and by reusing information and knowledge of that situation. Let us illustrate this by looking at some typical problem solving situations:

- A physician - after having examined a particular patient in his office - gets a reminding to a patient that he treated two weeks ago. Assuming that the reminding was caused by a similarity of important symptoms (and not the patient's hair-color, say), the physician uses the diagnosis and treatment of the previous patient to determine the disease and treatment for the patient in front of him.
- A drilling engineer, who have experienced two dramatic blow out situations, is quickly reminded of one of these situations (or both) when the combination of critical measurements matches those of a blow out case. In particular, he may get a reminding to a mistake he made during a previous blow-out, and use this to avoid repeating the error once again.
- A financial consultant working on a difficult credit decision task, uses a reminding to a previous case, which involved a company in similar trouble as the current one, to recommend that the loan application should be refused.

1.2. Case-based problem solving.

As the above examples indicate, reasoning by re-using past cases is a powerful and frequently applied way to solve problems for humans. This claim is also supported by results from cognitive psychological research. Part of the foundation for the case-based approach, is its psychological plausibility. Several studies have given empirical evidence for the dominating role of specific, previously experienced situations (what we call cases) in human problem solving (e.g. [Ross-89]). Schank [Schank-82] developed a theory of learning and reminding based on retaining of experience

[1]Our own experience from active CBR research over the last 5 years started out from different backgrounds and motivations, and we may have developed different views to some of the major issues involved. We will give examples of our respective priorities and concerns related to CBR research as part of the discussion about future trends towards the end of the paper.

in a dynamic, evolving memory[2] structure. Anderson [Anderson-83] has shown that people use past cases as models when learning to solve problems, particularly in early learning. Other results (e.g. by W.B. Rouse [Kolodner-85]) indicate that the use of past cases is a predominant problem solving method among experts as well. Studies of problem solving by analogy (e.g. [Gentner-83, Carbonell-86]) also shows the frequent use of past experience in solving new and different problems. Case-based reasoning and analogy are sometimes used as synonyms (e.g. by Carbonell). Case-based reasoning can be considered a form of *intra-domain analogy*. However, as will be discussed later, the main body of analogical research [Kedar-Cabelli-86, Hall-89, Burstein-89] have a different focus, namely analogies across domains.

In CBR terminology, a *case* usually denotes a *problem situation*. A previously experienced situation, which has been captured and learned in a way that it can be reused in the solving of future problems, is referred to as a past case, previous case, stored case, or retained case. Correspondingly, a new case or unsolved case is the description of a new problem to be solved. Case-based reasoning is - in effect - a cyclic and integrated process of solving a problem, learning from this experience, solving a new problem, etc.

Note that the term problem solving is used here in a wide sense, coherent with common practice within the area of knowledge-based systems in general. This means that problem solving is not necessarily the finding of a concrete solution to an application problem, it may be any problem put forth by the user. For example, to justify or criticize a solution proposed by the user, to interpret a problem situation, to generate a set of possible solutions, or generate expectations in observable data are also problem solving situations.

1.3. Learning in Case-based Reasoning.

A very important feature of case-based reasoning is its coupling to learning. The driving force behind case-based methods has to a large extent come from the machine learning community, and case-based reasoning is also regarded a subfield of machine learning[3]. Thus, the notion of case-based reasoning does not only denote a particular reasoning method, irrespective of how the cases are acquired, it also denotes a machine learning paradigm that enables sustained learning by updating the case base after a problem has been solved. Learning in CBR occurs as a natural by-product of problem solving. When a problem is successfully solved, the experience is retained in order to solve similar problems in the future. When an attempt to solve a problem fails, the reason for the failure is identified and remembered in order to avoid the same mistake in the future.

Case-based reasoning favours learning from experience, since it is usually easier to learn by retaining a concrete problem solving experience than to generalize from it. Still, effective learning in CBR requires a well worked out set of methods in order to extract relevant knowledge from the experience, integrate a case into an existing knowledge structure, and index the case for later matching with similar cases.

---

[2]The term 'memory' is often used to refer to the storage structure that holds the existing cases, i.e. to the case base. A memory, thus, refers to what is remembered from previous experiences. Correspondingly, a reminding is a pointer structure to some part of memory.

[3]The learning approach of case-based reasoning is sometimes referred to as case-based learning. This term is sometimes also used synonymous with example-based learning, and may therefore point to classical induction and other generalization-driven learning methods. Hence, we will here use the term case-based reasoning both for the problem solving and learning part, and explicitly state which part we talk about whenever necessary.

1.4. Combining cases with other knowledge.

By examining theoretical and experimental results from cognitive psychology, it seems clear that human problem solving and learning in general are processes that involve the representation and utilization of several types of knowledge, and the combination of several reasoning methods. If cognitive plausibility is a guiding principle, an architecture for intelligence where the reuse of cases is at the centre, should also incorporate other and more general types of knowledge in one form or another. This is an issue of current concern in CBR research [Strube-91].

The rest of this paper is structured as follows: The next section gives a brief historical overview of the CBR field. This is followed by a grouping of CBR methods into a set of characteristic types, and a presentation of the descriptive framework which will be used throughout the paper to discuss CBR methods. Sections 4 to 8 discuss representation issues and methods related to the four main tasks of case-based reasoning, respectively. In chapter 9 we look at CBR in relation to integrated architectures and multistrategy problem solving and learning. This is followed by a short description of some fielded applications, and a few words about CBR development tools. The conclusion part briefly summarizes the paper, and point out some possible trends.

2. History of the CBR field

The roots of case-based reasoning in AI is found in the works of Roger Schank on dynamic memory and the central role that a reminding of earlier situations (episodes, cases) and situation patterns (scripts, MOPs) has in problem solving and learning [Schank-82]. Other trails into the CBR field has come from the study of analogical reasoning [Gentner-83], and - further back - from theories of concept formation, problem solving and experiential learning within philosophy and psychology (e.g. [Wittgenstein-53, Tulving-72, Smith-81]). For example, Wittgenstein observed that 'natural concepts', i.e. concepts that are part of the natural world - such as bird, orange, chair, car, etc. - are polymorphic. That is, their instances may be categorized in a variety of ways, and it is not possible to come up with a useful classical definition, in terms of a set of necessary and sufficient features, for such concepts. An answer to this problem is to represent a concept extensionally, defined by its set of instances - or cases.

The first system that might be called a case-based reasoner was the CYRUS system, developed by Janet Kolodner [Kolodner-83], at Yale University (Schank's group). CYRUS was based on Schank's dynamic memory model and MOP theory of problem solving and learning [Schank-82]. It was basically a question-answering system with knowledge of the various travels and meetings of former US Secretary of State Cyrus Vance. The case memory model developed for this system has later served as basis for several other case-based reasoning systems (including MEDIATOR [Simpson-85], PERSUADER [Sycara-88], CHEF [Hammond-89], JULIA [Hinrichs-92], CASEY [Koton-89]).

Another basis for CBR, and another set of models, were developed by Bruce Porter and his group [Porter-86] at the University of Texas, Austin. They initially addressed the machine learning problem of concept learning for classification tasks. This lead to the development of the PROTOS system [Bareiss-89], which emphasized on integrating general domain knowledge and specific case knowledge into a unified representation structure. The combination of cases with general domain knowledge was pushed further in GREBE [Branting-91], an application in the domain of law. Another early significant contribution to CBR was the work by Edwina Rissland and her group at the University of Massachusetts, Amhearst. With several law scientists in the group, they were interested in the role of precedence reasoning in legal judgements [Rissland-83]. Cases (precedents) are here not used to produce a single answer, but to interpret a situation in court, and to produce and assess arguments for both parties. This resulted in the HYPO system [Ashley-90], and later the combined case-based and rule-based system CABARET [Skalak-92]. Phyllis Koton at MIT studied the use of case-based reasoning to optimize performance in an existing knowledge based system, where the domain (heart failure) was described by a deep, causal model. This resulted in the CASEY system [Koton-89], in which case-based and deep model-based reasoning was combined.

In Europe, research on CBR was taken up a little later than in the US. The CBR work seems to have been stronger coupled to expert systems development and knowledge acquisition research than in the US. Among the earliest results was the work on CBR for complex technical diagnosis within the MOLTKE system, done by Michael Richter together with Klaus Dieter Althoff and others at the University of Kaiserslautern [Althoff-89]. This lead to the PATDEX system [Richter-91], with Stefan Wess as the main developer, and later to several other systems and methods [Althoff-91]. At IIIA in Blanes, Enric Plaza and Ramon Lopez de Mantaras developed a case-based learning apprentice system for medical diagnosis [Plaza-90], and Beatrice Lopez investigated the use of case-based methods for strategy-level reasoning [Lopez-90]. In Aberdeen, Derek Sleeman's group studied the use of cases for knowledge base refinement. An early result was the REFINER system, developed by Sunil Sharma [Sharma-88]. Another result is the IULIAN system for theory revision [Oehlmann-92]. At the University of Trondheim, Agnar Aamodt and colleagues at Sintef studied the learning aspect of CBR in the context of knowledge acquisition in general, and knowledge maintenance in particular. For problem solving, the combined use of cases and general domain knowledge was focused [Aamodt-89]. This lead to the development of the CREEK system and integration framework [Aamodt-91], and to continued work on knowledge-intensive case-based reasoning. On the cognitive science side, early work was done on analogical reasoning by Mark Keane, at Trinity College, Dublin, [Keane-88], a group that has developed into a strong environment for this type of CBR. In Gerhard Strube's group at the University of Freiburg, the role of episodic knowledge in cognitive models was investigated in the EVENTS project [Strube-90], which lead to the group's current research profile of cognitive science and CBR.

Currently, the CBR activities in the United States as well as in Europe are spreading out (see, e.g. [DARPA-91], [IEEE-92], [EWCBR-93], [Allemagne-93], and the rapidly growing number of papers on CBR in almost any AI journal). Germany seems to have taken a leading position in terms of number of active researchers, and several groups of significant size and activity level have been established recently. From Japan and other Asian countries, there are also activity points, for example in India [Venkatamaran-93]. In Japan, the interest is to a large extent focused towards the parallel computation approach to CBR [Kitano-93].

3. Fundamentals of case-based reasoning methods

Central tasks that all case-based reasoning methods have to deal with are to identify the current problem situation, find a past case similar to the new one, use that case to suggest a solution to the current problem, evaluate the proposed solution, and update the system by learning from this experience. How this is done, what part of the process that is focused, what type of problems that drives the methods, etc. varies considerably, however. Below is an attempt to classify CBR methods into types with roughly similar properties in this respect.

Main types of CBR methods.

The CBR paradigm covers a range of different methods for organizing, retrieving, utilizing and indexing the knowledge retained in past cases. Cases may be kept as concrete experiences, or a set of similar cases may form a generalized case. Cases may be stored as separate knowledge units, or splitted up into subunits and distributed within the knowledge structure. Cases may be indexed by a prefixed or open vocabulary, and within a flat or hierarchical index structure. The solution from a previous case may be directly applied to the present problem, or modified according to differences between the two cases. The matching of cases, adaptation of solutions, and learning from an experience may be guided and supported by a deep model of general domain knowledge, by more shallow and compiled knowledge, or be based on an apparent, syntactic similarity only. CBR methods may be purely self-contained and automatic, or they may interact heavily with the user for support and guidance of its choices. Some CBR method assume a rather large amount of widely distributed cases in its case base, while others are based on a more limited set of typical ones. Past cases may be retrieved and evaluated sequentially or in parallel.

Actually, "case-based reasoning" is just one of a set of terms used to refer to systems of this kind. This has lead to some confusions, particularly since case-based reasoning is a term used both as a generic term for several types of more specific approaches, as well as for one such approach. To some extent, this can also be said for analogy reasoning. An attempt of a clarification, although not resolving the confusions, of the terms related to case-based reasoning are given below.

- *Exemplar-based reasoning*.
  The term is derived from a classification of different views to concept definition into "the classical view", "the probabilistic view", and "the exemplar view" (see [Smith-81]). In the exemplar view, a concept is defined *extensionally*, as the set of its exemplars. CBR methods that address the learning of concept definitions (i.e. the problem addressed by most of the research in machine learning), are sometimes referred to as exemplar-based. Examples are early papers by Kibler and Aha [Kibler-87], and Bareiss and Porter [Porter-86]. In this approach, solving a problem is a *classification task*, i.e. finding the right class for the unclassified exemplar. The class of the most similar past case becomes the solution to the classification problem. The set of classes constitutes the set of possible solutions. Modification of a solution found is therefore outside the scope of this method.

- *Instance-based reasoning*.
  This is a specialization of exemplar-based reasoning into a highly *syntactic* CBR-approach. To compensate for lack of guidance from general background knowledge, a relatively large number of instances are needed in order to close in on a concept definition. The representation of the instances are usually simple (e.g. feature vectors), since a major focus is to study *automated learning* with no user in the loop. Instance-based reasoning labels recent work by Kibler and Aha and colleagues [Aha-91], and serves to distinguish their methods from more knowledge-intensive exemplar-based approaches (e.g. Protos' methods). Basically, this is a non-generalization approach to the concept learning problem addressed by classical, inductive machine learning methods.

- *Memory-based reasoning*.
  This approach emphasizes a collection of cases as a *large memory*, and reasoning as a process of accessing and searching in this memory. Memory organization and access is a focus of the case-based methods. The utilization of *parallel processing* techniques is a characteristic of these methods, and distinguishes this approach from the others. The access and storage methods may rely on purely syntactic criteria, as in the MBR-Talk system [Stanfill-88], or

they may attempt to utilize general domain knowledge, as in PARADYME [Kolodner-88] and the work done in Japan on massive parallel memories [Kitano-93].

- *Case-based reasoning*.
  Although case-based reasoning is used as a generic term in this paper, the *typical* case-based reasoning methods have some characteristics that distinguish them from the other approaches listed here. First, a typical case is usually assumed to have a certain degree of richness of information contained in it, and a certain *complexity* with respect to its internal organization. That is, a feature vector holding some values and a corresponding class is not what we would call a typical case description. What we refer to as typical case-based methods also has another characteristic property: They are able to *modify*, or adapt, a retrieved solution when applied in a different problem solving context. Paradigmatic case-based methods also utilizes *general background knowledge* - although its richness, degree of explicit representation, and role within the CBR processes varies. Core methods of typical CBR systems borrow a lot from cognitive psychology theories.

- *Analogy-based reasoning*.
  This term is sometimes used, as a synonym to case-based reasoning, to describe the typical case-based approach just described [Veloso-92]. However, it is also often used to characterize methods that solve new problems based on past cases from a *different domain,* while typical case-based methods focus on indexing and matching strategies for single-domain cases. Research on analogy reasoning is therefore a subfield concerned with mechanisms for identification and utilization of cross-domain analogies [Kedar-Cabelli-88, Hall-89]. The major focus of study has been on the *reuse* of a past case, what is called the mapping problem: Finding a way to transfer, or map, the solution of an identified analogue (called source or base) to the present problem (called target).

Throughout the paper we will continue to use the term case-based reasoning in the generic sense, although our examples, elaborations, and discussions will lean towards CBR in the more typical sense. The fact that a system is described as an example of some other approach, does not exclude it from being a typical CBR system as well. To the degree that more special examples of, e.g. instance-based, memory-based, or analogy-based methods will be discussed, this will be stated explicitly.

A descriptive framework.

Our framework for describing CBR methods and systems has two main parts:

- A process model of the CBR cycle
- A task-method structure for case-based reasoning

The two models are complementary and represent two views on case-based reasoning. The first is a dynamic model that identifies the main subprocesses of a CBR cycle, their interdependencies and products. The second is a task-oriented view, where a task decomposition and related problem solving methods are described. The framework will be used in subsequent parts to identify and discuss important problem areas of CBR, and means of dealing with them.

The CBR cycle

At the highest level of generality, a general CBR cycle may be described by the following four processes[4]:

[4]As a mnemonic, try "the four REs".

1. RETRIEVE the most similar case or cases
2. REUSE the information and knowledge in that case to solve the problem
3. REVISE the proposed solution
4. RETAIN the parts of this experience likely to be useful for future problem solving

A new problem is solved by *retrieving* one or more previously experienced cases, *reusing* the case in one way or another, *revising* the solution based on reusing a previous case, and *retaining* the new experience by incorporating it into the existing knowledge-base (case-base). The four processes each involve a number of more specific steps, which will be described in the task model. In figure 1, this cycle is illustrated.

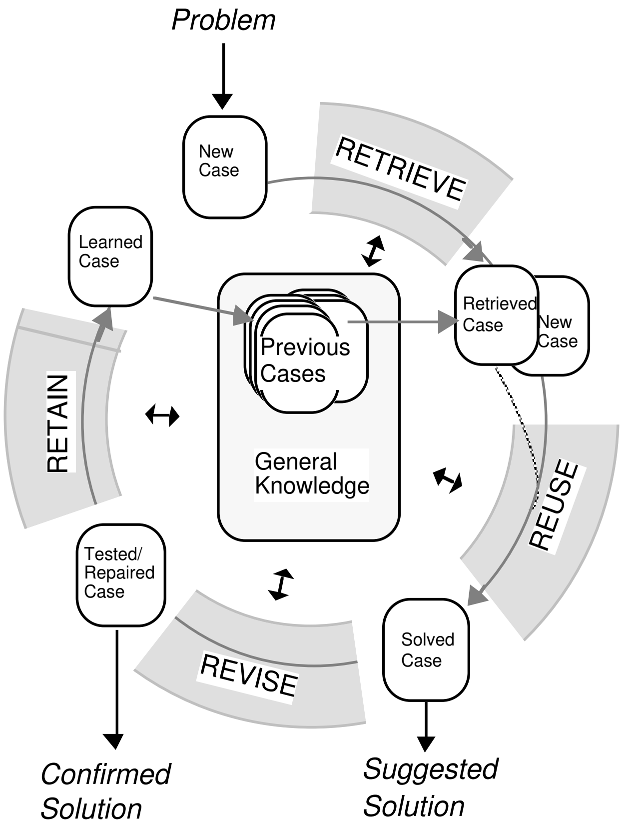

Figure 1. The CBR Cycle

An initial description of a problem (top of figure) defines a *new case*. This new case is used to RETRIEVE a case from the collection of *previous cases*. The *retrieved case* is combined with the new case - through REUSE - into a *solved case*, i.e. a proposed solution to the initial problem. Through the REVISE process this solution is tested for success, e.g. by being applied to the real world environment or evaluated by a teacher, and repaired if failed. During RETAIN, useful experience is retained for future reuse, and the case base is updated by a new *learned case*, or by modification of some existing cases.

As indicated in the figure, general knowledge usually plays a part in this cycle, by supporting the CBR processes. This support may range from very weak (or none) to very strong, depending on the type of CBR method. By general knowledge we here mean general domain-dependent knowledge, as opposed to specific knowledge embodied by cases. For example, in diagnosing a patient by retrieving and reusing the case of a previous patient, a model of anatomy together with causal relationships between pathological states may constitute the general knowledge used by a CBR system. A set of rules may have the same role.

A hierarchy of CBR tasks

The process view just described was chosen in order to emphasize on CBR as a cycle of sequential steps. To further decompose and describe the four top-level steps, we switch to a task-oriented view, where each step, or subprocess, is viewed as a task that the CBR reasoner has to achieve. While a process-oriented view enables a global, external view to what is happening, a task oriented view is suitable for describing the detailed mechanisms from the perspective of the CBR reasoner itself. This is coherent with a task-oriented view of knowledge level modeling [Van de Velde-93]. At the knowledge level, a system is viewed as an agent which has goals, and means to achieve its goals. A system description can be made from three perspectives: Tasks, methods and domain knowledge models. Tasks are set up by the goals of the system, and a task is performed by applying one or more methods. For a method to be able to accomplish a task, it needs knowledge about the general application domain as well as information about the current problem and its context. Our framework and analysis approach is strongly influenced by knowledge level modeling methods, particularly the Components of Expertise methodology [Steels-90, Steels-93].

The task-method structure we will refer to in subsequent parts of the paper is shown in figure 2. Tasks have node names in bold letters, while methods are written in italics. The links between task nodes (plain lines) are *task decompositions*, i.e. part-of relations, where the direction of the relationship is downwards. The top-level task is problem solving and learning from experience and the method to accomplish the task is *case-based reasoning* (indicated in a special way by a stippled arrow). This splits the top-level task into the four major CBR tasks corresponding to the four processes of figure 1, retrieve, reuse, revise, and retain. All the four tasks are necessary in order to perform the top-level task. The retrieve task is, in turn, partitioned in the same manner (by a retrieval method) into the tasks identify (relevant descriptors), search (to find a set of past cases), initial match (the relevant descriptors to past cases), and select (the most similar case).

All task partitions in the figure are complete, i.e. the set of subtasks of a task are intended to be sufficient to accomplish the task, at this level of description. The figure does not show any control structure over the subtasks, although a rough sequencing of them is indicated by having put earlier subtasks higher up on the page than those that follow (for a particular set of subtasks). The actual control is specified as part of the problem solving method. The relation between tasks and methods (stippled lines) identify alternative methods applicable for solving a task. A method specifies the algorithm that identifies and controls the execution of subtasks, and accesses and utilizes the knowledge and information needed to do this. The methods shown are high level method classes, from which one or more specific methods should be chosen. The method set as shown is incomplete, i.e. one of the methods indicated may be sufficient to solve the task, several methods may be combined, or there may be other methods that can do the job. The methods shown in the figure are task decomposition and control methods. At the bottom level of the task hierarchy (not shown), a task is solved directly, i.e. by what may be referred to as task execution methods.

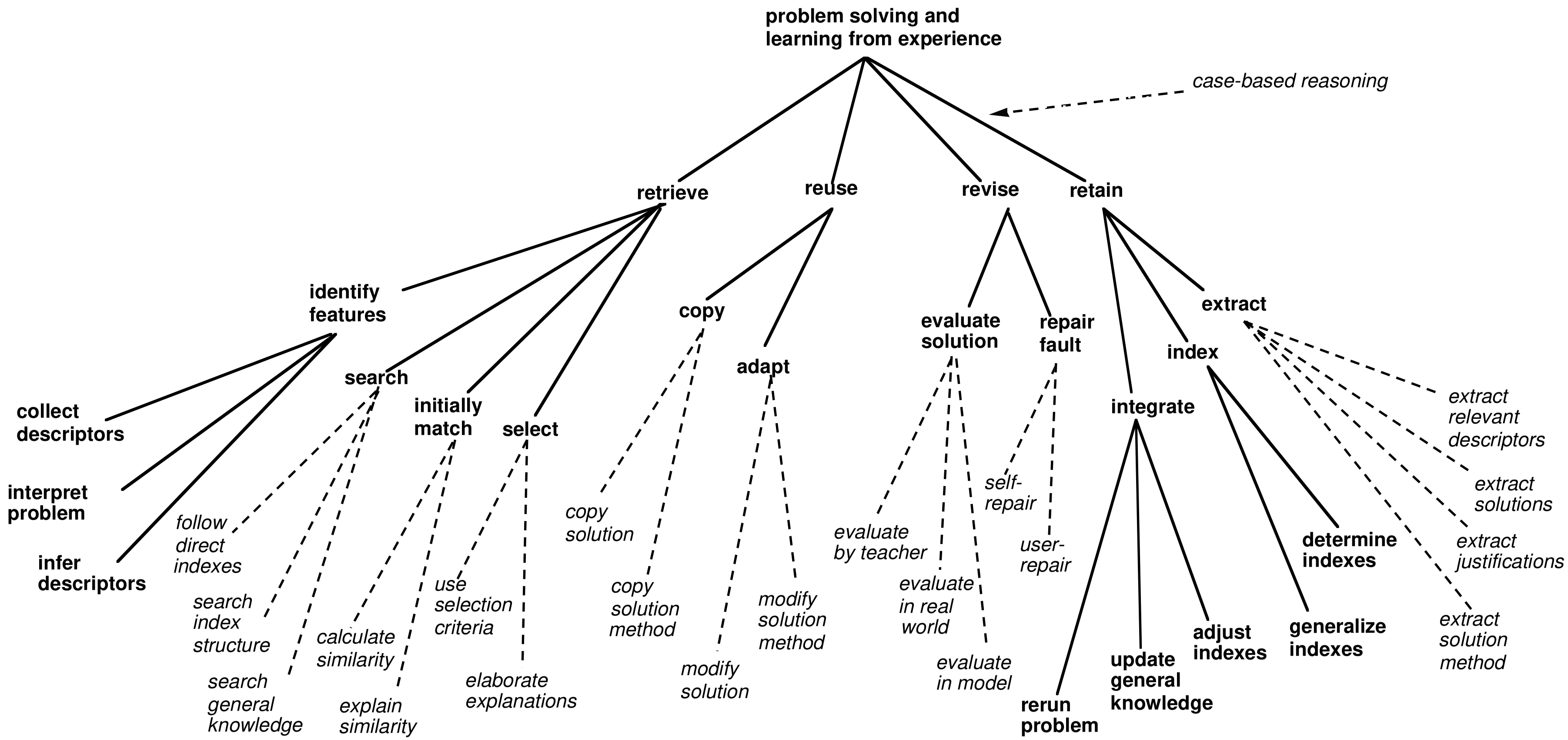

Figure 2. A task-method decomposition of CBR

CBR Problem Areas

As for AI in general, there are no universal CBR methods suitable for every domain of application. The challenge in CBR as elsewhere is to come up with methods that are suited for problem solving and learning in particular subject domains and for particular application environments. In line with the task model just shown, core problems addressed by CBR research can be grouped into five areas. A set of coherent solutions to these problems constitutes a CBR method:

- *Knowledge representation*
- *Retrieval methods*
- *Reuse methods*
- *Revise methods*
- *Retain methods*

In the next five sections, we give an overview of the main problem issues related to these five areas, and exemplify how they are solved by some existing methods. Our examples will be drawn from the six systems PROTOS, CHEF, CASEY, PATDEX, BOLERO, and CREEK. In the recently published book by Janet Kolodner [Kolodner-93] these problems are discussed and elaborated to substantial depth, and hints and guidelines on how to deal with them are given.

4. Representation of Cases

A case-based reasoner is heavily dependent on the structure and content of its collection of cases - often referred to as its case memory. Since a problem is solved by recalling a previous experience suitable for solving the new problem, the case search and matching processes need to be both effective and reasonably time efficient. Further, since the experience from a problem just solved has to be retained in some way, these requirements also apply to the method of integrating a new case into the memory. The representation problem in CBR is primarily the problem of deciding what to store in a case, finding an appropriate structure for describing case contents, and deciding how the case memory should be organized and indexed for effective retrieval and reuse. An additional problem is how to integrate the case memory structure into a model of general domain knowledge, to the extent that such knowledge is incorporated.

In the following subsection, two influential case memory models are briefly reviewed: The dynamic memory model of Schank and Kolodner, and the category-exemplar model of Porter and Bareiss[5].

The Dynamic Memory Model

As previously mentioned, the first system that may be referred to as a case-based reasoner was Kolodner's CYRUS system, based on Schank's dynamic memory model [Schank-82]. The case memory in this model is a hierarchical structure of what is called 'episodic memory organization packets' (E-MOPs [Kolodner-83a, Kolodner-83b]), also referred to as generalized episodes [Koton-89]. This model was developed from Schank's more general MOP theory. The basic idea is to organize specific cases which share similar properties under a more general structure (a generalized episode). A generalized episode (GE) contains three different types of objects: *Norms*, *cases* and *indices*. Norms are features common to all cases indexed under a GE. Indices are features which discriminate between a GE's cases. An index may point to a more specific

[5]Other early models include Rissland and Ashley's HYPO system [Rissland-87] in which cases are grouped under a set of domain-specific dimensions, and Stanfill and Waltz' MBR model, designed for parallel computation rather than knowledge-based matching.

generalized episode, or directly to a case. An index is composed of two terms: An index name and an index value.

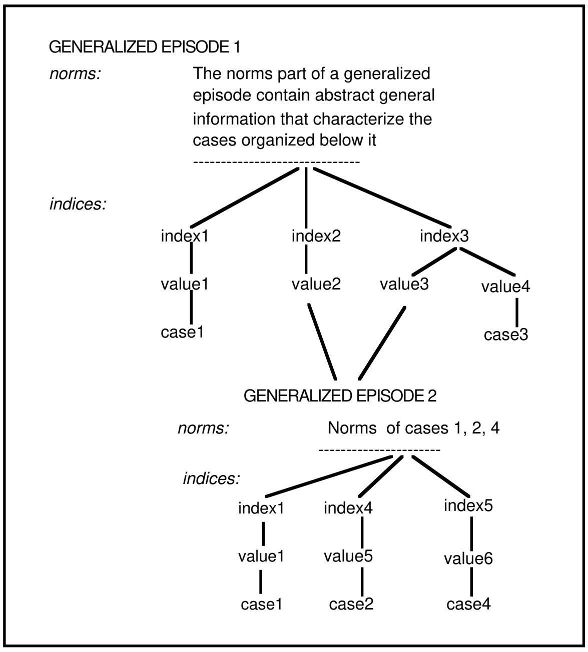

Figure 3: Structure of cases and generalized episodes.

Figure 3 illustrates this structure. The figure illustrates a complex generalized episode, with its underlying cases and more specific GE. The entire case memory is a discrimination network where a node is either a generalized episode (containing the norms), an index name, index value or a case. Each index-value pair points from a generalized episode to another generalized episode or to a case. An index value may only point to a single case or a single generalized episode. The indexing scheme is redundant, since there are multiple paths to a particular case or GE. This is illustrated in the figure by the indexing of case1.

When a new case description is given and the best matching is searched for, the input case structure is 'pushed down' the network structure, starting at the root node. The search procedure is similar for case retrieval as for case storing. When one or more features of the case matches one or more features of a GE, the case is further discriminated based on its remaining features. Eventually, the case with most features in common with the input case is found[6]. During storing of a new case, when a feature of the new case matches a feature of an existing case, a generalized episode is created. The two cases are then discriminated by indexing them under different indices below this generalized episode. If - during the storage of a case - two cases (or two GEs) end up under the same index, a new generalized episode is automatically created. Hence, the memory structure is *dynamic* in the sense that similar parts of two case descriptions are dynamically generalized into a GE, and the cases are indexed under this GE by their difference features.

A case is retrieved by finding the GE with most norms in common with the problem description. Indices under that GE are then traversed in order to find the case which contains most of the additional problem features. Storing a new case is performed in the same way, with the additional

[6]This may not be the right similarity criterion, and is mentioned just to illustrate the method. Similarity criteria may favour matching of a particular subset of features, or there may be other means of assessing case similarity. Similarity assessment criteria can in turn can be used to guide the search - for example by identifying which indexes to follow first if there is a choice to be made.

process of dynamically creating generalized episodes, as described above. Since the index structure is a discrimination network, a case (or pointer to a case) is stored under each index that discriminates it from other cases. This may easily lead to an explosive growth of indices with increased number of cases. Most systems using this indexing scheme therefore put some limits to the choice of indices for the cases. In CYRUS, for example, only a small vocabulary of indices is permitted.

CASEY stores a large amount of information in its cases. In addition to all observed features, it retains the causal explanation for the diagnosis found, as well as the list of states in the heart failure model for which there was evidence in the patient. These states, referred to as generalized causal states, are also the primary indices to the cases.

The primary role of a generalized episode is as an indexing structure for matching and retrieval of cases. The dynamic properties of this memory organization, however, may also be viewed as an attempt to build a memory structure which integrates knowledge from specific episodes with knowledge generalized from the same episodes. It is therefore claimed that this knowledge organization structure is suitable for learning generalized knowledge as well as case specific knowledge, and that it is a plausible - although simplified - model of human reasoning and learning.

The Category & Exemplar Model

The PROTOS system, built by Ray Bareiss and Bruce Porter [Bareiss-89, Porter-90], proposes an alternative way to organize cases in a case memory. Cases are also referred to as *exemplars*. The psychological and philosophical basis of this method is the view that 'real world', natural concepts should be defined extensionally. Further, different features are assigned different importances in describing a case's membership to a category. Any attempt to generalize a set of cases should - if attempted at all - be done very cautiously. This fundamental view of concept representation forms the basis for this memory model.

The case memory is embedded in a network structure of *categories*, *cases*, and *index pointers*. Each case is associated with a category. An index may point to a case or a category. The indices are of three kinds: Feature links pointing from problem descriptors (features) to cases or categories (called remindings), case links pointing from categories to its associated cases (called exemplar links), and difference links pointing from cases to the neighbour cases that only differs in one or a small number of features. A feature is, generally, described by a name and a value. A category's exemplars are sorted according to their degree of prototypicality in the category.

Figure 4 illustrates a part of this memory structure, i.e. the linking of features and cases (exemplars) to categories. The unnamed indices are remindings from features to a category.

Within this memory organization, the categories are inter-linked within a semantic network, which also contains the features and intermediate states (e.g. subclasses of goal concepts) referred to by other terms. This network represents a background of general domain knowledge, which enables explanatory support to some of the CBR tasks. For example, a core mechanism of case matching is a method called 'knowledge-based pattern matching'.

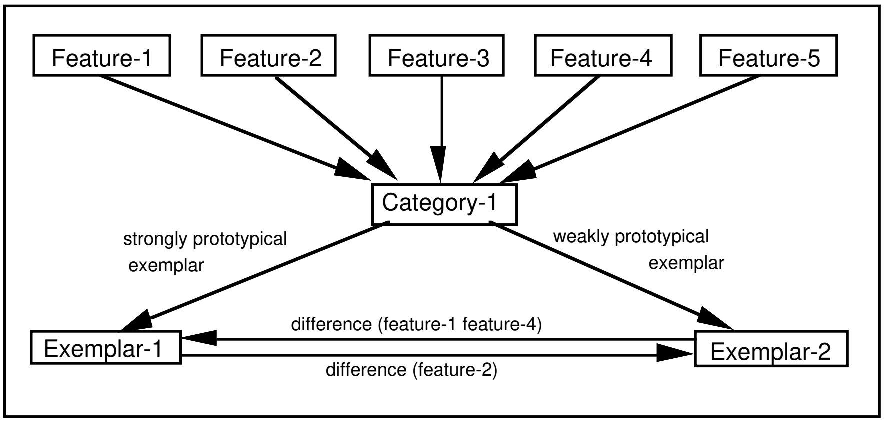

Figure 4: The Structure of Categories, Features and Exemplars

Finding a case in the case base that matches an input description is done by combining the input features of a problem case into a pointer to the case or category that shares most of the features. If a reminding points directly to a category, the links to its most prototypical cases are traversed, and these cases are returned. As indicated above, general domain knowledge is used to enable matching of features that are semantically similar. A new case is stored by searching for a matching case, and by establishing the appropriate feature indices. If a case is found with only minor differences to the input case, the new case may not be retained or the two cases may be merged by following taxonomic links in the semantic network.

5. Case Retrieval

The Retrieve task starts with a (partial) problem description, and ends when a best matching previous case has been found. Its subtasks are referred to as Identify Features, Initially Match, Search, and Select, executed in that order. The identification task basically comes up with a set of relevant problem descriptors, the goal of the matching task is to return a set of cases that are sufficiently similar to the new case - given a similarity threshold of some kind, and the selection task works on this set of cases and chooses the best match (or at least a first case to try out).

While some case-based approaches retrieve a previous case largely based on superficial, *syntactical similarities* among problem descriptors (e.g. the CYRUS system [Kolodner-83a], ARC [Plaza-90], and PATDEX-1 [Richter-91] systems), some approaches attempt to retrieve cases based on features that have deeper, *semantical similarities* (e.g. the PROTOS [Bareiss-88], CASEY [Koton-89], GREBE [Branting-89], CREEK [Aamodt-90], and MMA [Plaza-93] systems). Ongoing work in the FABEL project, aimed to develop a decision support system for architects, explores various methods for combined reasoning and mutual support of different knowledge types [Fabel-Consortium-93]. In order to match cases based on semantic similarities and relative importance of features, an extensive body of general domain knowledge is needed to produce an explanation of why two cases match and how strong the match is. Syntactic similarity assessment - sometimes referred to as a "knowledge-poor" approach - has its advantage in domains where general domain knowledge is very difficult or impossible to acquire. On the other hand, semantical oriented approaches - referred to as "knowledge-intensive"[7] - are able to use the contextual meaning of a problem description in its matching, for domains where general domain knowledge is available.

[7]Note that syntactic oriented methods may also contain a lot of general domain knowledge, implicit in their matching methods. The distinction between knowledge-poor and knowledge-intensive is therefore related to *explicitly represented* domain knowledge. Further, it refers to *generalized* domain knowledge, since cases also contain explicit knowledge, but this is *specialized* (specific) domain knowledge.

A question that should be asked when deciding on a retrieval strategy, is the purpose of the retrieval task. If the purpose is to retrieve a case which is to be adapted for reuse, this can be accounted for in the retrieval method. Approaches to 'retrieval for adaptation' have for example been suggested for retrieval of cases for design problem solving [Börner-93], and for analogy reasoning [Cunningham-93, O'Hara-92].

Identify Feature

To identify a problem may involve simply noticing its input descriptors, but often - and particularly for knowledge-intensive methods - a more elaborate approach is taken, in which an attempt is made to 'understand' the problem within its context. Unknown descriptors may be disregarded or requested to be explained by the user. In PROTOS, for example, if an input feature is unknown to the system, the user is asked to supply an explanation that links the feature into the existing semantic network (category structure). To understand a problem involves to filter out noisy problem descriptors, to infer other relevant problem features, to check whether the feature values make sense within the context, to generate expectations of other features, etc. Other descriptors than those given as input, may be inferred by using a general knowledge model, or by retrieving a similar problem description from the case base and use features of that case as expected features. Checking of expectations may be done within the knowledge model (cases and general knowledge), or by asking the user.

Initially Match

The task of finding a good match is typically split into two subtasks: An initial matching process which retrieves a set of plausible candidates, and a more elaborate process of selecting the best one among these. The latter is the Select task, described below. Finding a set of matching cases is done by using the problem descriptors (input features) as indexes to the case memory in a direct or indirect way. There are in principle three ways of retrieving a case or a set of cases: By following direct index pointers from problem features, by searching an index structure, or by searching in a model of general domain knowledge. PATDEX implements the first strategy for its diagnostic reasoning, and the second for test selection. A domain-dependent, but global similarity metric is used to assess similarity based on surface match. Dynamic memory based systems takes the second approach, but general domain knowledge may be used in combination with search in the discrimination network. PROTOS and CREEK combines one and three, since direct pointers are used to hypothesize a candidate set which in turn is justified as plausible matches by use of general knowledge.

Cases may be retrieved solely from input features, or also from features inferred from the input. Cases that match all input features are, of course, good candidates for matching, but - depending on the strategy - cases that match a given fraction of the problem features (input or inferred) may also be retrieved. PATDEX uses a global similarity metric, with several parameters that are set as part of the domain analysis. Some tests for relevance of a retrieved case is often executed, particularly if cases are retrieved on the basis of a subset of features. For example, a simple relevance test may be to check if a retrieved solution conforms with the expected solution type of the new problem. A way to assess the degree of similarity is needed, and several 'similarity metrics' have been proposed, based on surface similarities of problem and case features.

Similarity assessment may also be more knowledge-intensive, for example by trying to understand the problem more deeply, and using the goals, constraints, etc. from this elaboration process to guide the matching [Aamodt-93]. Another option is to weigh the problem descriptors according to their importance for characterizing the problem, during the learning phase. In PROTOS, for example, each feature in a stored case has assigned to it a degree of importance for the solution of the case. A similar mechanism is adopted by CREEK, which stores both the predictive strength (discriminatory value) of a feature with respect to the set of cases, as well as a features criticality, i.e. what influence the lack of a feature has on the case solution.

Select

From the set of similar cases, a best match is chosen. This may have been done during the initial match process, but more often a set of cases are returned from that task. The best matching case is usually determined by evaluating the degree of initial match more closely. This is done by an attempt to generate explanations to justify non-identical features, based on the knowledge in the semantic network. If a match turns out not to be strong enough, an attempt to find a better match by following difference links to closely related cases is made. This subtask is usually a more elaborate one than the retrieval task, although the distinction between retrieval and elaborate matching is not distinct in all systems. The selection process typically generates consequences and expectations from each retrieved case, and attempts to evaluate consequences and justify expectations. This may be done by using the system's own model of general domain knowledge, or by asking the user for confirmation and additional information. The cases are eventually ranked according to some metric or ranking criteria. Knowledge-intensive selection methods typically generate explanations that support this ranking process, and the case that has the strongest explanation for being similar to the new problem is chosen. Other properties of a case that are considered in some CBR systems include relative importance and discriminatory strengths of features, prototypicality of a case within its assigned class, and difference links to related cases.

6. Case Reuse

The reuse of the retrieved case solution in the context of the new case focuses on two aspects: (a) the differences among the past and the current case and (b) what part of a retrieved case can be transferred to the new case.

Copy

In simple classification tasks the differences are abstracted away (they are considered non relevant while similarities are relevant) and the solution class of the retrieved case is transferred to the new case as its solution class. This is a trivial type of reuse. However, other systems have to take into account differences in (a) and thus the reused part (b) cannot be directly transferred to the new case but requires an *adaptation* process that takes into account those differences.

Adapt

There are two main ways to reuse past cases[8]: (1) reuse the past case solution (transformational reuse), and (2) reuse the past method that constructed the solution (derivational reuse). In transformational reuse the past case solution is not directly a solution for the new case but there exists some knowledge in the form of transformational operators {T} such that applied to the old solution they transform it into a solution for the new case. A way to organize this T operators is to index them around the differences detected among the retrieved and current cases. An example of this is CASEY, where a new causal explanation is built from the old causal explanations by rules with condition-part indexing differences and with a transformational operator T at the action part of the rule. Transformational reuse does not look how a problem is solved but focuses on the

[8] We adapt here the distinction between transformational and derivational analogy, put forth in [Carbonell-86].

equivalence of solutions, and this requires a strong domain-dependent model in the form of transformational operators {T} plus a control regime to organize the operators application.

Derivational reuse looks at how the problem was solved in the retrieved case. The retrieved case holds information about the method used for solving the retrieved problem including a justification of the operators used, subgoals considered, alternatives generated, failed search paths, etc. Derivational reuse then reinstantiates the retrieved method to the new case and "replays" the old plan into the new context (usually general problem solving systems can be seen here as planning systems). During the replay successful alternatives, operators, and paths will be explored first while filed paths will be avoided; new subgoals are pursued based on the old ones and old subplans can be recursively retrieved for them. An example of derivational reuse is the Analogy/Prodigy system that reuses past plans guided by commonalties of goals and initial situations, and resumes a means-ends planning regime if the retrieved plan fails or is not found.

7. Case Revision

When a case solution generated by the reuse phase is not correct, an opportunity for learning from failure arises. This phase is called case revision and consists of two tasks: (1) evaluate the case solution generated by reuse. If successful, learning from the success (case retainment, see next section), (2) otherwise repair the case solution using domain-specific knowledge.

Evaluate solution
The evaluation task takes the result from applying the solution in the real environment (asking a teacher or performing the task in the real world). This is usually a step outside the CBR system, since it - at least for a system in normal operation - involves the application of a suggested solution to the real problem. The results from applying the solution may take some time to appear, depending on the type of application. In a medical decision support system, the success or failure of a treatment may take from a few hours up to several months. The case may still be learned, and be available in the case base in the intermediate period, but it has to be marked as a non-evaluated case. A solution may also be applied to a simulation program that is able to generate a correct solution. This is used in CHEF, where a solution (i.e. a cooking recipe) is applied to an internal model assumed to be strong enough to give the necessary feedback for solution repair.

Repair fault
Case repair involves detecting the errors of the current solution and retrieving or generating explanations for them. The best example is the CHEF system, where causal knowledge is used to generate an explanation of why certain goals of the solution plan were not achieved. CHEF learns the general situations that will cause the failures using an explanation-based learning technique. This is included into a failure memory that is used in the reuse phase to predict possible shortcomings of plans. This form of learning moves detection of errors in a post hoc fashion to the elaboration plan phase were errors can be predicted, handled and avoided. A second task of the revision phase is the solution repair task. This task uses the failure explanations to modify the solution in such a way that failures do not occur. For instance, the failed plan in the CHEF system is modified by a repair module that adds steps to the plan that will assure that the causes of the errors will not occur. The repair module possesses general causal knowledge and domain knowledge about how to disable or compensate causes of errors in the domain. The revised plan can then be retained directly (if the revision phase assures its correctness) or it can be evaluated and repaired again.

8. Case Retainment - Learning

This is the process of incorporating what is useful to retain from the new problem solving episode into the existing knowledge. The learning from success or failure of the proposed solution is triggered by the outcome of the evaluation and possible repair. It involves selecting which information from the case to retain, in what form to retain it, how to index the case for later retrieval from similar problems, and how to integrate the new case in the memory structure.

Extract

In CBR the case base is updated no matter how the problem was solved. If it was solved by use of a previous case, a new case may be built or the old case may be generalized to subsume the present case as well. If the problem was solved by other methods, including asking the user, an entirely new case will have to be constructed. In any case, a decision need to be made about what to use as the source of learning. Relevant problem descriptors and problem solutions are obvious candidates. But an explanation or another form of justification of why a solution is a solution to the problem may also be marked for inclusion in a new case. In CASEY and CREEK, for example, explanations are included in retained cases, and reused in later modification of the solution. CASEY uses the previous explanation structure to search for other states in the diagnostic model which explains the input data of the new case, and to look for causes of these states as answers to the new problem. This focuses and speeds up the explanation process, compared to a search in the entire domain model. The last type of structure that may be extracted for learning is the problem solving method, i.e. the strategic reasoning path, making the system suitable for derivational reuse.

Failures, i.e. information from the Revise task, may also be extracted and retained, either as separate failure cases or within total-problem cases. When a failure is encountered, the system can then get a reminding to a previous similar failure, and use the failure case to improve its understanding of - and correct - the present failure.

Index

The 'indexing problem' is a central and much focused problem in case-based reasoning. It amounts to deciding what type of indexes to use for future retrieval, and how to structure the search space of indexes. Direct indexes, as previously mentioned, skips the latter step, but there is still the problem of identifying what type of indexes to use. This is actually a knowledge acquisition problem, and should be analyzed as part of the domain knowledge analysis and modeling step. A trivial solution to the problem is of course to use all input features as indices. This is the approach of syntax-based methods within instance-based and memory-based reasoning. In the memory-based method of CBR-Talk [Stanfill-86], for example, relevant features are determined by matching, in parallel, all cases in the case-base, and filtering out features that belong to cases with few features in common with the problem case.

In CASEY, a two-step indexing method is used. Primary index features are - as referred to in the section on representation - general causal states in the heart failure model that are part of the explanation of the case. When a new problem enters, the features are propagated in the heart failure model, and the states that explain the features are used as indices to the case memory. The observed features themselves are used as secondary features only.

Integrate

This is the final step of updating the knowledge base with new case knowledge. If no new case and index set has been constructed, it is the main step of Retain. By modifying the indexing of existing cases, CBR systems learn to become better similarity assessors. The tuning of existing indexes is an important part of CBR learning. Index strengths or importances for a particular case or solution are adjusted due to the success or failure of using the case to solve the input problem. For features that have been judged relevant for retrieving a successful case, the association with the case is strengthened, while it is weakened for features that lead to unsuccessful cases being retrieved. In this way, the index structure has a role of tuning and adapting the case memory to its use. PATDEX has a special way to learn feature relevance: A relevance matrix links possible features to the diagnosis for which they are relevant, and assign a weight to each such link. The weights are updated, based on feedback of success or failure, by a connectionist method.

In knowledge-intensive approaches to CBR, learning may also take place within the general conceptual knowledge model, for example by other machine learning methods (see next section) or through interaction with the user. Thus, with a proper interface to the user (whether a competent end user or an expert) a system may incrementally extend and refine its general knowledge model, as well as its memory of past cases, in the normal course of problem solving. This is an inherent method in the PROTOS system, for example. All general knowledge in PROTOS is assumed to be acquired in such a bottom-up interaction with a competent user.

The case just learned may finally be tested by re-entering the initial problem and see whether the system behaves as wanted.

9. Integrated approaches

Most CBR systems make use of general domain knowledge in addition to knowledge represented by cases. Representation and use of that domain knowledge involves integration of the case-based method with other methods and representations of problem solving, for instance rule-based systems or deep models like causal reasoning. The overall architecture of the CBR *system* has to determine the interactions and control regime between the CBR *method* and the other components. For instance, the CASEY system integrates a model-based causal reasoning program to diagnose heart diseases. When the case-based method fails to provide a correct solution CASEY executes the model-based method to solve the problem and stores the solution as a new case for future use. Since the model-based method is complex and slow, the case-based method in CASEY is essentially a way to achieve speed-up learning. The integration of model-based reasoning is also important for the case-based method itself: the causal model of the disease of a case is what is retrieved and reused in CASEY.

An example of integrating rules and cases is the BOLERO system [Lopez-93]. BOLERO is a meta-level architecture where the base-level is composed of rules embodying knowledge to diagnose the plausible pneumonias of a patient, while the meta-level is a case-based planner that, at every moment, is able to dictate which diagnoses are worthwhile to consider. Thus in BOLERO the rule-based level contains domain knowledge (how to deduce plausible diagnosis from patient facts) while the meta-level contains strategic knowledge (it plans, from all possible goals, which are likely to be successfully achieved). The case-based planner is therefore used to control the space searched by the rule-based level, achieving a form of speed-up learning. The control regime between the two components is interesting: the control passes to the meta-level whenever some new information is known at the base level, assuring that the system is dynamically able to generate a more appropriate strategic plan. This control regime in the meta-level architecture assures that the case-based planner is capable of *reactive behaviour*, i.e. of modifying plans reacting to situation changes. Also the clear separation of rule-based and case-based methods in two different levels of computation is important: it clarifies their distinction and their interaction.

The integration of CBR with other reasoning paradigms is closely related to the general issue of architectures for unified problem solving and learning. These approach is a current trend in machine learning with architectures such as Soar, Theo, or Prodigy. CBR as such is a form of combining problem solving (through retrieval and reuse) and learning (through retainment). However, as we have seen, other forms of representation and reasoning are usually integrated into a CBR system and thus the general issue is an important dimension into CBR research. In the CREEK architecture, the cases, heuristic rules, and deep models are integrated into a unified knowledge structure. The main role of the general knowledge is to provide explanatory support to the case-based processes [Aamodt-93], rules or deep models may also be used to solve problems on their own if the case-based method fails. Usually the domain knowledge used in a CBR system is acquired through knowledge acquisition in the normal way for knowledge-based systems. Another option would be to also learn that knowledge from the cases. In this situation it can be learnt in a case-based way or by induction. This line of work is currently being developed in Europe by systems like the Massive Memory Architecture and INRECA [Manago-93]. These systems are closely related to the *multistrategy learning systems* [Michalski 92]: the issues of integrating different problem solving and learning methods are essential to them.

The Massive Memory Architecture (MMA) [Plaza 93a] is an integrated architecture for learning and problem solving based on reuse of case experiences retained in the systems memory. A goal of MMA is understanding and implementing the relationship between learning and problem solving into a reflective or introspective framework: the system is able to inspect its own past behaviour in order to learn how to change its structure so as to improve is future performance. Case-based reasoning methods are implemented by retrieval methods (to retrieve past cases), a language of preferences (to select the best case) and a form of derivational analogy (to reuse the retrieved method into the current problem). A problem in the MMA does not use one CBR method, since several CBR methods can be programmed for different subgoals by means of specific retrieval methods and domain-dependent preferences. Learning in MMA is viewed as a form of introspective inference, where the reasoning is not about a domain but about the past behaviour of the system and about ways to modify and improve this behaviour. This view supports integration of case-based learning as well as of other forms of learning from examples, like inductive methods, which are also integrated into the MMA and combined with case-based methods.

10. Example applications and tools

As a relatively young field, CBR can not yet brag about a lot of fielded applications. But there are some. We briefly summarize two of these systems, to illustrate how CBR methods can successfully realize knowledge-based decision support systems.

Two Fielded Applications

At Lockheed, Palo Alto, the first fielded CBR system was developed. The problem domain is optimization of autoclave loading for heat treatment of composite materials [Hennesy-92]. The autoclave is a large convection oven, where airplane parts are treated in order to get the right properties. Different material types need different heating and cooling procedures, and the task is to load the autoclave for optimized throughput, i.e. to select the parts that can be treated together, and distribute them in the oven so that their required heating profiles are taken care of. There are always more parts to be cured than the autoclave can take in one load. The knowledge needed to perform this task reasonably well used to reside in the head of a just a few experienced people. There is no theory and very few generally applicable schemes for doing this job, so to build up experience in the form of previously successful and unsuccessful situations is important. The motivation for developing this application was to be able to remember the relevant earlier situations. Further, a decision support system would enable other people than the experts to do the job, and to help training new personnel. The development of the system started in 1987, and it has been in regular use since the fall 1990. The results so far are very positive. The current system handles the configuration of one loading operation in isolation, and an extended system to handle the sequencing of several loads is under testing. The development strategy of the application has been to hold a low-risk profile, and to include more advanced functionalities and solutions as experience with the system has been gained over some time.

The second application has been developed at General Dynamics, Electric Boat Division [Brown-91]. During construction of ships, a frequently re-occurring problem is the selection of the most appropriate mechanical equipment, and to fit it to its use. Most of these problems can be handled by fairly standard procedures, but some problems are harder and occur less frequently. These type of problems - referred to as "non-conformances" - also repeat over time, and because regular procedures are missing, they consume a lot of resources to get solved. General Dynamics wanted to see whether a knowledge-based decision support tool could reduce the cost of these problems. The application domain chosen was the selection and adjustment of valves for on-board pipeline systems. The development of the first system started in 1986, using a rule-based systems approach. The testing of the system on real problems initially gave positive results, but problems of brittleness and knowledge maintenance soon became apparent. In 1988 a feasibility study was made of the use of case-based reasoning methods instead of rules, and a prototype CBR system was developed. The tests gave optimistic results, and an operational system was fielded in 1990. The rule-base was taken advantage of in structuring the case knowledge and filling the initial case base. In the fall of 1991 the system was continually used in three out of four departments involved with mechanical construction. A quantitative estimate of cost reductions has been made: The rule-based system took 5 man-years to develop, and the same for the CBR system (2 man-years of studies and experimental development and 3 man-years for the prototype and operational system). This amounts to $750.000 in total costs. In the period December 90 - September 91 20.000 non-conformances were handled. The cost reduction, compared to previous costs of manual procedures, was about 10%, which amounts to a saving of $240.000 in less than one year.

There are many any other applications in test use or more or less regular use. A rapidly growing application type is "help desk systems" [Kolodner-92, Simoudis-92], where basically case-based indexing and retrieval methods are used to retrieve cases, which then are viewed as information chunks for the user, instead of sources of knowledge for reasoning[9]. Such a system can also be a first step towards a more full-fledged CBR system.

Tools

Several commercial companies offer shells for building CBR systems. Just as for rule-based systems shells, they enable you to quickly develop applications, but at the expense of flexibility of representation, reasoning approach and learning methods. In [Harmon-92] Paul Harmon reviewed four such shells: ReMind from Cognitive Systems Inc., CBR Express/ART-IM from Inference Corporation, Esteem from Esteem Software Inc., and Induce-it (later renamed to CasePower) from Inductive Solutions, Inc. The first three of these were reviewed more thoroughly by Thomas Schult for the German AI journal [Schult-92]. The example of CBR Express and ART-IM is typical, since

---

[9]A gradual transition from such a system, starting with some help desk and information filtering [AI-Com-92] functions, and moving to an advice-giving case-based decision support tool, is the approach taken in a system currently being specified for the Norwegian oil industry [Nordbø-92].

many vendors offer CBR extensions to an existing tool. On the European scene Acknosoft in Paris offers the shell KATE-CBR as part of their CaseCraft Toolbox, Isoft, also in Paris, has a shell called ReCall. TechInno in Kaiserslautern has S3-Case, a PATDEX-derived tool that is part of their S3 environment for technical systems maintenance.

As an example of functionality, the ReMind shell offers an interactive environment for acquisition of cases, domain vocabulary, indexes and prototypes. The user may define hierarchical relations among attributes and a similarity measure based on them. Indexing is done inductively by building a decision tree and allowing the user to graphically edit the importance of attributes. Several retrieval methods are supported: (1) inductive retrieval matching the most specific prototype in a prototype hierarchy, (2) nearest neighbour retrieval, and (3) SQL-like template retrieval. Case adaptation is based on formulas that adjust values based on retrieved vs. new case differences. ReMind also has the capability of representing causal relationships using a qualitative model. The first commercial products appeared in 1991 including Help-Desk systems, technical diagnosis, classification and prediction, control and monitoring, planning, and design applications. ReMind is a trade mark of Cognitive Systems Inc. and was developed with the DARPA support.

ReCall is a CBR system trademark of ISoft, a Paris based AI company, and applications include (according to ISoft) help desk systems, fault diagnosis, bank loan analysis, control and monitoring. Retrieval methods are a combination of methods (1) and (2) in ReMind, but it offers standard adaptation mechanisms such as vote and analogy, and a library of adaptation methods.

The KATE-CBR tool, named CaseWork, integrates an instance-based CBR approach within a tool for inductive learning of, and problem solving from, decision trees. The inductive and case-based methods can be used separately, or integrated into a single combined method. There are editor facilities to graphically build parts of the case/index structure, and to generate user dialogues. The tool has incorporated initial results on integration of case-based and inductive methods from the INRECA project [Manago-93].

Some academic CBR tools are freely available, e.g. by anonymous ftp, or via contacting the developers.[10]

11. Conclusions and Future trends

Summarizing the paper, we can say that case-based reasoning (CBR) puts forward a paradigmatic way to attack AI issues, namely problem solving, learning, usage of general and specific knowledge, combining different reasoning methods, etc. In particular we have seen that CBR emphasizes problem solving and learning as two sides of the same coin: problem solving uses the results of past learning episodes while problem solving provides the backbone of the experience from which learning advances. The current state of the art in Europe regarding CBR is characterized by a strong influence of the USA ideas and CBR systems, although Europe is catching up and provides a somewhat different approach to CBR, particularly in its many activities related to integration of CBR and other approaches and by its movement toward the development of application-oriented CBR systems.

[10] Protos, for example, is available from the University of Texas, and code for implementing a simple version of dynamic memory, as described in [Riesbeck-89], is available from the Institute of Learning Sciences at Northwestern University.

The development trends of CBR *methods* can be grouped around four main topics: Integration with other learning methods, integration with other reasoning components, incorporation into massive parallel processing, and method advances by focusing on new cognitive aspects. The first trend, integration of other learning methods into CBR, forms part of the current trend in ML research toward *multistrategy learning* systems. This research aims at achieving an integration of different learning methods (for instance case-based learning and induction as is done in the MMA and INRECA systems) into a coherent framework, where each learning method fulfills a specific and distinct role in the system. The second trend, integration of several reasoning methods aims at using the different sources of knowledge in a more thorough, principled way, like what is done in the CASEY system with the use of causal knowledge. This trend emphasizes the increasing importance of knowledge acquisition issues and techniques in the development of knowledge-intensive CBR systems, and the European Workshop on CBR showed a strong European commitment towards the utilization of knowledge level modeling in CBR systems design.

The massive memory parallelism trend applies case-based reasoning to domains suitable for shallow, instance-based retrieval methods on a very large amount of data. This direction may also benefit from integration with neural network methods, as several Japanese projects currently are investigating [Kitano-93]. By the fourth trend, method advances from focusing on the cognitive aspects, what we particularly have in mind is the follow-up of work initiated on creativity (e.g. [Schank-89]) as a new focus for CBR methods. It is not just an 'application type', but a way to view CBR in general, which may have significant impact on our methods.

The trends of CBR *applications* clearly indicates that we will initially see a lot of help desk applications around. This type of systems may open up for a more general coupling of CBR - and AI in general - to information systems. The use of cases for human browsing and decision making is also likely to lead to an increased interest in intelligent computer-aided learning, training, and teaching. The strong role of user interaction, of flexible user control, and the drive towards total interactiveness of systems (of 'situatedness', if you like) favours a case-based approach to intelligent computer assistance, since CBR systems are able to continually learn from, and evolve through, the capturing and retainment of past experiences.

Case-based reasoning has blown a fresh wind and a well justified degree of optimism into AI in general and knowledge based decision support systems in particular. The growing amount of ongoing CBR research - within an AI community that has learned from its previous experiences - has the potential of leading to significant breakthroughs of AI methods and applications.

References

[1] Aamodt, A., (1989) Towards robust expert systems that learn from experience - an architectural framework. In John Boose, Brian Gaines, Jean-Gabriel Ganascia (eds.): *EKAW-89; Third European Knowledge Acquisition for Knowledge-Based Systems Workshop* , Paris, July 1989. pp 311-326.

[2] Aamodt, A. (1991). *A knowledge-intensive approach to problem solving and sustained learning,* Ph.D. dissertation, University of Trondheim, Norwegian Institute of Technology, May 1991. (University Microfilms PUB 92-08460)

[3] Aamodt, A. (1993) Explanation-driven retrieval, reuse, and learning of cases, In *EWCBR-93: First European Workshop on Case-Based Reasoning*. University of Kaiserslautern SEKI Report SR-93-12 (SFB 314) (Kaiserslautern, Germany, 1993) 279-284.

[4] Althoff, K.D (1989). Knowledge acquisition in the domain of CNC machine centers; the MOLTKE approach. In John Boose, Brian Gaines, Jean-Gabriel Ganascia (eds.): *EKAW-89; Third European Workshop on Knowledge-Based Systems,* Paris, July 1989. pp 180-195.

[5] Althoff, K-D. (1992). Machine learning and knowledge acquisition in a computational architecture for fault diagnosis in engineering systems. *Proceedings of the ML-92 Workshop on Computational Architectures for Machine Learning and Knowledge Acquisition*. Aberdeen, Scotland, July 1992.

[6] Anderson, J. R., (1983). *The architecture of cognition*. Harvard University Press, Cambridge.

[7] Aha, D. , Kibler, D. , and Albert, M. K. (1991). Instance-Based Learning Algorithms. *Machine Learning*, vol.6 (1).

[8] Allemange, D. (1993) (Review of EWCBR-93, AICom, this issue)

[9] Bareiss, R. (1989). *Exemplar-based knowledge acquisition: A unified approach to concept representation, classification, and learning.* Boston, Academic Press.

Ashley-90

K. Ashley (1991). *Modeling legal arguments: Reasoning with cases and hypotheticals.* MIT Press, Bradford Books, Cambridge.

Bareiss, 1988

Ray Bareiss: *PROTOS; a unified approach to concept representation, classification and learning*. Ph.D. Dissertation, University of Texas at Austin, Dep. of Computer Sciences 1988. Technical Report AI88-83.

Branting-1991

Karl Branting: Exploiting the complementarity of rules and precedents with reciprocity and fairness. In: *Proceedings from the Case-Based Reasoning Workshop 1991*, Washington DC, May 1991. Sponsored by DARPA. Morgan Kaufmann, 1991. pp 39-50.

Brown-91

Brown, B., Lewis, L. (1991). A case-based reasoning solution to the problem of redundant resolutions of non-conformances in large scale manufacturing. In: R. Smith, C. Scott (eds.): *Innovative Applications for Artificial Intelligence 3*. MIT Press, 1991.

Burstein-89

Burstein, M.H. (1989) Analogy vs. CBR; The purpose of mapping. *Proceedings from the Case-Based Reasoning Workshop*, Pensacola Beach, Florida, May-June 1989. Sponsored by DARPA. Morgan Kaufmann. pp 133-136.

Börner-93

Börner, K. (1993). Structural similarity as guidance in case-based design. In: *First European Workshop on Case-based Reasoning, Posters and Presentations,* 1-5 November 1993. Vol. I. University of Kaiserslautern, pp. 14-19.

Carbonell-86

Carbonell, J. (1986) Derivational analogy; A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (eds.): *Machine Learning - An Artificial Intelligence Approach, Vol.II*, Morgan Kaufmann, pp. 371-392.

Cunningham-93

Padraig Cunningham, Saen Slattery (1993). Knowledge enigneering requirements in derivational analogy. In: *First European Workshop on Case-based Reasoning, Posters and Presentations,* 1-5 November 1993. Vol. I. University of Kaiserslautern, pp. 108-113.

DARPA-89

Proceedings from the Case-Based Reasoning Workshop, Pensacola Beach, Florida, May-June 1989. Sponsored by DARPA. Morgan Kaufmann, 1989.

DARPA-91

Proceedings from the Case-Based Reasoning Workshop, Washington D.C., May 8-10, 1991. Sponsored by DARPA. Morgan Kaufmann, 1989.

EWCBR-93

First European Workshop on Case-based Reasoning, Posters and Presentations, 1-5 November 1993. Vol. I-II. University of Kaiserslautern.

Fabel-93

The FABEL Consortium (1993). *Survey of FABEL.* FABEL Report No. 2, GMD, Sankt Augustin.

Gentner-83

D. Gentner: Structure mapping - a theoretical framework for analogy. *Cognitive Science*, Vol.7. s.155-170. 1983.

Hall-89

R. P. Hall: Computational approaches to analogical reasoning; A comparative analysis. *Artificial Intelligence*, Vol. 39, no. 1, 1989. pp 39-120.

Hammond-89

Kristian J. Hammond: *Case-based planning*. Academic Press. 1989.

Harmon-92

Harmon, P. (1992) Case-based reasoning III, *Intelligent Software Strategies*, VIII (1).

Hennesy-91

Hennessy, D., and Hinkle, D. (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert* 7(5), pp. 21-26.

Hinrichs-92

T.R. Hinrichs: *Problem solving in open worlds*. Lawrence Erlbaum Associates, 1992.

IEEE-92

IEEE Expert 7(5), Special issue on case-based reasoning. October 1992

Keane-88

M. Keane: Where's the Beef? The Absence of Pragmatic Factors in Pragmatic Theories of Analogy In: *Proc. ECAI-88*, pp. 327-332

Kedar-Cabelli-88

S. Kedar-Cabelli: Analogy - from a unified perspective. In: D.H. Helman (ed.), *Analogical reasoning*. Kluwer Academic, 1988. pp 65-103.

Kibler-87

D. Kibler, D. Aha: Learning representative exemplars of concepts; An initial study. *Proceedings of the Fourth International Workshop on Machine Learning*, UC-Irvine, June 1987. pp 24-29.

Kitano-93

H. Kitano: Challenges for massive parallelism. *IJCAI-93, Proceedings of the Thirteenth International Conference on Artificial Intelligence*, Chambery, France, 1993. Morgan Kaufman 1993. pp. 813-834.

Kolodner-83a

Janet Kolodner: Maintaining organization in a dynamic long-term memory. *Cognitive Science*, Vol.7, s.243-280. 1983.

Kolodner-83b

Janet Kolodner: Reconstructive memory, a computer model. *Cognitive Science*, Vol.7, s.281-328. 1983.

Kolodner-88

Kolodner, J. (1988) Retrieving events from case memory: A parallel implementation. In: *Proceedings from the Case-based Reasoning Workshop, DARPA,* Clearwater Beach, 1988, pp. 233-249.

Kolodner-92

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

Kolodner-93

J. Kolodner: *Case-based reasoning*. Morgan Kaufmann, 1993.

Koton-89

Phyllis Koton: *Using experience in learning and problem solving*. Massachusetts Institute of Technology, Laboratory of Computer Science (Ph.D. diss, October 1988). MIT/LCS/TR-441. 1989.

López-90

Beatriz López, Enric Plaza: Case-based learning of strategic knowledge. Centre d'Estudis Avançats de Blanes, CSIC, Report de Recerca GRIAL 90/14. Blanes, Spain, October 1990 (published in Y. Kodratoff (Ed.) *Machine Learning-EWSML-91*, 398-411. Lecture Notes in Computer Science, Springer Verlag).

López-93

B. Lopez, E. Plaza, Case-based planning for medical diagnosis, In: J. Komorowski, Z. W. Ras (Eds.) *Methodologies for Intelligent Systems: 7th International Symposium, ISMIS '93*, p. 96-105. Lecture Notes in Artificial Intelligence 689, Springer Verlag, 1993.

Manago-93

M. Manago, K-D. Althoff, R. Traphöner: Induction and reasoning from cases. In: *ECML - European Conference on Machine Learning, Workshop on Intelligent Learning Architectures.* Vienna, April 1993.

Michalski-92

Michalski, R and Tecuci, G: *Proc. Multistrategy Learning Workshop*. George Mason University, 1992.

Nordbø-92

Nordbø, I., Skalle, P., Sveen, J., Aakvik, G., and Aamodt, A. (1992). *Reuse of experience in drilling - Phase 1 Report*. SINTEF DELAB and NTH, Div. of Petroleum Engineering. STF 40 RA92050 and IPT 12/92/PS/JS. Trondheim.

Oehlmann-92

Oehlmann, R. (1992) Learning causal models by self-questioning and experimentation. *AAAI-92 Workshop on Communicating Sientific and Technical Knowledge.* American Association of Artificial Intelligence.

O'Hara-92

Scott O'Hara, Bipin Indurkhya (1992). Incorporating (re)-interpretation in case-based reasoning. In: *First European Workshop on Case-based Reasoning, Posters and Presentations,* 1-5 November 1993. Vol. I. University of Kaiserslautern, pp. 154-159

Plaza-90

E. Plaza, R. López de Mántaras: A case-based apprentice that learns from fuzzy examples. In Z Ras, M. Zemankova, M. L. Emrich (Eds.) *Methodologies for Intelligent System 5*, . pp 420-427. North Holland, 1990.

Plaza-93

Plaza, E. Arcos J. L., Reflection and Analogy in Memory-based Learning, *Proc. Multistrategy Learning Workshop*., 1993. p. 42-49.

Porter-86

Porter, B. and Bareiss, R. (1986). PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. In: *Proceedings of the First International Meeting on Advances in Learning (IMAL)*, Les Arcs, France, pp. 159-174.

Porter-90

B. Porter, R. Bareiss, Robert Holte: Concept learning and heuristic classification in weak theory domains. *Artificial Intelligence*, vol. 45, no. 1-2, September 1990. pp 229-263.

Richter-1991

A.M. Richter, S. Weiss: Similarity, uncertainty and case-based reasoning in PATDEX. In R.S. Boyer (ed.): *Automated reasoning, essays in honour of Woody Bledsoe*. Kluwer, 1991, pp. 249-265.

Riesbeck-89

C. Riesbeck, R. Schank: *Inside case-based reasoning*. Lawrence Erlbaum, 1989.

Rissland-83

Rissland, E. (1983). Examples in legal reasoning: Legal hypotheticals. In: *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* IJCAI, Karlsruhe.

Ross-89

B.H Ross: Some psychological results on case-based reasoning. *Case-Based Reasoning Workshop* , DARPA 1989. Pensacola Beach. Morgan Kaufmann, 1989. pp. 144-147).

Schank-82

R. Schank: *Dynamic memory; a theory of reminding and learning in computers and people*. Cambridge University Press. 1982.

Schank-89

Roger Schank, David Leake: Creativity and learning in a case-based explainer. *Artificial Intelligence*, Vol. 40, no 1-3, 1989. pp 353-385.

Schult-92

Schult, T. (1992). Werzeuge für fallbaseierte systeme. *Künstliche Intelligenz* 3(92.

Sharma-88

Sharma, S., Sleeman, D (1988) REFINER; a case-based differential diagnosis aide for knowledge acquisition and knowledge refinement. In: *EWSL 88; Proceedings of the Third European Working Session on Learning*, Pitman, 1988. pp 201-210.

Sinoudis-92

Simoudis, E. (1992). Using case-based reasoning for customer techical support. *IEEE Expert* 7(5), pp. 7-13.

Simpson-85

Robert L. Simpson: *A computer model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation*. Technical Report GIT-ICS-85/18, Georgia Institute of Technology. 1985.

Skalak-92

Skalak, C.B, and Rissland, E. (1992). Arguments and cases: An inevitable twining. *Artificial Intelligence and Law, An International Journal*, 1(1), pp.3-48.

Slade-91

Slade, S. (1991). Case-based reasoning: A research paradigm. *AI Magazine* Spring 1991, pp. 42-55.

Smith-81

E. Smith, D. Medin: *Categories and concepts*. Harvard University Press. 1981.

Stanfill-88

Craig Stanfill, David Waltz: The memory based reasoning paradigm. In: *Case based reasoning. Proceedings from a workshop*, Clearwater Beach, Florida, May 1988. Morgan Kaufmann Publ. pp.414-424.

Steels-90

L. Steels, Components of expertise, *AI Magazine,* 11 (2) (Summer 1990) 29-49.

Steels-93

L. Steels, The componential framework and its role in reusability, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems* (Spinger, 1993) 273-298.

Strube-90

Strube, G. and Janetzko, D. (1990). Episodishes Wissen und Fallbasierte Schliessen: Aufgabe fur die Wissendsdiagnostik und die Wissenspsychologie. *Schweizerische Zeitschrift fur Psychologie*, 49, 211-221.

Strube-91

G. Strube, The role of cognititve science in knowledge engineering, In: F. Schmalhofer, G. Strube (eds.), *Contemporary knowledge engineering and cognition: First joint workshop, proceedings*, (Springer 1991) 161-174.

Sycara-88

Sycara, K. (1988). Using case-based reasoning for plan adaptation and repair. *Proceedings Case-Based Reasoning Workshop, DARPA.* Clearwater Beach, Florida. Morgan Kaufmann, pp. 425-434.

Tulving-72

Tulving, E. (1977). Episodic and semantic memory. In E. Tulving and W. Donaldson: *Organization of memory*, Academic Press, 1972. pp. 381-403.

Veloso--93

Veloso, M.M., Carbonell, J. (1993). Derivational analogy in PRODIGY. In *Machine Learning* 10(3), pp. 249-278.

Venkatamaran-93

Venkatamaran, S., Krishnan, R. and Rao, K.K. (1993) A rule-rule-case based system for image analysis. In: *First European Workshop on Case-based Reasoning, Posters and Presentations,* 1-5 November 1993. Vol. II. University of Kaiserslautern, pp. 410-415.

Wittgestein-53

Wittgenstein, L. (1953) *Philosophical investigations*. Blackwell, pp. 31-34.