The Case for Structure-based Representations

Kathryn E. Sanders, Brian P. Kettler, and James A. Hendler *

Department of Computer Science
University of Maryland
College Park, MD 20742
{kettler,sanders,hendler}@cs.umd.edu
Fax: (301) 405-6707

Abstract

Case-based reasoning involves reasoning from *cases*: specific pieces of experience, the reasoner's or another's, that can be used to solve problems. As a result, case representation is critical: an incomplete case representation limits the system's reasoning power. In this paper we argue for *structure-based* case representations, which express arbitrary relations among objects in a flexible way, over more limited or inflexible methods. We motivate the distinction between these kinds of representations with examples from information retrieval systems, CBR systems, and computational models of human analogical reasoning. Structure-based representations provide the benefits of greater expressivity and economy. We give examples of these benefits from two case-based planning systems we have developed, CaPER and CHIRON, and show how the case matching and case acquisition costs can be reduced through the use of massively parallel techniques.

This paper is being submitted as a scientific paper.

Keywords: case memory structures; knowledge representation.

*Submitted to the 1995 International Conference on Case-based Reasoning. This work has benefited from the comments of Bill Anderson, Karl Branting, Sean Luke, and Robert McCartney. Research by B. Kettler and J. Hendler was supported in part by grants from NSF (IRI-8907890), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039 and by ARI (MDA-903-92-R-0035, subcontract through Microelectronics and Design, Inc.). Research by K. Sanders was supported in part by grants from NSF PYI Award (IRI-8957601) to Thomas Dean, AFOSR and ARPA (F30602-91-C-0041), ONR (N00014-91-J-4052), ARPA Order 8225, NSF and ARPA (IRI-8905436), IBM (17290066, 17291066, 17292066, 17293066), and by NSF (IRI-8801253).

1 Introduction

Case-based reasoning (CBR) involves reasoning from *cases*: specific pieces of experience, the reasoner's or another's, that can be used to solve problems. As a result, case representation is critical. A system's case representation must support its operations: indexing, retrieval, and comparison or adaptation. An incomplete case representation limits the system's reasoning power.

Efficiency pushes systems in the direction of simpler representations. An extreme example of this is found in information-retrieval (IR) systems. Like CBR systems, IR systems index and retrieve information, but without adaptation or further processing. Most IR systems have used very simple representations: essentially, each document is represented by the set of words it contains. Representing a document as a list of words is not necessarily trivial; but it is relatively easy to automate, an important factor given that commercial IR systems such as Dialog, Lexis, and Nexis incorporate hundreds of thousands of documents.

In this paper, we adopt the term "feature-based," used to describe these simple IR representations [4], and use it to denote any representation that expresses facts about individual objects in a case without relating them to each other. Such representations may be implemented using a set of attribute-value pairs, a feature vector, or frames with atomic slot-fillers.

We use the term "structure-based" (also drawn from information-retrieval [4]) for representations that express arbitrary relations among objects in a flexible way. Structure-based representations are often implemented as graph structures such as semantic networks or as lists of concrete propositions in some logic. Whether the representation is implemented as a list of propositions or as a graph is immaterial, since there is a simple translation from propositional to graph representation [23]. What is important is that the topology of the graph corresponding to the case representation varies from case to case.

Balancing the need for an expressive representation with efficiency considerations, most CBR systems have used representations that fall between feature-based and structure-based. Feature-based representations cannot express the information needed by CBR systems in complex real-world domains. Several of these hybrids combine structure-based representations with feature-based indexing (e.g., [11]). Others include some relational information but use the same structure for each case (e.g., [3]).

We believe that pure structure-based representations offer significant advantages, and thus we are investigating ways to implement such representations efficiently. We make a "case-based argument" using examples from two systems, CHIRON and CaPER, to show the benefits of a structure-based representation, particularly in case-based planning, and to illustrate approaches to overcoming its costs. Other systems that have used structure-based representations include PLEXUS [1], GREBE [5], ARCS [25], and SME [8], and COOKIE [20]. These systems are described briefly in Section 5.

2 Overview of CaPER and CHIRON

This section gives a brief overview of CHIRON and CaPER. Details can be found in [24] and [15].

CHIRON is a hybrid rule-based and case-based system in the domain of tax planning. It uses rules and structured cases to solve a cluster of problems having to do with buying, selling, renting, and owning residential housing.

CHIRON's knowledge base includes representations of part of the United States Internal Revenue Code and approximately twenty-four cases under various provisions of that statute. It also includes safe harbor plans, or prototypes, that satisfy the rules; a representation of the relationship between the rules, prototypes, and cases; and finally, a representation of the input description of the taxpayer's goals and current situation. The facts of both previous cases and the current situation are represented as lists of propositions in a temporal modal logic.

CHIRON's case-based reasoner takes partial plans generated by the hierarchical planner, refines them, and generates arguments in support of the resulting plan. Given a partial plan, the case-based planner first retrieves a prototype for that plan and adapts it along directions suggested by previous cases, to the extent necessary to fit the current situation. It then retrieves all the previous cases that share any fact with the resulting plan. Next, it computes a mapping between the facts of each case and the facts of the current situation, in order to determine the overlap between the two, and sorts the cases by inserting them into a HYPO-style case lattice [3]. It uses the case lattice to determine whether there is sufficient support for the plan being considered (generally, whether the plan falls between the prototype and previous successful cases of a given strategy), and if so, also uses the lattice to generate HYPO-style arguments for and against the success of a plan.

CaPER is a case-based planning system that makes use of massive parallelism to access a large casebase (currently several hundred cases) [17, 15]. CaPER uses structure-based representations for cases and conceptual knowledge, implemented as a single semantic network. Given the availability of fast parallel case retrieval methods, memory does not have to be pre-indexed for efficiency and thus can be accessed flexibly and often.[1] Retrieval is flexible because any feature of the target problem description can be included in the retrieval probe, a graph to be matched against the subgraphs of the semantic net. A case can be retrieved via any of its constituent nodes. Domain knowledge and planning techinques are used to form probes at case retrieval time. Multiple plans (or subplans) can be retrieved and merged into a target plan.

CaPER's semantic network memory is implemented using Parka, a massively parallel frame-based knowledge representation system that runs on the Connection Machine (CM-2 and CM-5) and provides very fast inferencing mechanisms [7]. A case includes the goals, initial situation, and plan/subplan hierarchy for a planning problem. Plan validation structures based on those used in the PRIAR system [14] are also stored. These capture interdependencies among plan actions and are used by CaPER to detect interactions that arise during plan adaptation and plan merging. CaPER is being tested in our transport logistics planning domain.

3 Benefits of Structure-Based Representations

Structure-based representations allow a system to capture a reasonably complete description of a case. By "reasonably complete," we mean a representation that includes all the information about a case that is likely to be useful. As argued in [22], in order to maximize their usefulness, case representations should be as complete as possible. Complete representations increase a system's reasoning power: a system cannot reason with information that it does not have.

In particular, a structure-based representation makes it possible to express the relations between objects. Consider the graph shown in Figure 1. This graph corresponds to part of the representation of *Hughston v. Commissioner*, one of the cases in CHIRON's casebase. The nodes in the graph correspond to objects in CHIRON's logical representation; edges correspond to binary predicates; and the endpoints of an edge correspond to the predicate's parameters. For simplicity, the node labels are not shown in Figure 1, but every node has one or more such labels. Node labels correspond to unary predicates on a given object. For example, Hughston is a lawyer; house1 is both a house and real property; Shell-Oil-Company is a corporation; and so forth. For the actual representation and the original text of the case, see [24].

In this case, the taxpayer, Hughston, was a lawyer for Shell Oil, in Texas.

[1] We use "indexing" to mean the use of pointers for case retrieval, rather than a more recent, broader interpretation of "indexing" that includes any domain knowledge used in the retrieval of cases (e.g., [18]).

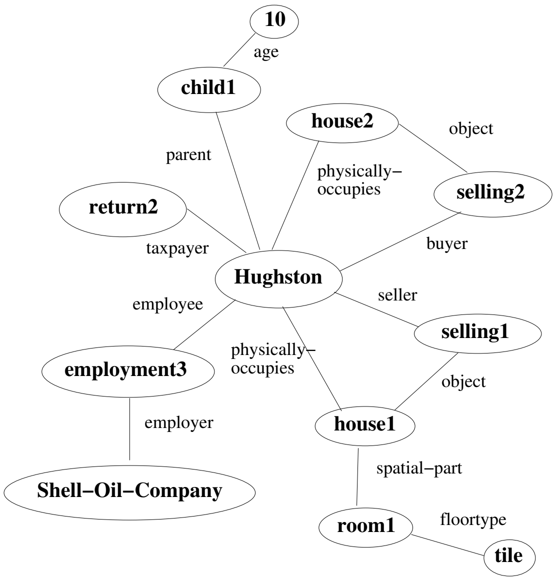

Figure 1: Part of CHIRON's representation of *Hughston v. Commissioner.*

When he was transferred to a new location in Texas, he sold his old house (which had a bathroom with a tile floor) and bought another, closer to his new place of work. He had three children, of whom the oldest was ten years old. At issue was the question of whether he would have to pay taxes on the profit made on the first house. The relations between the objects in this case are important: the fact that the taxpayer is selling one house and buying another, for example; the facts (represented in the case, but not shown here), that the two houses are far apart, but the second one is close to the taxpayer's new place of employment; and so forth. These could not be expressed in a simple feature vector.

In CaPER, structure-based representations provide the required expressivity to represent relations between objects and relations between plan actions. In our transport logistics domain the configuration of packages with respect to vehicles in a problem's initial situation is important in retrieving a case to adapt to the target problem. Consider the graph based on a piece of a case from CaPER's transport logistics domain, shown in Figure 2. This case involves a plan to deliver a package, Pkg99, starting at 3 p.m. on March 15, 1994, in a truck (Truck22) whose original location is Boston. To represent information such as the fact that Truck22 and Pkg99 have the same origin, structure-based representations are required. To match cases on such relations, "structure" queries that contain two or more variables are needed: e.g., "find all plans in whose initial situation $\texttt{inst-of}(t,\texttt{Truck}) \wedge \texttt{inst-of}(p,\texttt{Package}) \wedge \texttt{origin}(t,x) \wedge \texttt{origin}(p,x)$." CaPER also needs to represent and match on plan validations – relations among plan actions.

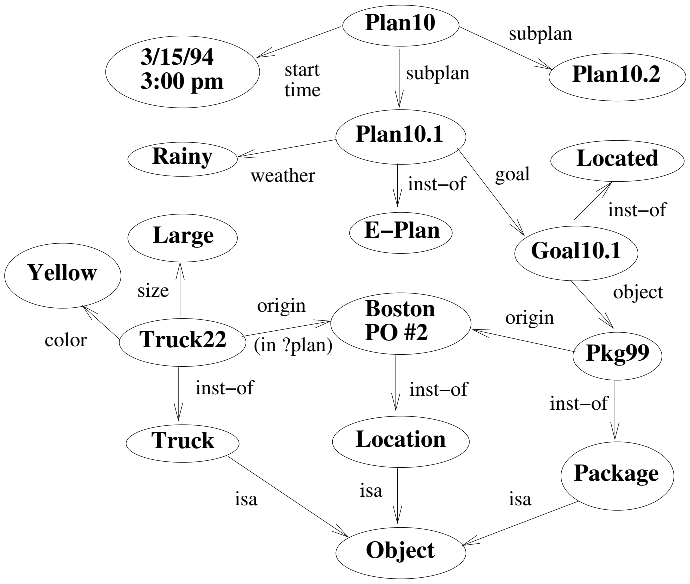

Figure 2: Sample Piece of CaPER's Memory for Transportation Logistics Domain. ("inst-of" links are "Instance Of" ($\in$). "isa" links are $\subset$ relations.)

CHIRON's structure-based representation permits the system to capture the detailed facts of its cases, and the system's algorithms for indexing, adapting, comparing, and contrasting cases exploit this rich representation. For example, every case is indexed under each of the predicates used in its case representation; thus, the system can identify as many partial matches as its rich representation language can express. Every fact can form the basis for an argument. A fact shared with a previous successful case supports an argument that the taxpayer in the current situation should succeed; a fact shared with a previous unsuccessful case suggests a possible problem. Similarly, unshared facts suggest possible distinctions between the current situation and previous cases. Finally, any fact in one of the previous cases can potentially be added to the prototype to make a new plan. Similarly, facts can be subtracted from the prototype, and their parameters can be varied.

In general, a complete case description is desirable for several reasons. First, it is not always possible at case-representation time to be sure of the future use of a case. A legal case may be used as the basis for analyzing a future situation, a plan, or a prediction of issues that might arise as the result of execution of a plan. A plan for transporting a package may be used for constructing a new transport plan or for recognizing the plan of another agent.

Second, even if a case is reused for the same purpose, it is not always clear which facts of the original case were relevant to the result. For example, a taxpayer trying to avoid the loss sustained by the taxpayer in *Hughston* might succeed if he is not a lawyer; if he buys a house that is exactly comparable to the house sold; or simply if he takes advantage of a different provision of the statute. In CaPER, we can express the fact that a delivery was made to or through a particular city (e.g., Boston, in Figure 2). Cities have idiosyncratic restrictions on the types of packages and deliveries that pass through them.

Finally, each new case has unpredictable idiosyncratic facts. Designers of hybrid systems that use a fixed template for cases must either decide on the template after examining all the cases to be included in their casebase, omit information from new cases, or revise the representation for all previous cases whenever a new useful fact is encountered.

Even if all the important facts are predictable, hybrid systems using a fixed structure for cases must choose between completeness and economy of representation. The important facts may vary widely from one case to the next. If a critical fact occurs in one case out of a thousand in the casebase, it would be wasteful to provide a slot for it in every single case representation.

For example, CHIRON currently has seventy-five distinct predicates. CaPER has 204. New ones are added with every new case represented. Many occur only in one or two cases, and no case includes them all. In CHIRON, for example, he fact that one taxpayer is a war veteran, or another is a lawyer, or a third is selling rights to a rent-controlled apartment rather than a house, are all potentially significant. If the system had a fixed representation, it would be necessary to choose between omitting these facts and wasting space in every case represented. A structure-based representation allows both systems to represent facts in the cases where they occur, without alloting space for them in cases where they are absent.

4 Overcoming the costs

The benefits of structure-based representations described in the previous section come at the cost of increased computational complexity of matching and increased case acquisition effort. The approaches taken to reducing these costs in CaPER, CHIRON, and several other systems are described in the following sections.

4.1 Matching

The matching operation puts two cases in correspondence and is used for case retrieval and comparison. For feature-based representations (feature vectors and frames with atomic slot-fillers), this operation is linear in the number of features. For hybrid representations with fixed structures, it is also linear.

Using a structure-based representation increases the computational cost of matching. If cases were represented as unlabelled graphs, matching would be the subgraph isomorphism problem, which is NP-complete [9]. If the graphs had node labels and no node label ever occurred more than once in a given case, matching would be linear in the sum of the number of nodes and the number of edges. In practice, the complexity of matching cases with a structure-based representation lies somewhere between these two extremes.

In CHIRON, for example, the matching algorithm works as follows. First, the system retrieves each case that shares any predicate (or in graph terms, any node or edge label) with the current plan. Then, for each node in the description of the current situation, the system identifies the nodes in the previous case that could match that node. In the case shown in Figure 1, for example, there are two houses; so if the current situation involves a house, the current house could match either of these. Finally, the system considers each of the possible mappings permitted by that list and returns the one that causes the most edges to match.

In practice, the average case has forty-four facts, and there are often a couple of "sellings," or two or three "objects," or a couple of "houses." There are between zero and three mappings for each node. Altogether, there are probably no more than fifteen possible mappings for any case, and the search could be pruned so that not all of these are examined (as done in GREBE [5]. For a small casebase, in a domain where instantaneous response time is not required, CHIRON's response times (typically three to five minutes per problem) are tolerable. Still, the time required for matching is significant, and increases linearly with the size of the casebase.

Some serial systems with structure-based representations have used indexing to reduce the cost of matching. Indexing restricts the search for relevant cases to a subset of the casebase. Feature-based techniques are often employed such as the construction of a discrimination network using selected case features (attributes). For example, CHEF indexes cooking plans under their main ingredients and cooking failures under sets of causally relevant features [11]. Similarly, CHASER indexes tort cases under features such as the harm caused and possible legal defenses [6]. The main disadvantage of indexing is that it hinders flexibility at case retrieval time. Cases that share unindexed features with the target problem will not be retrieved. (See discussion in [22, 17, 15]).

A few systems have used parallel techniques to reduce the cost of matching. PARADYME, for example, is a massively parallel frame system that has been used to implement a memory for a CBR system [19].[2] In CaPER, the massively parallel mechanisms of the Parka Structure Matcher are used to match a probe graph to CaPER's semantic network memory [2].[3]

To evaluate the average-case behavior of matching a (labelled) probe graph to a subgraph of CaPER's (labelled) memory graph, we have done some preliminary experiments. We generated representative retrieval probes for a transport logistics domain and recorded the time taken to process them by the Parka Structure Matcher. The purpose of these experiments was to assess the absolute parallel retrieval times and the scalability of the parallel methods used, for a variety of representative retrieval probes (generated by hand) and casebase sizes. The probes were matched to an *unindexed* memory using Parka on a CM-2 and a serial version of Parka and the results are shown in figure 3.[4]

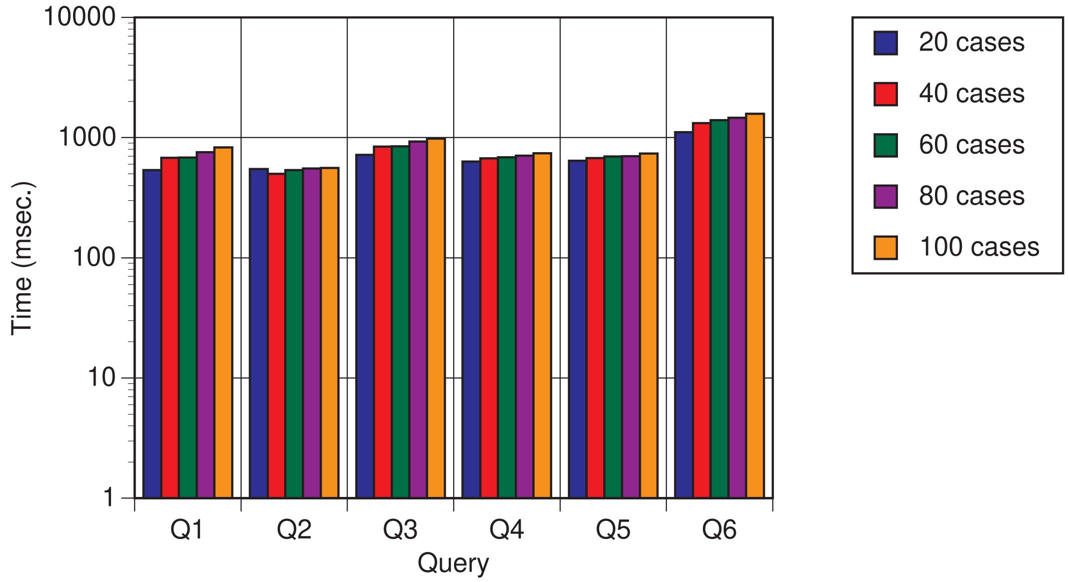

Figure 3: Parallel timings for sample queries on transport logistics casebases of varying size (log scale).

These results, detailed in [17], show parallel retrieval times of about a second, even for a 100 case memory which contained 1213 (sub)plans (8616 frames). Not only are the absolute parallel retrieval times low but they scale well to large casebases: the retrieval times appear to grow better than logarithmically

[2]For a comparison of PARADYME to Parka, see [7].

[3]A sample memory graph is shown in Figure 2. The probe graph contains nodes that are constants or variables and links that are constants (predicates).

[4]The CM-2 with 32K virtual (16K real) processors. An alternate, *optimized serial* implementation on an *indexed* casebase would have $O(\log n)$ performance if a balanced discrimination tree were used. Of course such a system would have the disadvantages of pre-indexing previously mentioned.

in the number of cases (frames). The complexity of the underlying subgraph-matching algorithm on actual casebases is much less than the worst case complexity (exponential in the number of binary constraints). For a detailed analysis of the algorithm, see [2].

4.2 Case Acquisition

The cost of case acquisition can be high, particularly if a CBR system uses many cases. Case acquisition includes the costs of representing and indexing individual cases. The time cost of acquiring cases can be lower in systems using feature-based representations. In IR systems, for example, a document is represented and indexed by the words it contains and does not have to be hand-encoded nor hand-indexed: thus case acquisition can be fully automated.

In most CBR systems, the initial cases are derived from the experience of the system designer or some other expert, rather than from a document. As a result, a simple automated translation is not possible. The need to encode a case by hand makes case acquisition more costly.

Some hybrid systems solve this problem by using the same structure for each case. For example, for each case a person might fill a frame with fixed slots or create a feature vector for predesignated features. Representing cases using a fixed template is harder than translating a document into a set of words, but it is still fairly easy: having chosen a frame representation, the system designer can use the representation as a checklist when entering cases. There is no need to analyze each case separately, and different cases are likely to be treated consistently.

In systems that use pure structure-based representations, a structure needs to be created for each case during case representation. For example, in CHIRON, the facts of a case are hand-encoded into a graph representation. Since the facts can vary widely from case to case, representing a case is more complicated than merely filling in fixed slots in a frame.

Automated methods can be used to create structure-based representations. CaPER uses a generative planner, UM Nonlin, to seed the initial casebase. Plans created by UM Nonlin on randomly generated problems are automatically converted to cases in CaPER's semantic net representation. As cases are not indexed, the initial casebase acquisition process is completely automated.

Natural language processing methods have also been employed to automatically generate cases. For example, COOKIE's RAS module scanned in recipes from a cookbook in English and converted them to case representations [21].

5 Related Work

Other CBR systems that have used structure-based representations include PLEXUS, GREBE, and COOKIE. ARCS and SME, two cognitive models of human analogical reasoning, also used structure-based representations.

PLEXUS is an adaptive planning system in a commonsense reasoning domain that uses case-based methods to adapt parts of a previous plan to a new situation on the fly [1]. PLEXUS uses a structure-based representation (a semantic net) and can ascend (descend) abstraction links to generalize (specialize) actions in a plan being adapted. PLEXUS interleaves plan adaptation and execution.

GREBE is a case-based legal analysis system in the domain of workers' compensation. It uses a structure-based representation (a semantic net). As a serial structure-based system, it faced the same issues of matching and case acquisition as CHIRON, but in the context of analysis rather than planning [5].

COOKIE is an integrated case-based planning and execution system in the meal-preparation domain that examined the use of highly detailed case descriptions. A case in COOKIE is a list of propositions in a temporal logic of actions and facts that represents meal-preparation episodes. Issues faced included how to retrieve multiple partially-matching cases that could be adapted and combined into a single solution [20].

SME is a system for analogue mapping [8]. SME has been used to test the structure-matching theory, which claims humans use structural similarity (isomorphism) on structure-based representations when mapping between analogues [10]. Humans are able to find structural similarities between two analogues which are dissimilar on the surface: e.g., in proportional analogies such as 3:6 :: 2:4. Feature-based models of similarity are not able to account for analogies such as these [10]. SME can produce mappings found by humans.

ARCS is a system for analogue retrieval built to test a theory that humans use the constraints of semantic (surface) similarity, structural isomorphism, and pragmatic centrality (the purpose of the analogy, goals of the reasoner, etc.) when retrieving analogues [25, 12]. ARCS uses parallel constraint satisfaction techniques on structure-based representations. Its performance compares favorably with that of humans, and ARCS is better able to use structural similarity at retrieval time and thus can find relevant cases that a human might not. **ACME** is a system for analogue mapping that is very similar to ARCS [13].

6 Conclusions

In this paper we have argued for *structure-based* case representations, which express arbitrary relations among objects in a flexible way, over more limited or inflexible methods, illustrating the distinction between these kinds of representations with examples of information retrieval systems, CBR systems, and computational models of human analogical reasoning. Structure-based representations provide the benefits of greater expressivity and economy. We have given examples of these benefits from two case-based planning systems, CaPER and CHIRON, and demonstrated how the case matching and case acquisition costs can be reduced through the use of massively parallel techniques.

We are currently working on extending and improving the parallel techniques used for retrieval and matching of structure-based cases. We are building a thousand-case casebase (approx. 100K frames) for the transport logistics domain and a casebase of CHIRON's legal cases translated into Parka, and we plan further empirical evaluation (see [16] for details).

References

[1] R. Alterman. Adaptive planning. *Cognitive Science*, 12:393–421, 1988.

[2] W. A. Andersen, J. A. Hendler, M. P. Evett, and B. P. Kettler. Massively parallel matching of knowledge structures. In H. Kitano and J. Hendler, editors, *Massively Parallel Artificial Intelligence*, pages 52–73. AAAI Press/The MIT Press, Menlo Park, California, 1994.

[3] K. D. Ashley. Modelling legal argument: reasoning with cases and hypotheticals. Technical Report 88-01, University of Massachusetts, Amherst, Department of Computer and Information Science, 1988. (PhD Thesis).

[4] N. J. Belkin and W. B. Croft. Retrieval techniques. *Annual Review of Information Science and Technology*, 22:109–145, 1987.

[5] L. K. Branting. Integrating rules and precedents for classification and explanation: automating legal analysis. Technical Report AI90-146, Artificial Intelligence Laboratory, Department of Computer Sciences, University of Texas at Austin, 1990. (PhD Thesis).

[6] B. Cuthill and R. McCartney. Issue spotting in legal cases. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law,* Amsterdam, pages 245–253, 1993.

[7] M. P. Evett, J. A. Hendler, and L. Spector. Parallel knowledge representation on the Connection Machine. *Journal of Parallel and Distributed Computing*, 22:168–184, 1994.

[8] B. Falkenhainer, K. D. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1990.

[9] M. R. Garey and D. S. Johnson. *Computers and intractability.* W. H. Freeman and Company, New York, 1979.

[10] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7:155–170, 1983.

[11] K. J. Hammond. Case-based planning: A framework for planning from experience. *Cognitive Science*, 14:384–443, 1990.

[12] K. J. Holyoak and P. Thagard. *Mental Leaps: Analogy in Creative Thought.* The MIT Press, Cambridge, Mass., 1995.

[13] K. J. Holyoak and P. R. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, pages 295–355, 1989.

[14] S. Kambhampati and J. A. Hendler. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence*, 55:193–258, 1992.

[15] B. P. Kettler. *Case-based Planning with a Massively Parallel Memory.* Doctoral dissertation, University of Maryland at College Park, Dept. of Computer Science, 1995. In preparation.

[16] B. P. Kettler and J. A. Hendler. Evaluating a case-based planning system. In D. W. Aha, editor, *Case-based Reasoning Workshop, Twelfth National Conference on Artificial Intelligence*, pages 157–163. AAAI, unpublished, 1994.

[17] B. P. Kettler, J. A. Hendler, W. A. Andersen, and M. P. Evett. Massively parallel support for case-based planning. *IEEE Expert*, pages 8–14, Feb. 1994.

[18] J. L. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, San Mateo, California, 1993.

[19] J. L. Kolodner and R. Thau. Design and implementation of a case memory. Technical Report RL88-1, Thinking Machines Corporation, Aug. 1988.

[20] R. McCartney. Episodic cases and real-time performance in a case-based planning system. *Expert Systems with Applications*, 6:9–22, 1993.

[21] R. McCartney, B. Moreland, and M. Pukinskis. Case acquisition from plain text: reading recipes from a cookbook. In *Proceedings of the first international conference on information and knowledge management*, Baltimore, MD, November 1992.

[22] R. McCartney and K. E. Sanders. The case for cases: a call for purity in case-based reasoning. In *Proceedings of the AAAI Symposium on Case-based Reasoning*, pages 12–16, 1990.

[23] N. J. Nilsson. *Principles of Artificial Intelligence.* Morgan Kaufman, San Mateo, California, 1980.

[24] K. E. Sanders. Chiron: planning in an open-textured domain. Technical Report 94-38, Computer Science Department, Brown University, 1994. (PhD Thesis).

[25] P. R. Thagard, K. J. Holyoak, C. Nelson, and D. Gochfeld. Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46:259–310, 1990.