Case-Based Reasoning Integrations

Cynthia Marling, Mohammed Sqalli, Edwina Rissland, Hector Muñoz-Avila, and David Aha

■ This article presents an overview and survey of current work in case-based reasoning (CBR) integrations. There has been a recent upsurge in the integration of CBR with other reasoning modalities and computing paradigms, especially rule-based reasoning (RBR) and constraint-satisfaction problem (CSP) solving. CBR integrations with model-based reasoning (MBR), genetic algorithms, and information retrieval are also discussed. This article characterizes the types of multimodal reasoning integrations where CBR can play a role, identifies the types of roles that CBR components can fulfill, and provides examples of integrated CBR systems. Past progress, current trends, and issues for future research are discussed.

Case-based reasoning (CBR) is an AI paradigm that can be synergistically combined with other approaches to facilitate a broad array of tasks (Aha and Daniels 1998). This article presents a brief introduction to CBR, a review of other approaches with which CBR has been combined, an overview of tasks CBR integrations can perform, a discussion of open issues in CBR integration, and a look at synergies achieved through CBR integration. It characterizes the types of multimodal reasoning integrations where CBR can play a role, identifies the types of roles that CBR components can fulfill, and provides examples of integrated CBR systems.

CBR is the process of using past cases to interpret or solve new problem cases (Kolodner 1993; Riesbeck and Schank 1989). Excellent extended introductions to CBR are available (Aamodt and Plaza 1994; Kolodner and Leake 1996); therefore, this overview is brief. Central to CBR is the determination of which past cases are similar enough to the new case to warrant consideration for reuse in the new problem. There are two basic kinds of CBR: (1) interpretive and (2) problem solving. *Interpretive CBR* involves use of past cases, typically called *precedents,* to create an analysis and justification for the interpretation of a new case. *Problem-solving CBR* involves adapting a solution to a past problem, typically a design or plan, to meet the requirements of the new situation. A lawyer arguing a case, for example, would use interpretive CBR, but an engineer configuring a motor would use problem-solving CBR. Problem-solving CBR can be further subdivided into transformational and derivational CBR. In *transformational CBR,* past solutions are used directly. In *derivational CBR*, the problem-solving processes by which solutions are derived are reused.

All CBR follows the same basic steps: (1) analyze the new case (or problem); (2) based on this analysis, retrieve relevant past cases from a case base; (3) based on a "similarity metric," rank retrieved cases according to how relevant or useful they are with respect to the new case, and select one or more "best" cases to use in solving the new case; (4) create a solution to the new case; (5) test and explore the proposed solution; and (6) if appropriate, add the new case and its solution to the case base and index it so that it can be retrieved for future use.

Critical ingredients in CBR are means to index and retrieve cases from the case base, methods to assess similarity, methods to compare and contrast cases, and methods to create solutions. Solution methods in the two basic types of CBR are quite different: In interpretive CBR, the methods produce interpretations with supporting rationales, and in problem-solving CBR, new plans or designs are produced. Interpretive CBR relies heavily on methods to reason analogically with similarities and differences between cases. Good examples of interpretive CBR can be found in the law. The HYPO system was one of the earliest and best examples of an interpretive CBR system (Ashley 1990; Rissland, Valcarce, and Ashley 1984). HYPO modeled how lawyers argue cases involving trade secrets law by presenting points and counterpoints based on cases previously settled in courts of law. Problem-solving CBR relies on methods to adapt cases and assess how modifications impact the overall solution (for example, does satisfying a new goal undo satisfaction of an old one). One of the earliest and best examples of problem-solving CBR was CHEF (Hammond 1989, 1986). CHEF was a planner, which created new plans through adaptation of old ones. A plan in CHEF was a recipe for preparing a complex dish.

Note that a system that only analyzes and retrieves cases is not a complete CBR system. However, in many hybrid systems, it is exactly the early steps of CBR—analysis, retrieval, ranking, selection—that are often used in conjunction with another reasoning method. Other, more complex, hybrid systems use full-scale CBR. One might distinguish between these systems with the use of a small or a big "r"; that is, CBr or CBR. In this article, the primary focus is on CBr (small r) hybrids.

Although CBR is a powerful paradigm in its own right, there are strong motivations for integrating it with other reasoning modalities and computing techniques, including rule-based reasoning (RBR), constraint-satisfaction problem (CSP) solving, model-based reasoning (MBR), genetic algorithms, and information retrieval. Researchers have reported that integrated approaches have enabled them to more accurately model the knowledge available in a problem domain, compensate for the lack of a predictive or complete model of a problem domain, simplify the knowledge-acquisition process, improve solution quality, improve run-time efficiency, leverage past problem-solving experiences, and compensate for the shortcomings of one approach by capitalizing on the strengths of another (Aha and Daniels 1998; Rissland and Skalak 1991).

In summary, there are many reasons for integrating CBR with other reasoning modalities and many ways to do so. In the next section, different approaches to CBR integration are presented. Then, tasks that have benefited from CBR integration are described, including interpretation and argumentation, design and synthesis, planning, and the management of long-term medical conditions. A discussion of open issues in CBR integration follows. The article closes with a summary and conclusions.

Case-Based–Reasoning Integration Techniques

In this section, approaches and techniques for integrating CBR with other reasoning modalities and computing paradigms are presented.

Case-Based Reasoning and Rule-Based Reasoning

RBR was the first modality to be successfully integrated with CBR. RBR is the reasoning approach behind classical expert systems, including MYCIN, which diagnosed infectious blood diseases; XCON, which configured VAX computers for the Digital Equipment Corporation; and PROSPECTOR, which determined where to search for mineral deposits (Hayes-Roth, Waterman, and Lenat 1983). An RBR system contains a rule base, or a set of if-then rules; an inference engine, or a means of applying the rules to solve problems; a working memory, or a means of maintaining the current problem state; and frequently, an explanation facility, or a means of showing a user the sequence of rules that led to a conclusion.

Each rule represents a small piece of knowledge that can be combined, or chained together, with other rules to infer conclusions or derive solutions to problems. RBR differs fundamentally from CBR in both knowledge representation and knowledge utilization. Some important distinctions noted in Kolodner (1993) are that cases represent larger chunks of knowledge than rules, and they compose this knowledge in advance of run-time inferencing; rules represent patterns, but cases are constants; and rules are applied when they match a situation exactly, whereas cases can be used in partially matching, or similar, situations.

Some rationales for integrating CBR with RBR are that problem domains might naturally contain both rules and cases; human reasoners might naturally use both rules and cases; the rules of a domain might be vague, or open textured, and therefore require interpretation; rules can have exceptions that must be handled; rules can enable evaluation of solutions without supporting initial generation of solutions; and cases can be few or difficult to acquire in a domain.

The earliest CBR-RBR hybrids were built in statutory legal domains. These domains naturally include rules, or statutes, and cases, or legal precedents. A famous hypothetical rule from legal philosophy illustrates the problems of statutory interpretation (Hart 1958). Consider the rule "No vehicles in the park." Part of the problem in administering this rule is determining what counts as a vehicle. Clearly, a car or a bus is a vehicle, but is a motorized bike a vehicle? A motorized wheelchair? A nonmotorized wheelchair? Further, how ironclad is this rule: Does it really mean *never*, or are there some situations when a vehicle is allowed? For example, how should the rule be applied to fire trucks, ambulances, or park maintenance trucks? Does the rule really mean "No vehicles, except for emergency vehicles or park maintenance vehicles, are allowed in the park"? Should the rule always permit such vehicles access? What about a park maintenance truck that only drives in the park so its crew can have a picnic lunch? No matter how carefully crafted such rules are, there will always be interpretive questions about their meaning and scope and their constituent terms. The law deals with these problems by using cases in conjunction with rules. Cases can stand for implicit updates to the rules, for example, by registering unspoken exceptions (emergency vehicles) and requirements (on official business). Cases allow a rule-based regime to adapt to its changing environment without necessarily requiring the rules to be rewritten at every new turn of events.

Good examples of hybrid CBR-RBR legal systems are CABARET (Rissland and Skalak 1991), GREBE (Branting 1991), and IKBALS (Zeleznikow, Vossos, and Hunter 1994). The CABARET system operates in the area of U.S. tax law concerning the home-office deduction. It integrates CBR and RBR—tax cases and tax regulations—by using a rule-based agenda mechanism. Based on the status of tasks in progress by CBR and RBR modules, heuristic rules are used to post and prioritize tasks on an agenda of tasks. This seminal system is described in more detail later. GREBE operates in the area of Texas employment law concerning worker disability for injuries sustained during the course of employment. It uses CBR and RBR to determine and justify the legal conclusions of its cases. Unlike CABARET, it does not use an agenda to mediate between strategies. Rather, it always tries both strategies to find the best justification for a conclusion. IKBALS works in two Australian legal domains: (1) worker disability and (2) lending by financial institutions. RBR is its dominant reasoning mode. CBr is used to retrieve appropriate cases for review by lawyers when rules proved inadequate to derive a solution. IKBALS also integrates information retrieval, giving users access to sources such as legal treatises in addition to statutes and cases.

Another early system that capitalized on CBR-RBR synergy is ANAPRON (Golding and Rosenbloom 1991). ANAPRON is a speech synthesizer that pronounces American surnames aloud. Because American surnames originate in many different languages and become Americanized in many different ways, ANAPRON's task is complex. ANAPRON uses a CBR module to improve the accuracy of an essentially RBR system. It uses rules to generate a probable pronunciation for a name and then uses cases to handle exceptions to the rules. Although this example is not from Golding and Rosenbloom (1991), one might think of the exception made for the popular British television character Hyacinth Bucket, who insists that her surname be pronounced *bouquet*. Numerous exceptions arise as people try to stand out or fit in or defer to whatever their neighbors happen to call them. Such cases might be viewed as extremely specific rules. For example, if you happen to meet Hyacinth Bucket, then remember to address her as Mrs. Bouquet. Golding and Rosenbloom reported that rules and cases had complementary strengths in ANAPRON. Rules were good for capturing large trends, and cases were good for compensating in situations not covered by the trends. Without CBR, ANAPRON's accuracy might have been improved through continued refinement of the rule set. However, with over 600 rules in the system, a point of diminishing returns had been reached. It was easier to integrate cases than to formalize more rules.

CBR and RBR have also been combined to perform the task of harmonizing melodies (Sabater, Arcos, and López de Mántaras 1998). This task has traditionally been approached in a rule-based manner because there are well-established harmonization rules that reflect common musical practice. However, these rules are not constructive in nature. As Sabater et al. explain, "... the rules don't make the music; it is the music which makes the rules" (p. 147). It is easier for people to appreciate good harmonization or detect bad harmonization than it is for them to generate good harmonization in the first place. GYMEL integrates cases, which are musical phrases from Catalan folk songs, with general harmonization rules. It inputs a musical phrase and outputs the same musical phrase with a set of accompanying chord sequences. CBR is the dominant mode of reasoning, and RBR is applied when CBR can not provide a solution. Although expanding GYMEL's case base might allow CBR to provide more solutions, Sabater et al. did not find it easy to formalize a large number of musical phrases as cases. They believe that their approach is applicable to other domains

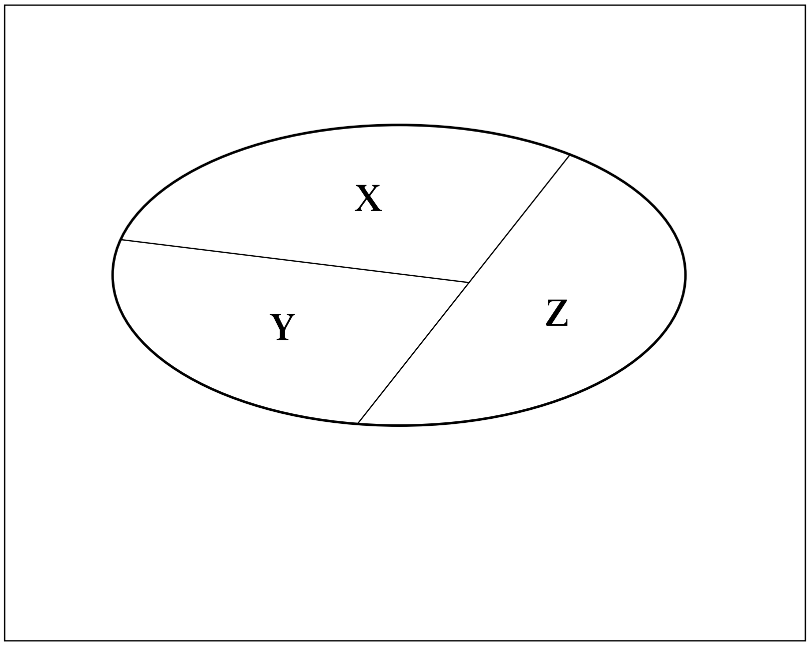

Figure 1. Map-Coloring Problem.

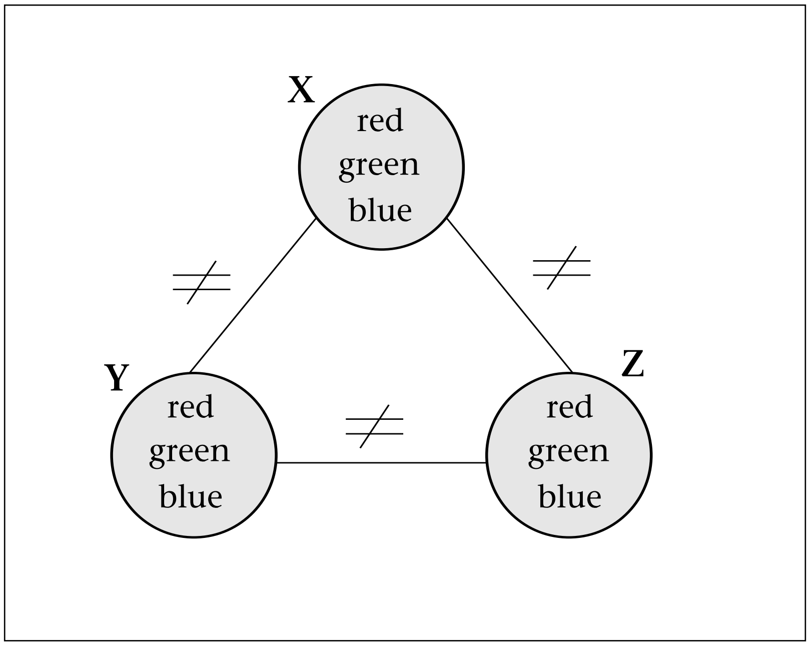

Figure 2. Constraint Graph.

where there are an insufficient number of cases and an inadequate rule base for single strategy problem solving.

Case-Based Reasoning and Constraint-Satisfaction Problem Solving

Constraint satisfaction is a powerful and extensively used AI paradigm (Freuder and Mackworth 1992). It is a natural way of representing problems because the user needs only to state the constraints of the system to be modeled. In addition, it can be applied to many different domains because of its simple but rich representation. Constraint-satisfaction problems (CSPs) involve finding values for variables subject to restrictions on which combinations of values are acceptable.

The advantage of CSP is that it is a reasoning mode that provides both modeling and solving of a problem within the same framework. CSP provides a simple and convenient way of representing problems because it is a natural and declarative approach to modeling. CSP is also domain independent because it can hide many domain-specific issues and be used at a more abstract level. When a problem is represented as a CSP, it can be solved independently of the initial context or domain of application. The CSP methods are applied to the CSP representation of the problem, which hides the context used. CSP provides many advanced algorithms to deal with hard problems (Kumar 1992). Constraint reasoning takes advantage of many mathematical methods and algorithms that were improved to deal with CSPs.

CSP uses a representation called a *constraint graph* in which the vertexes are variables of the problem, and the edges are constraints between variables. Each variable has labels, which are the potential values it can be assigned. CSPs are solved using search and inference methods.

The map-coloring problem illustrates how CSP operates. This problem can be stated as follows: "Given a map with *N* regions bordering each other and *M* colors that can be used to color each region, determine whether there is an assignment of one color to each region such that no two bordering regions have the same color." Figure 1 shows a map-coloring problem.

This problem can be represented as a graph-coloring problem where the nodes represent the different regions, the labels for each node are the different colors a region can have, and the existence of an edge between two nodes indicates that the two corresponding regions cannot have the same color (that is, there is a constraint of "not equal" between these two nodes). Figure 2 shows the graph-coloring problem that corresponds to the map-coloring problem in figure 1.

The graph-coloring problem can be formulated as a CSP. The variables of this problem represent the regions (*X, Y*, and *Z*), the values are the different colors (red, blue and green), and the constraint is that no bordering regions have the same color (that is, no two variables connected by a "not equal" edge can be assigned the same value).

The constraint graph of a CSP is the graphic representation of the problem. In this example, the constraint graph of the map-coloring problem is the same as the corresponding graph-coloring problem. The nodes are the variables, the labels are the values, and the edges are the constraints.

The CSP has a solution if there is an assignment of values to variables such that all the constraints are satisfied. A solution in CSP can mean different things depending on the context and the goal to be achieved. The goal can be to determine whether there is a solution, determine how many solutions there are, find any solution, find an optimal solution, or find a solution with specific characteristics. Figure 3 shows one solution of the map-coloring problem in figure 1.

Many other toy problems such as the *N*-queens problem can also be represented and solved using CSP, and these problems have helped in developing methods and tools that are used in real-world applications. Many real-world applications have used CSP for problem representation and modeling as well as for problem solving (Puget and Leconte 1995; Wallace 1996). These applications include design and configuration, diagnosis, debugging, verification, graphics, decision support, scheduling, planning, and resource allocation.

Figure 4 shows how CSP is used to represent and solve problems. With the CSP representation of a problem, different methods can be used to solve it independently of the context of the application. The two main solving techniques are search and inference. There are many algorithms that use search exclusively, such as backtracking. Backtracking search can require exploration of the entire tree of possibilities to find a solution. Other algorithms make use of an inference method such as arc consistency. Research and experience have shown that the most successful techniques for solving CSPs are the ones that combine search and inference. The questions, then, are how and when to combine them to get the best results. The answers to these questions depend on the domain of application and the available resources.

The earliest systems to integrate CBR and CSP were CADSYN (Maher and Zhang 1991) and JULIA (Hinrichs 1992). CADSYN generates structural designs for buildings. It uses design constraints for case adaptation. CADSYN transforms previous building designs to fit new requirements after conflicts are detected. JULIA designs meal plans for groups of diners. It uses a constraint propagator to identify and resolve constraint violations that arise during meal planning.

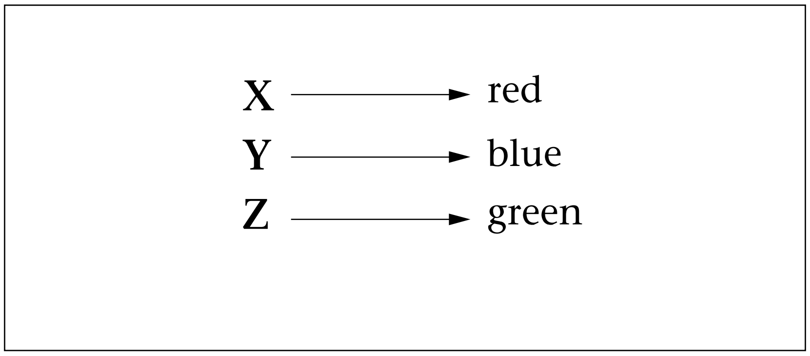

Figure 3. Constraint-Satisfaction Problem Solution.

The two main approaches to integrating CSP and CBR have been to (1) use CBR to initialize the CSP system and (2) use CSP in the adaptation step of CBR. In the first approach, a similar past case retrieved by CBR is used to position the CSP process at a point in its problem space from which to begin its search for a solution; the search travels outwards by means of CSP-based adaptations of the initial case (Sqalli, Purvis, and Freuder 1999). This usage is an example of CBr because only the retrieval phase of CBR is used. In the second approach, CSP provides CBR with a specific method for accomplishing adaptation; thus, CSP is a process used internally within CBR. The use of CSP to accomplish adaptation is relatively new.

Weigel's research prototype (Weigel and Faltings 1998) and the COMPOSER (Purvis and Pu 1996), CADRE (Faltings 1996; Hua and Faltings 1993), IDIOM (Smith, Lottaz, and Faltings 1995) and CHARADE (Avesani, Perini, and Ricci 1993) projects exemplify the two integration approaches. The rationale for Weigel's system was that in configuration problems, the user might have incomplete knowledge of the domain; so, CBR can be used to provide an initial solution to help the user specify the problem. Even though a configuration domain can allow for the development of a complete constraint model, customers might not fully be able to describe the products they want because of their incomplete knowledge of the products or the complexity of the domain. Weigel's system proposes an initial solution that is close to a customer's desired product and then modifies it to meet the remaining requirements. Here, case adaptation is considered to be the process of replacing values in a solution (case) with interchangeable values. It is easier to adapt a nearly satisfactory, or close-by, solution than to construct a complete solution from scratch. Finally, CSP, in turn, helps in creating the case base of initial solutions. Using CSP to create new cases, as well as using CBR to

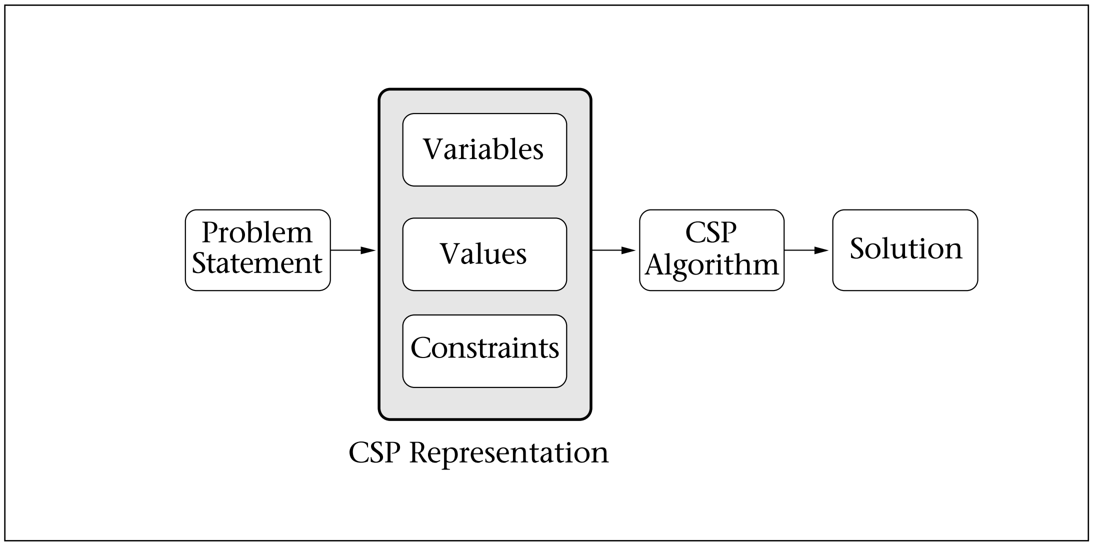

Figure 4. Constraint-Satisfaction Problem for Problem Representation and Problem Solving.

help CSP get started, makes Weigel's integration of CSP and CBR highly synergistic.

COMPOSER solves assembly sequence and configuration design problems using a CSP engine as the adaptation mechanism (Purvis and Pu 1996). In the COMPOSER system, the motivation was to achieve domain independence in case adaptation through CBR-CSP integration. Here, the CBR adaptation process is accomplished using CSP. Each case is formulated as a CSP, and a CSP repair algorithm is used to perform adaptation, resulting in a domain-independent adaptation mechanism. Representing cases as CSPs also allows case combination to occur naturally. When the case-based reasoner is presented with a new problem, similar cases are retrieved, and a CSP repair algorithm (the minimum conflicts algorithm) is used to achieve adaptation. COMPOSER finds the correspondence between an old case and the new one using structure mapping and nearest-neighbor similarity metrics. The minimum-conflicts algorithm is the means by which COMPOSER combines all the solutions to the matched cases into a consistent solution for the new problem. Because case representation and case adaptation both use CSP, the COMPOSER system is one of the most thoroughly integrated uses of CBR and CSP. A strong synergy results because CBR increases the efficiency of CSP, and CSP helps to formalize the adaptation process for CBR (Purvis 1998).

CADRE (Faltings 1996; Hua and Faltings 1993) is used in the domain of floor-plan design. It works together with designers and uses case-based spatial reasoning for the design of buildings. Constraints in CADRE represent numeric relationships among dimension variables. They can be architectural or structural and are used to reduce the adaptation space. CADRE formulates cases as instances of particular buildings, and the constraints restrict the possible modifications that can be made to these cases. There are two forms of adaptation: (1) *dimensional adaptation*, where only dimensions of the case are changed, and (2) *topological adaptation,* where the arrangement and number of spaces and walls are also modified. Constraints expressing the admissible modifications are first solved to obtain a constraint system that is as simple as possible. The complexity of adaptation is then manageable so that it can be carried out by the user in an interactive manner. CADRE has been tested on examples of realistic complexity.

IDIOM is a system for composing apartment layout designs using cases (Smith, Lottaz, and Faltings 1995). The goal of the IDIOM project was to make the solution space easier to explore by designers. It supports designers by reducing constraint complexity and managing design preferences, thereby restraining proposed solutions and further adaptation within feasible design spaces. There are three sources of constraints in IDIOM: (1) the library of cases, in which cases are represented as CSPs; (2) the interpretation of the design by the user; and (3) the domain models. When a case is introduced into a design, all its associated constraints are added to the current set of constraints. The user can then add further constraints to interpret the case in its new environment. A solution is then calculated for the layout. Case combination is supported through incrementally solving relevant constraints. Constraints are restricted to linear and simple nonlinear relationships so that they can be solved rapidly.

The CHARADE Project combines CBR and CSP for situation assessment and planning in forest fire fighting (Avesani, Perini, and Ricci 1993). It aids decision making in environmental emergencies by also incorporating human assessment and reasoning. CSP is used to represent cases in CHARADE. Constraint propagation techniques are applied to part of the plan representation to support adaptation and repair of a sector plan. Thus, CSP supports case adaptation in CHARADE. CBR supports the goal of providing a quick assessment of an emergency situation, and the constraint-solving process enables the system to determine the best exploitation of available resources to handle the emergency.

There have also been other approaches to CBR-CSP integration. ADIOP, a system for diagnosing interoperability problems in asynchronous transfer mode (ATM) networks, uses CBR to compensate for incompleteness in the CSP model and to correct and update the CSP model (Sqalli and Freuder 1998). Here, a diagnostic problem is first modeled as a CSP, and then CBR is used to compensate for what is missing in this CSP model. The CSP model is incomplete or incorrect if it is not completely or correctly modeling an interoperability test. ADIOP can determine incompleteness or incorrectness by considering similar cases and interacting with the user. When a model is incomplete or incorrect, ADIOP uses CBR to check if similar cases of model incompleteness or incorrectness occurred in the past and to adapt them to the new situation. Thus, the user is allowed to complete or correct the model by updating it using the new adapted test case. CBR is also used to update the CSP model to make it more robust in solving future problems. Cases provide information for updating the CSP model that might change some of its components, for example, by adding new constraints. An advantage of CSP in this application is that it reduces the number of cases that would be required by a purely CBR system.

One advantage of CBR-CSP integration is that CSP-based approaches to adaptation offer sophisticated ways to deal with the long-standing problem of case adaptation. Over the years, many researchers have considered adaptation to be the most difficult part of CBR. Some even feel that it ought to be avoided altogether because of its complexity (Barletta 1994). Viewing adaptation as a CSP occurred very early in the evolution of CBR. For example, in the early 1980s, Rissland's CEG (CONSTRAINED EXAMPLE GENERATION) system generated mathematical examples and counterexamples ("ceg's") that satisfied posted desiderata or constraints (Rissland 1980; Rissland and Soloway 1980). CEG did this by first retrieving past examples that already satisfied many of the constraints and then attempting to modify these examples to satisfy the rest of the constraints. Although this retrieval-plus-modification approach worked well enough when there was little constraint interaction, the approach was far too simplistic for many problems. CSP is exactly the sort of mechanism needed to handle these problems. This system from 20 years ago is not only one of the forerunners of CBR but a forerunner of the mixed-paradigm CBr approach of using case retrieval to position further problem solving, such as with CSP.

There are drawbacks and trade-offs, as well as advantages, in CBR-CSP integration. First, there is overhead involved in accommodating both modes in one system, and many additional system components can be required. Second, each reasoning mode has time and space limitations. CBR might need more space for storing cases and more time to perform retrieval and matching. CSP might add more complexity because of the NP-completeness of constraint solving. Third, it might not be appropriate to update CSP models in all problem domains. However, when a balanced integration can be achieved, CBR and CSP complement each other as follows: CSP provides a domain-independent representation of a task; CBR can be used in incomplete domains, where CSP models are difficult or impossible to obtain; CSP provides a knowledge representation that supports problem formulation; CBR provides a useful learning capability; CSP provides many advanced algorithms to deal with hard problems; and CBR provides CSP with a starting point for using these algorithms. A synopsis of the basic CBR steps that have been accomplished using CSP is given in table 1.

Other Techniques

Model-based reasoning: Model-based reasoning (MBR) is an approach in which general knowledge is represented by formalizing the mathematical or physical relationships present in a problem domain. It is perhaps unfortunately named in that CSP relies on formal models, and RBR can be used to model problem domains in which the models are more heuristic or rulelike. MBR models typically capture causal relationships to diagnose problems or predict situation outcomes.

The first system to combine CBR with MBR was CASEY, which diagnosed heart failures (Koton 1988). CASEY was a primarily CBR system that interfaced with a previously existing MBR heart failure program. It used knowledge

System	Case Representation	Case Adaptation	Case Acquisition
Weigel's	No	Yes	Yes
COMPOSER	Yes	Yes	No
CADRE	Yes	Yes	No
IDIOM	Yes	Yes	No
CHARADE	Yes	Yes	No
ADIOP	Yes	No	No

Table 1. Case-Based Reasoning Steps Supported by Constraint-Satisfaction Problem Solving.

representing a physiological model of the heart to match new cases to old ones and derive new diagnoses from old ones. When CASEY could not find a close enough match in its case base, it invoked the original MBR system to solve the problem. Although the MBR system was accurate and reliable, CBR enabled CASEY to improve system efficiency.

A more recent CBR-MBR hybrid is CARMA, an advisory system that helps ranchers manage grasshopper infestations (Branting 1998). CARMA predicts forage loss and provides a cost-benefit analysis for the treatment options in a given situation. It incorporates numeric models developed by entomologists as well as specific cases of past infestations. In this domain, the available numeric models are not complete enough to accurately predict the effects of alternate treatment options. Further, there are not enough documented cases for CBR alone to provide adequate assistance to ranchers. In CARMA, CBR is used to select a similar case, and the model assists in adapting that case's prediction by simulating the effects of different treatment options. The CBR-MBR integration improves solution accuracy over that which is possible using either single approach. The approach used in CARMA, called *approximate–model-based adaptation,* enables prediction and control planning in other domains where individual knowledge sources are insufficient for these tasks. Two other systems that have used this approach are FORMTOOL (Cheatham and Graf 1997), which plans colorant formulas for coloring plastics, and SOPHIST (Aarts and Rousu 1997), which plans bioprocess recipes, such as those for brewing beer.

Genetic algorithms: Genetic algorithms are techniques patterned after the evolutionary process of natural selection. They learn good problem solutions by beginning with arbitrary solutions and creating future generations of better solutions by mutating and propagating the strongest solutions from preceding generations. Maher (1998) has combined genetic algorithms with CBR in an architectural design domain. In architectural design, it is not sufficient to recall a past blueprint that would meet current needs. Rather, some new synthesis that draws on past designs must be achieved. To date, methods that have attempted to address this issue have been time consuming and knowledge intensive. Genetic algorithms provide an efficient means of combining pieces of past designs. Genetic algorithms are used to combine pieces of past building designs into new composite designs. Design cases are represented as genes, which are manipulated using the traditional genetic algorithm techniques of crossover and mutation. Many building designs are quickly generated and tested, allowing satisfactory designs to be identified and maintained.

Information retrieval: Information retrieval applies primarily statistical techniques to the problem of locating appropriate text documents in large, unstructured collections, such as newspaper archives, legal libraries, and the World Wide Web. Typically, users indicate the type of document they deem appropriate by entering structured queries containing keywords. An information-retrieval system strives to achieve *perfect recall*, or retrieval of all relevant documents, and *perfect precision*, or retrieval of only relevant documents. The degree of recall and precision actually achieved can depend on the user's skill in formulating queries.

SPIRE, a CBR–information-retrieval integration, allows users to retrieve documents from a large legal corpus without having to specify formal queries (Daniels and Rissland 1997; Rissland and Daniels 1996). It uses cases of legal documents and passages to generate queries, which are then run by a full-text retrieval engine. SPIRE first analyzes a new problem with respect to its own case base in the way standard for legal CBR systems. It then uses the results of this analysis to drive the INQUERY information-retrieval system, which through its relevancy feedback module retrieves documents in its standard way. The CBR module of SPIRE analyzes the problem and selects the most relevant cases that it has in its case base. It hands the texts of these cases to INQUERY, which automatically generates a standard query composed of terms or pairs of terms. Thus, SPIRE bridges the gap between the purely symbolic CBR engine and the purely text-based retrieval engine. This enabled SPIRE to achieve excellent results, particularly when large numbers of terms (100 or more) were used. For example, in one experiment, SPIRE was tested on a problem case based on an actual tax case (the Weissman case) (Rissland and Daniels 1996) where its CBR module retrieved cases causing its information-retrieval module to generate a query of 150 terms. SPIRE returned several important home-office deduction cases, including two that were factually right on point in that they involved professors taking home-office deductions, just as in the problem case. Perhaps more importantly, SPIRE found the most important case (Soliman) on the topic in all its three versions: (1) a 1990 lower Tax Court case; (2) a 1991 case in the Court of Appeals; and (3) a 1993 Supreme Court case, which still is the definitive case on this topic. None of these cases were known to SPIRE, and all the Soliman cases were decided after SPIRE's case base was created. Thus, not only was SPIRE able to find highly relevant documents, it was also able to cope successfully with the "staleness" problem by accessing documents created long after its own case base.

SPIRE uses a similar approach in retrieving specific passages from within a large document. In this "inner loop," SPIRE uses past examples of text discussing an issue (for example, the honesty of a debtor applying for bankruptcy relief) to generate a query to run on the document to locate passages where the issue is discussed. SPIRE allows ordinary users to run queries at an expert level, minimizing the time they spend reviewing retrieved documents that they do not deem to be appropriate or relevant. SPIRE is an example of CBr in that CBR is used (twice!) to aid information retrieval, first to find relevant documents and then within each document to find relevant passages.

The STAMPING ADVISER system is another CBR–information-retrieval integration, which aids feasibility engineers in evaluating designs for stamped metal automotive parts at Ford (Leake et al. 1999). Here, CBR and information retrieval are integrated into the engineer's overall work flow to support the entire design process. Stamped parts make up the bodies of automobiles and, as such, are designed to meet aesthetic, structural, functional, cost, and manufacturability criteria. They are initially designed using a computer-aided design (CAD) tool, and then they are critiqued in a feasibility analysis to identify any potential problems before the design is finalized. Past cases of stamped parts, along with problems encountered in their design and solutions to those problems, are used in the STAMPING ADVISER. These past cases are integrated with other available knowledge sources, such as company guidelines, to ensure consistent styling at a reasonable cost. The STAMPING ADVISER uses *just-in-time retrieval;* that is, it presents information when it is needed rather than require the user to make explicit queries. Based on the task the feasibility engineer is engaged in, the system automatically generates queries to retrieve and present relevant style guidelines. Thus, CBR and information retrieval are integrated into the larger task context, providing the feasibility engineer with a tool that is natural to use.

Case-based reasoning: Researchers have also suggested integrating multiple CBR components within a single system. This approach is useful when a problem domain requires two different styles of CBR, such as transformational and derivational CBR, or two different levels of reasoning, such as domain-specific reasoning and metareasoning. A system that exemplifies this approach is DIAL, which operates in the domain of disaster response planning (Leake and Kinley 1998). It generates high-level plans for reacting to disasters, such as floods and earthquakes. DIAL might be considered a CBR-RBR hybrid in that RBR is used to support similarity assessment and case adaptation. However, its strength lies in integrating different CBR components to monitor, capture, and replay its reasoning processes. DIAL's controlling process is a transformational CBR module that generates new disaster response plans by remembering and adapting old ones. It is supported by derivational CBR modules that assist with similarity assessment, case retrieval, and case adaptation.

Tasks Benefiting from Case-Based–Reasoning Integration

Among tasks that have benefited from CBR integration are interpretation and argumentation, design and synthesis, planning, and the management of long-term medical conditions.

Interpretation and Argumentation

Interpretive CBR is frequently used in disciplines such as ethics, public policy, and law, where new fact situations are interpreted in light of past ones. In these disciplines, concepts are often *open textured* in that they cannot be defined by universally valid necessary and sufficient conditions but, rather, have a penumbra of borderline cases whose interpretation is arguable (that is, they could reasonably be classified either way, as positive or negative instances of the concept), and experts often disagree and argue about their interpretation. Appellate law typically deals with difficult, penumbral cases. In fact, CBR is the primary mode of reasoning underlying Anglo-American law. Such precedent-based legal systems are based on the doctrine of *stare decisis* that mandates that similar cases should be decided similarly.

Among tasks that have benefited from CBR integration are interpretation and argumentation, design and synthesis, planning, and the management of long-term medical conditions.

Statutory legal domains abound in interpretive CBR and RBR. A good example of such a domain is tax law, which is based on statutory rules and regulations but whose rules are not ironclad and whose ingredient terms—even the term *income*—are not well defined. Despite these shortcomings, they must nevertheless be applied to myriad situations. To inform their application, all involved—taxpayers, Internal Revenue Service administrators, accountants, lawyers, tax courts—use cases. For example, the so-called home-office deduction rule requires that a taxpayer use his/her home office on a "regular basis" (Section 280A(c)(1) of the tax code), but nowhere is there a definition of what constitutes such usage. The best guide is to look at how this term has been interpreted in past cases. The CABARET system (Rissland and Skalak 1991), one of the first CBR-RBR hybrids, operates in this domain.

CABARET's control regime gives CBR and RBR equal footing. Its domain-independent architecture has the following major components: (1) there are two independent co-reasoners, one case based and one rule based; (2) each reasoner has a dedicated monitor that makes observations on its processing, results, and partial results and recasts these observations in a language understandable by the controller; and (3) an agenda-based controller uses the observations of the monitors to propose and select tasks to be acted on by the individual co-reasoners.

CABARET's system architecture allows it to combine CBR and RBR in a highly opportunistic way. The system architecture, first described in Rissland and Skalak (1991), is shown in figure 5. CABARET works as follows: One of the co-reasoners works on a task; the monitor modules make observations; and based on the observations, the controller posts new tasks and selects the next one to be worked on. On conclusion, CABARET outputs a memo generated by filling in a stylized template.

CABARET uses four distinct sources of knowledge: (1) CBR knowledge, including a case base, indexes, and similarity metrics; (2) RBR knowledge, including rules and predicates; (3) control knowledge, or domain-independent control rules to propose and rank tasks based on observations made by the monitors; and (4) general domain knowledge, especially hierarchies available to all modules.

CABARET's control rules guide its problem solving and embody its theory of statutory interpretation. Examples are (1) if one mode of reasoning fails, then switch to the other; (2) once a conclusion is reached, switch the form of reasoning to check if it holds in the other mode; (3) if all but one antecedent of a rule can be established, then use CBR to show the missed antecedent can be established using cases; and (4) if all but one antecedent of a rule can be established, then use CBR to show the missed antecedent is not necessary. The last two examples are "near-miss" heuristics. Not only are they particularly useful in rule interpretation and argument in legal reasoning (Skalak and Rissland 1992), they could also be used in any mixed case-and-rule domain in which the rules are not logically ironclad.

Note that interpretive CBR is not confined to profound problems of law and philosophy. It is ubiquitous in everyday life. For example, most parents eventually face an instance of deciding whether or not to give their high school–aged son or daughter the keys to the family car to drive to an event (a rock concert, the senior prom) many miles away from home and likely to involve late night driving. Most children

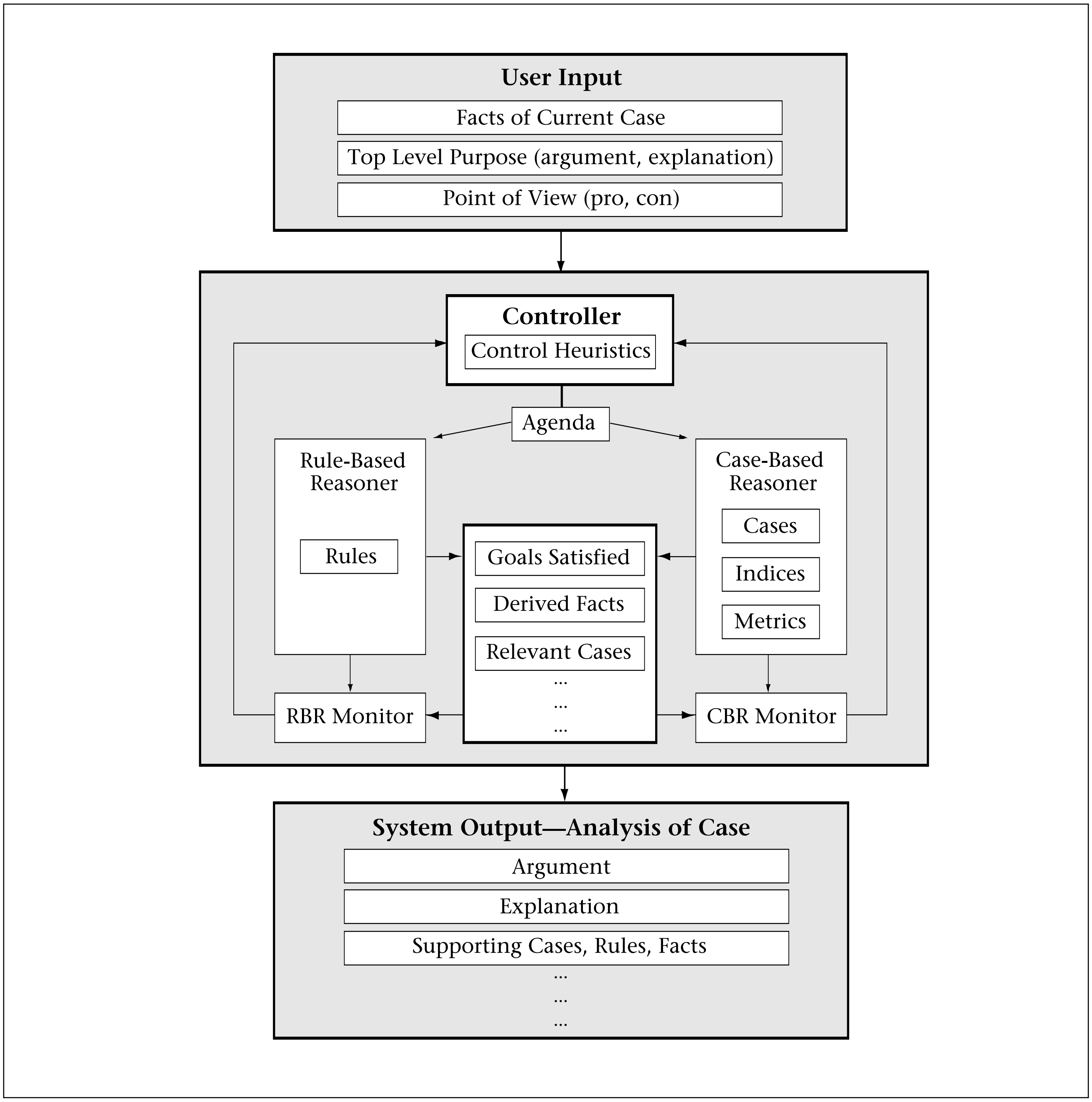

Figure 5. CABARET*'s Architecture.*

face a situation of making the "case" for such permission. Parents and child might argue their positions by considering how such requests were handled in past cases, for example, those involving older siblings, classmates, and neighbors. They would no doubt delve into such issues as driving experience, past behavior in situations requiring responsible action, the perceived hazards of the trip, likely weather and road conditions, and the community standards of the child's and the parents' peer groups.

The end result of this type of CBR is not only an interpretation of the new case but also a rationale justifying it. A mere yes or no would not suffice in the previous example (especially if the "petitioner" loses). Case-based justification involves comparing and contrasting the new case with relevant precedents; creating analogies between the new case and the supporting precedents; and breaking analogies with, or "distinguishing away," contrary cases.

Design and Synthesis

Design is the process of fully describing a new artifact that will, when produced, satisfy a given set of requirements. In case-based design, past designs or design experiences are used in the creation of new designs. The task of design is one in which integrating other approaches with CBR has proven fruitful.

FABEL is a CBR-MBR-RBR system for architectural design (Gebhardt et al. 1997). A building in FABEL contains tens of thousands of design objects. It comprises several cases, each containing from two to a few hundred design objects. Some design objects are concrete, such as walls or pipes, and others are more abstract, such as intended use or climate. The design of one building encompasses many problems, each of which can be solved by different means. FABEL provides architects and civil engineers with an integrated design environment called a *virtual construction site*. It includes a computer-aided architectural design (CAAD) tool, an object-oriented distributed database, and several problem solvers. Some problem solvers are case based, but others use models for building subsystems, and one uses rules that have been induced from the cases in the system. Gebhardt et al. maintain that the complexity of this real-world problem makes it untenable to use single design cases or a single problem-solving strategy.

CAMPER is a design system that combines CBR with RBR (Marling and Sterling 1998). It designs a daily menu for an individual in accordance with nutrition guidelines, personal preferences, and aesthetic criteria. Planning nutritional menus is a task researchers have tried to automate since the early 1960s (Balintfy 1964). The first computer-based menu planner used linear programming to optimize a menu for nutritional adequacy, cost, and consumer satisfaction. The menus it generated met its first two goals but were not considered appetizing by people. Subsequent attempts also failed, in part because rules for evaluating menus were insufficient for generating satisfactory menus in the first place. JULIA, the CBR-CSP system that planned menus for dinner parties, showed that past menus could provide a basis for planning appetizing new ones (Hinrichs 1992). However, it was a self-proclaimed "party animal" and did not try to incorporate nutrition criteria. CAMPER is used by a qualified nutritionist in consultation with a client who must constrain intake of calories, fat, sodium, calcium, protein, cholesterol, or other nutrients. In designing menus manually, nutritionists naturally use both CBR and RBR. CAMPER was built by combining the best features of independent CBR and RBR nutritional menu-planning systems. The CBR component stores, retrieves, and adapts daily menus, and the RBR component relies on a process of generate, test, and repair, using menu patterns, an ontology of foods, and common-sense knowledge of the ways in which foods can be combined. CBR and RBR complement each other in CAMPER. CBR contributes an initial menu that meets design constraints by capitalizing on food combinations that have proven satisfactory in the past. RBR allows analysis of alternatives, so that innovation beyond the tried and true becomes possible. After an initial menu is proposed through CBR, the user interacts with an RBR what-if analysis module to explore creative alternatives. The user can modify the menu, guided by rules, and then save significantly altered menus in the case base for future use.

Another CBR-RBR hybrid, SAXEX, integrates background musical knowledge into a primarily CBR system for generating expressive musical performances (Arcos, Lopez de Mantaras, and Serra 1998). This application is of practical use in music performance and synthesis because musical scores contain information about the pitch and duration of each note, but they do not help with the expressive parameters of rubato, vibrato, articulation, and attack. These parameters are ordinarily inferred and incorporated into a performance by a human performer. SAXEX operates in the context of tenor saxophone interpretation of standard jazz ballads. SAXEX infers a set of expressive transformations to apply to an inexpressive musical input phrase and then outputs an expressive performance of the same phrase. Spectral modeling and synthesis techniques are used to extract high-level parameters from expressive musical performances. The expressive performances are stored in SAXEX's case base. A case in SAXEX consists of a musical score plus the extracted parameters that allow it to be played expressively. Background musical knowledge is used for identifying cases that are similar to an input problem and transforming a new inexpressive piece based on features of a stored expressive piece. Experiments run on SAXEX showed results comparable to human performance. Researchers plan to extend SAXEX to make it a practical tool for musicians.

Planning

A *planning task* consists of constructing a sequence of actions to achieve a specified set of goals when starting from an initial situation. A classical generative planning process involves searching a space of possible actions to obtain the sequence of actions solving the problem. CBR techniques have been shown to be effective for two main reasons: First, frequently, there is no complete domain theory that could be used to generate plans from scratch. In this situation, cases serve as additional knowledge about the domain. Second, in domains where a complete domain theory is available, generating plans from scratch can be computationally expensive. CBR can be used as a source of metaknowledge indicating how to use domain theory to solve a given problem.

Most CBR approaches to planning that do not assume the existence of a complete domain theory use transformational analogy for plan adaptation, although derivational analogy can also be used. In transformational analogy, cases are transformed to obtain a solution plan, as in DIAL, the disaster response planner, which uses transformational analogy as its control process and derivational analogy for support (Leake and Kinley 1998). In derivational analogy, cases contain annotated derivational traces, which are followed when solving a new problem. Derivational analogy approaches are most frequently used in connection with first-principles planning systems.

CBR has been combined with two types of first-principles planning: (1) STRIPS-style planning and (2) hierarchical planning. STRIPS-style planning originated in the Stanford Research Institute Problem Solver (STRIPS) Project of the early 1970s. STRIPS was a general problem solver that gave instructions to SHAKEY the robot as it navigated its way through different rooms, moving objects from one room to another (Fikes and Nilsson 1971). In STRIPS-style planning, the assumption is made that changes only occur in the world if indicated by the operators in the domain definition. These operators are used to generate the ordered steps of a plan that, when executed, will achieve the intended goal. In hierarchical planning, plans have several levels of abstraction. Complex tasks are decomposed into simpler ones. Plans at the most concrete level consist of sequences of STRIPS-style operators, but planning begins at higher levels of abstraction. Hierarchical planning has been proven to be strictly more expressive than one-level STRIPS-style planning (Erol, Nau, and Hendler 1994).

The first system to combine CBR with STRIPS-style planning was PRODIGY/ANALOGY (Veloso 1994). PRODIGY/ANALOGY is a generative case-based planner. It is a total-order planning system, which means that the steps appear in the finished plan in the same order in which they are generated. PRODIGY/ANALOGY is domain independent, but it has been validated in a logistics transportation domain in which packages are moved among different locations by trucks and airplanes. Cases in PRODIGY/ANALOGY are used to guide the search process of the integrated planning system. Because the search space is of exponential size on the length of the plans, planning by first principles can be inefficient. The integration results in a significant improvement in terms of the time needed to generate plans. Two other systems that integrate CBR with STRIPS-style planning to improve overall efficiency are DERSNLP (Ihrig and Kambhampati 1994) and CAPLAN/CBC (Munoz-Avila and Weberskirch 1996). Unlike PRODIGY/ANALOGY, these two systems use partial-order planning, in which the order of steps in the plan is decoupled from the order in which the steps are generated. Both systems are domain independent, but DERSNLP was tested on PRODIGY/ANALOGY's logistics transportation domain, and CAPLAN/CBC was tested in a process-planning domain for manufacturing mechanical workpieces.

CBR has also been integrated into two hierarchical planning systems: (1) the joint maritime crisis action planning (JMCAP) system (desJardins, Francis, and Wolverton 1998) and the hierarchical interactive case-based architecture for planning (HICAP) system (Munoz-Avila et al. 1999). JMCAP and HICAP support military operations. JMCAP extends an existing hierarchical task network (HTN) by integrating past maritime crisis action-planning experience. The rationale for this extension is that it more closely models the processes used by human planners. HICAP was tested on planning noncombatant evacuation operations (NEOs). NEOs are conducted when nonmilitary personnel must be evacuated from dangerous situations and moved to safe havens. Here, there is no complete domain theory to use in generating plans. Cases provide knowledge about previous military operations, which would otherwise be unavailable for planning. Cases complement the knowledge provided by doctrine, the general guidelines for military planning, and standard operational procedures, the documented methods for achieving military objectives.

Management of Long-Term Medical Conditions

The medical domain has provided fertile ground for CBR integration research ever since the CBR-MBR system CASEY (Koton 1988), which diagnosed heart failures. Although early AI in medicine systems focused on diagnosis, a more recent research emphasis is on systems that support the treatment of chronic, or long-term, conditions, which cannot be cured but nevertheless must be managed in the best interests of the patient. Both problem-solving and interpretive CBR come into play, as do other reasoning modalities.

CARE-PARTNER assists clinicians responsible for the long-term follow-up care of cancer patients who have undergone bone-marrow transplants (Bichindaritz, Kansu, and Sullivan 1998). It integrates CBR, RBR, and information retrieval to support evidence-based medical practice. Cases are used to show how problems were solved for past patients. Rules are derived from standard practice guidelines and special practice pathways developed by experts. Information retrieval is used to provide relevant documents from the medical literature that cannot readily be translated into rules. CARE-PARTNER stores cases, rules, and documents in a unified knowledge base. No reasoning modality is dominant; rather, reasoning steps from different modalities are merged based on the problem at hand.

The Auguste Project is a current effort to provide decision support for planning the ongoing care of Alzheimer's disease patients (Marling and Whitehouse 2001). Formulated as a study in CBR integration, researchers are focusing first on those reasoning modalities naturally used in this task by physicians, nurses, and social workers, including CBR and RBR. They next plan to explore whether less intuitive approaches, such as Bayesian reasoning, might augment the natural reasoning processes. Geriatrics practitioners rely heavily on case studies for training, making suggestions for treatment, sharing best practices, and reporting clinical findings. They also rely on guidelines, or rules, to determine which interventions to prescribe and to tailor prescribed interventions to particular patients. Because the symptoms of Alzheimer's disease vary greatly from patient to patient, and within the same patient over time, there is no critical pathway, or overall set of rules, that can be applied to all patients. Although the study is still ongoing, CBR has already proven useful for classifying patients into subcategories, within which small and manageable rule sets can apply.

... much remains to be learned before system developers will be able to select just the right integration strategy in advance of system implementation.

Open Issues and Future Work

In a case-based manner, many examples, or cases, of CBR integration have been presented. These example systems are summarized in table 2. Much can be learned from these examples. However, much remains to be learned before system developers will be able to select just the right integration strategy in advance of system implementation. Thorough evaluations of integrated systems are needed to identify the contributions of each paradigm to the whole. New studies are needed that do not stop with what "works" for a particular application but that explore beyond working solutions to develop an understanding that could lead to even better solutions.

Open issues in CBR integration include finding the best system architectures and knowledge representations for hybrid systems. One key issue in multimodal reasoning, whether by person or program, is how the different modes of reasoning are integrated. If we consider CBR-RBR integrations, we find that the modes of reasoning with rules and cases are quite different—logical reasoning for rules and analogical reasoning for cases—and thus can be performed by separate modules that must, however, be coordinated. For example, is CBR only to be used to rescue RBR from difficulties or provide a sanity check, or is the processing more intimately intertwined? How can CBR enhance the other mode, and vice versa? Many hybrids use CBr to narrow the search space for the other mode by seeding or localizing problem solving to "neighborhoods" around the most relevant cases. Some integrations use failures in the non-CBR mode to trigger acquisition of cases. In most of the applications discussed in this article, the integration of CBR with another mode of reasoning is "straight line" in the sense that first the program performs one and then the other. In other systems, such as CABARET and CARE-PARTNER, processing is more intertwined.

At the 1998 American Association for Artificial Intelligence Workshop on CBR Integrations, Aha and Daniels (1998) trichotomized prevailing system architectures as follows: (1) *master-slave:* other reasoning methods support the CBR component; (2) *slave-master:* the CBR component supports other reasoning methods; (3) *collaborating:* there are less stratified collaborations. GYMEL, the system that harmonizes melodies, is an example of the master-slave architecture. CBR is the dominant mode, but RBR assists when insufficient cases are available to suggest a harmonization. ADIOP, which diagnoses interoperability problems in ATM networks, exemplifies the slave-master architecture. Here, CSP is the dominant paradigm, but CBR is used to compensate when CSP models are incomplete. CARE-PARTNER uses a collaborating architecture, intertwining CBR, RBR, and

System	Domain[1]	Task	CBR Plus
ADIOP	ATM networks	Diagnosis	CSP
ANAPRON	Speech	Synthesis	RBR
AUGUSTE Project	Medicine	Ongoing treatment	RBR
CABARET	Law	Argumentation, interpretation	RBR
CADRE	Architecture	Design	CSP
CADSYN	Architecture	Design	CSP
CAMPER	Menu planning	Design	RBR
CAPLAN/CBC	Manufacturing	Planning	STRIPS planning
CARE-PARTNER	Medicine	Ongoing treatment	RBR, Information retrieval
CARMA	Entomology	Prediction, planning	MBR
CASEY	Medicine	Diagnosis	MBR
CHARADE	Forest fires	Planning	CSP
COMPOSER	Engineering	Configuration	CSP
DERSNLP	Transportation	Planning	STRIPS planning
DIAL	Disaster response	Planning	CBR components
FABEL	Architecture	Design	MBR, RBR
FORMTOOL	Plastic colorants	Planning	MBR
GREBE	Law	Argumentation, interpretation	RBR
GYMEL	Music	Synthesis	RBR
HICAP	Military	Planning	Hierarchical planning
IDIOM	Architecture	Design	CSP
IKBALS	Law	Argumentation, interpretation	RBR, Information retrieval
JMCAP	Military	Planning	Hierarchical planning
JULIA	Menu planning	Design	CSP
Maher's	Architecture	Design	Genetic algorithms
PRODIGY/ANALOGY	Transportation	Planning	STRIPS planning
SAXEX	Music	Synthesis	RBR
SOPHIST	Bioprocess recipes	Planning	MBR
SPIRE	Law	Document retrieval	Information retrieval
STAMPING ADVISER	Engineering	Design	Information retrieval
Weigel's	Products for sale	Configuration	CSP

1. For domain-independent systems, the primary test domain is listed.

CSP = constraint-satisfaction problem.
RBR = rule-based reasoning.
MBR = model-based reasoning.

Table 2. Summary of Case-Based Reasoning Integrations.

information retrieval to assist with the long-term follow-up care of cancer patients. Participants at the AAAI-98 Workshop on CBR Integrations categorized their own systems, reporting 11 master-slave integrations, 6 slave-master integrations, and 12 collaborating reasoners.

Another key issue is the representation of knowledge. Different modalities have distinct ways of structuring knowledge. RBR relies on rules, which represent knowledge by expressing relationships, typically in the form of antecedents and consequents. Each rule contains a fairly small unit of knowledge, so multiple rules are chained together during inferencing. CSP represents knowledge as constraint graphs in which nodes are variables, labels on nodes are values, and edges are constraints. A single graph contains a problem description. MBR uses mathematical equations and relationships. Genetic algorithms represent knowledge as genes, which are bit strings in which each part of the string stands for an attribute-value pair. Information retrieval does not structure knowledge at all; rather, it is used to find information that has not been structured.

There is considerable flexibility in representing a case in a CBR system. It is important to capture a description of a past problem, the past solution to the problem, and the outcome of applying this solution to the problem. How these are captured varies greatly from system to system. Integrations in which cases are directly represented by constraint graphs or genes take advantage of this flexibility. However, most integrations to date maintain different knowledge representations for different reasoning modalities. What are the advantages and disadvantages of each representation? Could knowledge representations themselves be integrated to develop useful composite knowledge structures? More research is needed to answer these questions.

Summary and Conclusions

Over 10 years ago, Rissland and Skalak (1989) wrote, "At this point, researchers have only recently begun to write about the integration of CBR with other reasoning paradigms. We feel that such mixed-paradigm approaches are natural and shed light on both the cognitive skills involved in such reasoning and on questions of architecture and control of their computational models" (p. 526). Today, researchers have uncovered many synergistic ways to combine CBR with other modes of reasoning. CBR has fruitfully been combined with RBR, CSP solving, MBR, genetic algorithms, and information retrieval. CBR integrations have supported a wide range of tasks, including interpretation and argumentation, design and synthesis, planning, and management of long-term medical conditions. Many useful synergies have emerged as different reasoning strategies extend and complement each other. Integrated systems have enabled more accurate modeling of domain knowledge, compensation for incomplete domain models and rule bases, compensation for small case bases, simplification of knowledge acquisition, improved solution quality, improved system efficiency, leveraging of past experiences, and compensation for shortcomings inherent in individual reasoning strategies.

More research is needed because open issues remain to be resolved. Thorough system evaluations are needed to pinpoint the specific strengths and weaknesses of different reasoning strategies for performing different tasks. New studies are needed to clarify the advantages and disadvantages of the various system architectures and knowledge representations that can be used. Future efforts in this area might well lead to discovery of new synergies, greater methodological support for building practical integrated systems, and a better understanding of the ways in which people and machines reason.

Acknowledgments

This survey article grew out of discussions that were initiated at the AAAI-98 Workshop on CBR Integrations and continued by the coauthors. The coauthors are grateful to the workshop participants for their insights and contributions.

References

Aamodt, A., and Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7(I): 39–59.

Aarts, R., and Rousu, J. 1997. Qualitative Knowledge to Support Reasoning about Cases. In *Proceedings of the Second International Conference on Case-Based Reasoning,* 489–498. Berlin: Springer.

Aha, D., and Daniels, J. J., eds. 1998. *Case-Based Reasoning Integrations: Papers from the 1998 Workshop*. Menlo Park, Calif.: AAAI Press.

Arcos, J. L.; López de Mántaras, R.; and Serra, X. 1998. Integrating Background Musical Knowledge in a CBR System for Generating Expressive Musical Performances. In *Case-Based Reasoning Integrations: Papers from the 1998 Workshop,* 12–16. Menlo Park, Calif.: AAAI Press.

Ashley, K. D. 1990. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. Cambridge, Mass.: MIT Press.

Avesani, P.; Perini, A.; and Ricci, F. 1993. Combining CBR and Constraint Reasoning in Planning Forest Fire Fighting. In *Proceedings of the First European Workshop on Case-Based Reasoning,* 325–328. Berlin: Springer.

Balintfy, J. L. 1964. Menu Planning by Computer. *Communications of the ACM* 7(4): 255–259.

Barletta, R. 1994. Comments from Address Given at the Second European Conference on Case-Based Reasoning (ECCBR-94), 7–10 November, Chantilly, France.

Bichindaritz, I.; Kansu, E.; and Sullivan, K. M. 1998. Integrating Case-Based Reasoning, Rule-Based Reasoning, and Intelligent Information Retrieval for Medical Problem Solving. In *Case-Based Reasoning Integrations: Papers from the 1998 Workshop,* 22–27. Menlo Park, Calif.: AAAI Press.

Branting, L. K. 1998. Integrating Cases and Models through Approximate Model-Based Adaptation. In *Multimodal Reasoning: Papers from the 1998 AAAI Spring Symposium,* 1–5. Menlo Park, Calif.: AAAI Press.

Branting, L. K. 1991. Building Explanations from

Rules and Structured Cases. *International Journal of Man-Machine Studies* 34(6): 797–837.

Cheatham, W., and Graf, J. 1997. Case-Based Reasoning in Color Matching. In *Proceedings of the Second International Conference on Case-Based Reasoning,* 1–12. Berlin: Springer.

Daniels, J. J., and Rissland, E. L. 1997. What You Saw Is What You Want: Using Cases to Seed Information Retrieval. In *Proceedings of the Second International Conference on Case-Based Reasoning,* 325–336. Berlin: Springer.

desJardins, M.; Francis, A.; and Wolverton, M. 1998. Hybrid Planning: An Approach to Integrating Generative and Case-Based Planning. In *Case-Based Reasoning Integrations: Papers from the 1998 Workshop,* 45–49. Menlo Park, Calif.: AAAI Press.

Erol, K.; Nau, D.; and Hendler, J. 1994. HTN Planning: Complexity and Expressivity. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 1123–1128. Menlo Park, Calif.: American Association for Artificial Intelligence.

Faltings, B. 1996. Working Group in Model-Based Design and Reasoning. Part II: Design. *AI Communications* 9(2): 65–73.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving in Problem Solving. *Artificial Intelligence* 5(2): 189–208.

Freuder, E., and Mackworth, A. 1992. Constraint-Based Reasoning. *Artificial Intelligenc*e (Special Issue) 58.

Gebhardt, F.; Voss, A.; Gräther, W.; and Schmidt-Belz, B. 1997. *Reasoning with Complex Cases.* Boston, Mass.: Kluwer Academic.

Golding, A. R., and Rosenbloom, P. S. 1991. Improving Rule-Based Systems through Case-Based Reasoning. In Proceedings of the Ninth National Conference on Artificial Intelligence, 22–27. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hammond, K. J. 1989. *Case-Based Planning: Viewing Planning as a Memory Task*. San Diego, Calif.: Academic.

Hammond, K. J. 1986. CHEF: A Model for Case-Based Planning. In Proceedings of the Fifth National Conference on Artificial Intelligence, 267–271. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hart, H. L. A. 1958. Positivism and the Separation of Law and Morals. *Harvard Law Review* 71(4): 593–629. Reprinted in Hart, H. L. A. 1983. *Essays in Jurisprudence and Philosophy.* Oxford: Clarendon Press.

Hayes-Roth, F.; Waterman, D. A.; and Lenat, D. B., eds. 1983. *Building Expert Systems*. Reading, Mass.: Addison-Wesley.

Hinrichs, T. R. 1992. *Problem Solving in Open Worlds: A Case Study in Design.* Hillsdale, N. J.: Lawrence Erlbaum.

Hua, K., and Faltings, B. 1993. Exploring Case-Based Building Design—CADRE. *AI in Engineering Design, Analysis and Manufacturing* 7(2): 135–143.

Ihrig, L. H., and Kambhampati, S. 1994. Derivation Replay for Partial-Order Planning. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 992–997. Menlo Park, Calif.: American Association for Artificial Intelligence.

Kolodner, J. 1993. *Case-Based Reasoning.* San Francisco, Calif.: Morgan Kaufmann.

Kolodner, J. L., and Leake, D. B. 1996. A Tutorial Introduction to Case-Based Reasoning. In *Case-Based Reasoning: Experiences, Lessons, and Future Directions,* ed. D. B. Leake, 31–66. Menlo Park, Calif.: AAAI Press.

Koton, P. 1988. Reasoning about Evidence in Causal Explanations. In Proceedings of the Seventh National Conference on Artificial Intelligence, 256–261. Menlo Park, Calif.: American Association for Artificial Intelligence.

Kumar, V. 1992. Algorithms for Constraint-Satisfaction Problems: A Survey. *AI Magazine* 13(1): 32–44.

Leake, D. B., and Kinley, A. 1998. Integrating CBR Components within a Case-Based Planner. In *Case-Based Reasoning Integrations: Papers from the 1998 Workshop,* 80–84. Menlo Park, Calif.: AAAI Press.

Leake, D. B.; Birnbaum, L.; Hammond, K.; Marlow, C.; and Yang, H. 1999. Integrating Information Resources: A Case Study of Engineering Design Support. In *Proceedings of the Third International Conference on Case-Based Reasoning,* 482–496. Berlin: Springer.

Maher, M., and Zhang, D. 1991. CADSYN: Using Case and Decomposition Knowledge for Design Synthesis. In *Artificial Intelligence in Design,* ed. J. S. Gero, 137–150. Oxford, U.K.: Butterworth-Heineman.

Maher, M. L. 1998. CBR as a Framework for Design: Augmenting CBR with Other AI Techniques. In *Case-Based Reasoning Integrations: Papers from the 1998 Workshop,* 96–101. Menlo Park, Calif.: AAAI Press.

Marling, C., and Sterling, L. 1998. Integrating CBR and RBR for Nutritional Menu Design. In *Case-Based Reasoning Integrations: Papers from the 1998 Workshop,* 102–107. Menlo Park, Calif.: AAAI Press.

Marling, C. R., and Whitehouse, P. J. 2001. The AUGUSTE Project: Decision Support for Patient Care. In Proceedings of the International ICSC-NAISO Congress on Computational Intelligence: Methods and Applications, 65–70. Canada/The Netherlands: ICSC Academic.

Muñoz-Avila, H., and Weberskirch, F. 1996. Planning for Manufacturing Workpieces by Storing, Indexing and Replaying Planning Decisions. In *Proceedings of the International Conference on AI Planning Systems*. Menlo Park, Calif.: AAAI Press.

Muñoz-Avila, H.; McFarlane, D.; Aha, D. W.; Ballas, J.; Breslow, L. A.; and Nau, D. 1999. Using Guidelines to Constrain Interactive Case-Based HTN Planning. In *Proceedings of the Third International Conference on Case-Based Reasoning,* 288–302. Berlin: Springer.

Puget, J. F., and Leconte, M. 1995. Beyond the Glass Box: Constraints as Objects. In *Proceedings of the International Logic Programming Symposium,* 513–527. Cambridge, Mass.: MIT Press.

Purvis, L. 1998. Synergy and Commonality in Case-Based and Constraint-Based Reasoning. In *Multimodal Reasoning: Papers from the 1998 AAAI Spring Symposium,* 80–84. Menlo Park, Calif.: AAAI Press.

Purvis, L., and Pu, P. 1996. An Approach to Case Combination. Paper presented at the Workshop on Adaptation in Case-Based Reasoning, Twelfth European Conference on Artificial Intelligence, 11–16 August, Budapest, Hungary.

Riesbeck, C. K., and Schank, R. C. 1989. *Inside Case-Based Reasoning.* Hillsdale, N.J.: Lawrence Erlbaum.

Rissland, E. L. 1980. Example Generation. Paper presented at the Third National Conference of the Canadian Society for Computational Studies of Intelligence, May, Victoria, British, Columbia.

Rissland, E. L., and Daniels, J. J. 1996. The Synergistic Application of CBR to IR. *Artificial Intelligence Review* (Special Issue on the Use of AI in Information Retrieval) 10(5–6): 441–475.

Rissland, E. L., and Skalak, D. B. 1989. Combining Case-Based and Rule-Based Reasoning: A Heuristic Approach. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 524–530. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Rissland, E. L., and Skalak, D. B. 1991. CABARET: Rule Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies* 34(6): 839–887.

Rissland, E. L., and Soloway, E. M. 1980. Overview of an Example Generation System. In Proceedings of the First National Conference on Artificial Intelligence, 256–258. Menlo Park, Calif.: American Association for Artificial Intelligence.

Rissland, E. L.; Valcarce, E. M.; and Ashley, K. D. 1984. Explaining and Arguing with Examples. In Proceedings of the Fourth National Conference on Artificial Intelligence, 288–294. Menlo Park, Calif.: American Association for Artificial Intelligence.

Sabater, J.; Arcos, J. L.; and López de Mántaras, R. 1998. Using Rules to Support Case-Based Reasoning for Harmonizing Melodies. In *Multimodal Reasoning: Papers from the 1998 AAAI Spring Symposium,* 147–151. Menlo Park, Calif.: AAAI Press.

Skalak, D. B., and Rissland, E. L. 1992. Arguments and Cases: An Inevitable Intertwining. *Artificial Intelligence and Law: An International Journal* 1(1): 3–48. Reprinted in Skalak, D. B., and Rissland, E. L. 1998. *The Philosophy of Legal Reasoning: Scientific Models of Legal Reasoning,* ed. S. Brewer, 249–290. New York: Garland.

Smith, I.; Lottaz, C.; and Faltings, B. 1995. Spatial Composition Using Cases: IDIOM. In *Proceedings of the First International Conference on Case-Based Reasoning,* 88–97. Berlin: Springer.

Sqalli, M., and Freuder, E. 1998. Diagnosing Interoperability Problems by Enhancing Constraint Satisfaction with Case-Based Reasoning. Paper presented at the Ninth International Workshop on Principles of Diagnosis (DX-98), 24–27 May, Cape Cod, Massachusetts.

Sqalli, M. H.; Purvis, L.; and Freuder, E. C. 1999. Survey of Applications Integrating Constraint Satisfaction and Case-Based Reasoning. Paper presented at PACLP99: The First International Conference and Exhibition on the Practical Application of Constraint Technologies and Logic Programming, 19–21 April, London, United Kingdom.

Veloso, M. 1994. *Planning and Learning by Analogical Reasoning.* Berlin: Springer.

Wallace, M. 1996. Practical Applications of Constraint Programming. *Constraints* 1(1): 139–168.

Weigel, R., and Faltings, B. 1998. Interchangeability for Case Adaptation in Configuration Problems. In *Multimodal Reasoning: Papers from the 1998 AAAI Spring Symposium,* 69–73. Menlo Park, Calif.: AAAI Press.

Zeleznikow, J.; Vossos, G.; and Hunter, D. 1994. The IKBALS Project: Multimodal Reasoning in Legal Knowledge-Based Systems. *Artificial Intelligence and Law* 2(3):169–203.

Cynthia Marling is an assistant professor in the School of Electrical Engineering and Computer Science at Ohio University. She received her M.S. and Ph.D. in computer science from Case Western Reserve University. Between degrees, she worked as a knowledge engineer and project leader for General Electric. Her research interests are case-based reasoning, multimodal reasoning, robotic soccer, and AI in medicine. Her e-mail address is marling@ohio.edu.

Mohammed Sqalli is a senior automation testing specialist at the Telecom Innovation Center in Siemens Canada Limited. Sqalli received his M.S. in computer science from the University of New Hampshire in 1996 and his Master of Engineering in computer science from the Mahammadia Engineering School, Rabat, Morocco, in 1992. He is currently a Ph.D. candidate in Systems Design Engineering at the University of New Hampshire. His dissertation research involves integrating constraint satisfaction and case-based reasoning. His e-mail address is msqalli@cs.unh.edu.

Edwina Rissland is professor of computer science at the University of Massachusetts at Amherst. For many years, she was also a lecturer on law at the Harvard Law School, where she taught a course on her specialty of AI and law. She received her Sc.B. from Brown University and her Ph.D. in mathematics from the Massachusetts Institute of Technology. She has long been interested in how people and machines reason with examples and cases and was one of the founders of case-based reasoning. Her e-mail address is rissland@cs.umass. edu.

Hector Muñoz-Avila is an assistant professor in the Computer Science and Engineering Department at Lehigh University. He received his B.S. and M.S. in computer science and B.S. in mathematics from the University of the Andes in Columbia and his doctorate from the University of Kaiserslautern in Germany. Previously, he was a researcher at the University of Maryland at College Park and at the Navy Center for Applied Research in Artificial Intelligence. His e-mail address is munoz@cse.lehigh. edu.

David Aha leads the Intelligent Decision Aids Group at the Navy Center for Applied Research in Artificial Intelligence, a branch of the Naval Research Laboratory. His research interests currently focus on mixed-initiative planning and intelligent lessons learned systems. Aha is an editor for the *Machine Learning* journal, is an editorial board member for *Applied Intelligence*, and was program cochair for the 2001 International Conference on Case-Based Reasoning. His e-mail address is aha@aic.navy.mil.