Artificial Intelligence and Law **1:** 113–208, 1992.

Case-Based Reasoning and its Implications for Legal Expert Systems

KEVIN D. ASHLEY

Assistant Professor of Law and Intelligent Systems,
University of Pittsburgh,
School of Law
and
Learning Research and Development Center,
Pittsburgh, PA 15260, U.S.A.
e-mail: ashley@vms.cis.pitt.edu
tel: (412) 648–1495, 624–7496

(Received 16 November 1991, accepted 8 October 1992)

Abstract. Reasoners compare problems to prior cases to draw conclusions about a problem and guide decision making. All Case-Based Reasoning (CBR) employs some methods for generalizing from cases to support indexing and relevance assessment and evidences two basic inference methods: constraining search by tracing a solution from a past case or evaluating a case by comparing it to past cases. Across domains and tasks, however, humans reason with cases in subtly different ways evidencing different mixes of and mechanisms for these components.

In recent CBR research in Artificial Intelligence (AI), five paradigmatic approaches have emerged: statistically-oriented, model-based, planning/design-oriented, exemplar-based, and adversarial or precedent-based. The paradigms differ in the assumptions they make about domain models, the extent to which they support symbolic case comparison, and the kinds of inferences for which they employ cases.

Reasoning with cases is important in legal practice of all kinds, and legal practice involves a wide variety of case-based tasks and methods. The paradigms' respective benefits and costs suggest different approaches for different legal tasks.

CBR research and development in the field of AI and Law should be pursued vigoriously for several reasons. CBR can supplement rule-based expert systems, improving their abilities to reason about statutory predicates, solve problems efficiently, and explain their results. CBR can also contribute to the design of intelligent legal data retrieval systems and improve legal document assembly programs. Finally, in cognitive studies of various fields, it can model methods of transforming ill-structured problems into better structured ones through the use of case comparisons.

1. Introduction

Researchers in the Artificial Intelligence subfield of Case-Based Reasoning have accomplished a great deal in the last few years, and much of that work has significant ramifications for research in AI and Law. They have explored different domains and tasks, identified different modes of reasoning with cases and invented different designs and implementations. It is a good point in the intellectual development of the field to step back and compare their approaches and accomplishments and try to assess the implications for the development of legal expert systems.

This paper has three goals, to: (1) synthesize important CBR research developments; (2) assess what their implications are for building law-related expert systems and AI programs to model legal reasoning; and (3) argue that AI/CBR work in the legal domain can make an important contribution to the development of practical systems for the legal profession and to practical and intellectual advances in AI and cognitive modeling.

Conceptually, the paper is divided into three parts. The first part begins with a discussion of why humans, in and out of the legal field, reason with cases and why expert systems need to (Sections 1.1 and 1.2), provides some useful definitions, and relates the CBR field to AI's general concerns and alternative reasoning methodologies.

In lieu of a general theory of case-based reasoning, the second part presents five paradigmatic kinds of Case-Based Reasoning and illustrates each with a program, most of which are the subjects of recent doctoral dissertations (Section 2). This part concludes with a comparison of the CBR paradigms in terms of thirteen criteria (Section 3).

By this point, a foundation has been laid for a careful discussion of the implications of CBR to AI and Law. In the third part, I present five practical and intellectual reasons for pursuing a CBR approach to research in modeling legal reasoning and development of legal expert systems (Section 4). CBR, it is argued, can supplement rule-based expert systems, improving their abilities to reason about statutory predicates, solve problems efficiently, and explain their results. CBR can also contribute to the design of intelligent legal data retrieval systems, improve legal document assembly programs and contribute to Cognitive Science models of the use, in various fields, of case comparison methods to transform ill-structured problems into better structured ones. In the course of the discussion, I elucidate the paradigms' significance for modeling legal reasoning and discuss the applicability of relevant, very recent work in CBR.

1.1. WHY REASON WITH CASES?

A reasoner performs CBR when he, she, or it compares a problem to prior cases in order to draw conclusions about a problem and to guide making decisions. The primary goals of AI research in CBR are to identify the ways that human reasoners draw conclusions from comparing cases, and, where possible, to design computer programs to implement these methods or to assist humans to perform them.

AI research and expert systems development should incorporate CBR techniques because human experts in diverse fields reason with cases in performing a wide variety of tasks. The tasks are not rare or exotic. They represent a significant core of human expert advice-giving in many domains. Since, for each task, legal applications come readily to mind, designers of legal expert systems will also need to take cases into account.

Classification and Diagnosis: Human experts often decide how to categorize a new case by comparing it to previously categorized cases. Case comparisons are especially important where the analytic model of the classification task is not strong enough to support deductive classification. A problem's features may lead (via an index) to previously classified cases with similar features. By comparing the similarities and assessing the

significance of the differences in light of whatever classification theory is available, one may reasonably decide whether or not to apply the same category to the new case.

Diagnosis is a special kind of classification which involves not only categorizing a new case but also explaining why the new case features justify the categorization in terms of some analytical theory or model. Prior case explanations serve as patterns for explaining a new case. Considering an array of similar past cases with different diagnoses can appraise the reasoner of alternative classifications of the new case and help to focus on the most likely diagnosis.

Attorneys in all kinds of legal practice employ cases to make classifications. The classification decisions may involve almost anything: whether a new product or service fits a regulated category, whether a regulation fits a category of constitutionally valid regulations, what kinds of claims to pursue in connection with an injury, or how to characterize a corporate reorganization for tax purposes. Cases, defined broadly to include any authoritative past classifications, are useful in this connection. Where those cases have been explained in terms of some analytical model, such as in court opinions attorneys may engage in a kind of diagnosis, attempting to determine if the explanation can or ought to apply to a new problem.

Planning and Design: Planners reason by comparing past plans: Successful plans are models or templates for constructing a new plan to address a current problem; past failures are reminders of planning approaches to be avoided. Probably, cases also play a significant role in strategic planning, for instance, of sales campaigns, negotiation strategies or managing military forces. In deciding whether to adopt a plan, strategic planners often consider scenarios for the worst case, the best case, or the most likely cases. Design is a kind of planning. Given a set of constraints to satisfy, human designers often look to previous designs that satisfied some or all of them and adapt or combine these cases to fit the remaining constraints while avoiding the pitfalls of previously encountered design failures.

Legal planners reason from past plans in a variety of settings. In devising a plan to seek regulatory approval for a proposed action, attorneys may refer to past successful and unsuccessful attempts to determine the steps to take or avoid. Litigators look to judicial opinions not only as justifications, but also in devising plans and strategies for trial: what kinds of evidence are relevant? How can the evidence be obtained? By what standards will the evidence be judged? In planning strategies for an oral argument, and advocate attempts to anticipate the judges' concerns and reactions by engaging in practice strategy sessions with his or her partners: How shall I deal with my opponent's best case? What kinds of hypotheticals are the judges likely to pose? In negotiation, attorneys look to past negotiations to plan strategies, assess a party's range of acceptable conditions, formulate possible settlements, and persuade parties to accept. A recent negotiation in a related industry may motivate (and justify) seeking a hither-to unavailable goal, such as employee give-backs or a four-day work week.

Frequently attorneys are designers, too. Legal drafters often reuse plans for drafting contracts. They scope out the design by looking at past documents and modify them to

satisfy specific constraints of a new problem. Law firms and corporate legal departments maintain libraries of previously designed agreements for reuse.

Evaluation, Assessment and Extrapolation: Human experts are often called upon to assign a monetary value to a house, business, or business proposal. Frequently, the theories of valuation in the domain do not support deducing what the value should be; the calculations may be too complicated, time-consuming or expensive to apply. Instead, the theory of valuation is case-based. Expert appraisers draw comparisons to past cases of known value in order to extrapolate a value for the new case. For instance, starting with the most recently sold houses on the block, a real estate appraiser estimates a higher or lower price taking into account any specific differences (e.g., increase price for extra bathrooms or decks; decrease for damp basements, flaking paint, etc.) and also general trends (e.g., rapid price appreciation due to recent mortgage interest rate reductions.) Where evaluation is more complicated, for instance, estimating the cost/benefit tradeoffs of a proposed design, experts may employ prior evaluated cases to demark the end points of intervals in the value curve. By placing a current problem between some past cases, experts can set bounds on the range of possible values and extrapolate an intermediate value.

Frequently, attorneys need to assign values to things such as law suits and injuries, for instance in deciding how much to claim in damages, for how much to settle, or in allocating litigation resources. Often that evaluation is based on prior cases. For instance, the 'Damages' section of West's *Digests*, a comprehensive collection of legal cases indexed by topic, records damage amounts in cases where a party asserted the awards were inadequate or excessive. The cases are indexed by injuries, such as 'loss of leg or foot', 'loss of both legs', 'loss of fingers to toes', etc. Attorneys also extrapolate settlement values from past settlements. In defending complex, industry-wide law suits, defense counsel employ past settlements to convince judges of the reasonableness of proposed settlement figures. Counsel extrapolate values adjusting for perceived differences in the strengths of plaintiffs' cases and their chances of success. For instance, not long ago, companies which sold shares to consumers and invested the proceeds in commercial real estate ventures (i.e., Real Estate Investment Trusts or REIT's), were hit hard by a depression in commercial real estate values. Many investors suffered substantial losses and sued the REIT's for securities fraud, alleging that the REIT's fraudulently inflated the values of their real estate. Since the suits were class actions brought on behalf of many consumer investors, attorneys for the REIT's had to convince federal judges of the reasonableness of proposed settlements which the attorneys ultimately did by extrapolating values from past settlements of similar suits. This is an example of case-based evaluation in another sense. The REIT suits often turned on the failure of their investment company managements to employ approved case-based real estate appraisal methods. They failed accurately to assess the values of properties on which they extended loans because they did not base those values on recent sales of like properties.

Justification and Argumentation: Decision makers often rely on precedents, authoritatively decided past cases, to justify similar decisions in analogous circumstances. Where

domain theories are too weak to support authoritative logical proofs of a decision's correctness, a justification by analogy to a precedent supports a decision's reasonableness. Since there often are many precedents evidencing various analogical circumstances and competing outcomes, there often are competing justifications and an argument ensues.

Arguing with case precedents is a paradigmatic form of justification in law. 'The basic pattern of legal reasoning is reasoning by example. It is reasoning from case to case.' (Levi 1949, p.1). Especially in Common Law jurisdictions like the United States and the United Kingdom, arguing with cases has been developed institutionally, professionals share expectations about how the arguments should be constructed rhetorically and, to a lesser extent, there are common standards for evaluating the arguments (Levi 1949; Llewellyn 1930, 1989; Radin 1933; Burton 1985, pp. 11–24). Attorneys rely on cases to justify positions on procedural, as well as substantive, legal issues. In law, cases are even important in justifying assertions about the basic rules of the game, procedural questions of a court's jurisdiction, how to commence an action, discovery, conduct of trials, evidential matters, judgments and appeals. Informally, case-based arguments play a primary role in analyzing and evaluating problems and planning for their avoidance.

Explanation and Persuasion: Experts often explain their advice with cases as examples. Cases illustrate concepts, focus attention, demonstrate alternatives, and help to distinguish the problem from close but different situations. Even in domains where case-based argument is not formalized, experts may seek to persuade a client that the advice is correct and valuable by reciting a number of similar cases with favorable outcomes or cases where the advice was ignored and the outcomes disastrous.

Attorneys of all stripes, not just advocates in court, employ case examples to explain and persuade. Legal planners may explain how a deal was structured or an agreement drafted especially to avoid the problems or take advantage of opportunities presented in a particular case or paradigmatic example. Cases in legal explanations need not be technical; indeed case examples are a primary way of communicating with clients untutored in 'legalese'. It is hard to explain a complicated estate plan without examples. Attorneys who draft trusts and wills for clients often illustrate alternative distribution plans and provisions in light of stereotypical cases. Such cases may illustrate the appropriateness of the plans or provisions given different assumptions about the client's circumstances, physical and mental condition, expected size and nature of the trust's assets, his or her marital situation and intentions to benefit family members.

Interpretation: In many domains, rules are not well-defined. Terms are ambiguous and rules have exceptions or even competing formulations. Experts employ cases to interpret rules. They compare a problem situation to previous cases where the rule was or was not applied and argue by analogy for a similar result in the current case. They compare a problem situation to well-known exceptions to the rule before deciding that the rule applies. They describe the purposes of a rule by describing the paradigm case it was intended to govern.

Lawyers engaged in statutory interpretation employ cases to draw inferences about what legal rules mean (Llewellyn 1938; Burton 1985, pp. 68–77). 'It is only folklore which holds that a statute if clearly written can be completely unambiguous and applied as intended to a specific case.... [A]mbiguity is inevitable in both statute and constitution as well as with case law. Hence reasoning by example operates with all three.' (Levi 1949, p. 6). Such interpretations are important not only in formal legal arguments but also in informal legal planning. Corporate legal planners pay particular attention to trends in a government agency's interpretation of a regulation as evidenced by how the agency has applied it in the recent past. For instance, the Patent and Trademark Office's liberal treatment of patent applications on software-related inventions has been a barometer of change in the interpretation of legal rules governing the patentability of software. Its interpretations, at odds with those of courts, are a leading indicator and a sounder basis for corporate intellectual property planning.

Learning and Teaching: Cases represent experience and, thus, play a primary role in learning. From past cases people learn to avoid failures and to adopt successful plans as appropriate. Much scholarship involves selecting, organizing and inducing the lessons implicit in past cases in light of, or in contrast to, existing theories of the domain. Even after generalizations are formulated, past cases help to interpret them or to modify them as even newer cases arise. In a broad sense, the law institutionalizes learning from experience. Cases are retained, indexed, retrieved, reused, and reinterpreted. 'The rules change as the rules are applied. More important, the rules arise out of a process, which while comparing fact situations, creates the rules and then applies them.' (Levi 1949, p. 4). The generalizations that express what has been learned cannot be separated from the cases they abstract. Cases provide meanings for the generalizations; practitioners employ the cases to operate with and on legal concepts, to explain and interpret them.

Cases are a primary tool for teaching professional judgment. A case method is the teaching style of choice in law, business, accounting, appraisal, ethical reasoning, and certain medical skills. By leading novices to compare and contrast strategically selected cases, professors hone analytical skills and force students to grapple with the subtleties and problems of applying general domain knowledge in specific contexts. A Socratic, case-based method in legal education teaches novices to treat legal rules skeptically, to appreciate the rules' *lack* of certainty, and to employ cases to justify assertions about what rules mean and how they apply.

Discovery, Theory Building, and Testing: Cases play an important, if not well understood, role in the discovery of new theories and rejection of old ones (Kuhn 1970). Analogies from past explained cases of other domains may suggest new categories or rationales. In inventing new legal categories attorneys reason by analogy to categories applied in previous cases. Examples of such invention may be found in many legal fields. In commercial and banking law, attorneys invented NOW accounts by adapting ordinary savings bank withdrawal slips into check-like negotiable orders of withdrawal so as to avoid legal restrictions on interest-bearing checking accounts.[1] New 'hybrid' forms of investment or

tax shelters combine, vary, or eliminate features of existing forms. New court-made rules employ predicates invented by analogy to concepts employed in precedents. For instance, a noxious example involved the decision of the Supreme Court in *Mayor v. Miln*, 36 U.S. 102 (1837) to uphold a city regulation which was alleged to be invalid under the Constitution. The regulation required masters of vessels arriving from foreign ports to provide detailed information about immigrants on board. The Court decided the regulation was a valid exercise of the States' Police Power and justified its decision by an analogy to a prior case: just as state authorities could legally regulate 'physical pestilence' such as by quarantining vessels whose crew had infectious diseases, it could regulate 'moral pestilence' by determining if the immigrants were 'paupers, vagabonds, and possibly convicts' (Levi 1949, p. 68), evidence that the results of such case-based invention are not always worthy.

Experts test old theories and new hypotheses against past or hypothetical cases. The testing process is dialectical: a generalization is tested against a hypothetical case which leads to reformulating the generalizations and still other tests (Lakatos 1976). Attorneys employ real and hypothetical cases to test the limits of legal assertions. Appellate judges pose hypothetical scenarios to assess how a proposed rule for deciding a case will affect future decision making: 'If we draw the line as you suggest, counselor, will we be powerless to deal with the next case that comes along presenting even more extreme circumstances?' Legislative drafters pose hypothetical scenarios to try to anticipate unintended effects of proposed statutory language. Corporate drafters pose hypotheticals to attempt to anticipate how a change in the formulation of a standard contract may effect the way that business parties will interpret existing contracts with a client.

'How we handled this issue in the past' is likely to be very useful information, despite differences in legal systems. As the breadth of these examples suggests, even in Civil or Continental legal jurisdictions, which do not employ cases formally as justifications in arguments or a means for interpreting statutes, cases still may be important in other legal tasks such as planning, explanation, design, and evaluation. Presumably Civilians also reason from past plans, employ case examples to explain advice, and evaluate settlements in a case-based way. The cases of interest may not be reported opinions, but past episodes of negotiating a deal or drafting a contract or applying for regulatory approval.

Synthesizing from the examples of case-based expert tasks, CBR offers natural techniques for realizing six goals of expert systems development. Examples illustrating how each goal has been realized are presented in the sections listed:

1. *Compiling Past Solutions*: Solving a problem from scratch may be computationally expensive. To increase efficiency, such solutions may be complied and reused. If a past solution's conditions of applicability can be abstracted as an index, a reasoner does not have to reinvent the wheel but can find the past solution when it becomes relevant again and adapt it to the constraints of the current problem (see Sections 2.2 and 4.6)

[1] See Consumer Savings Bank v. Commissioner of Banks, 282 NE2d 416 (Mass. 1972).

2. *Avoiding Past Mistakes*: At least, a reasoner should be able to avoid its past failures. If the conditions that caused the failure can be abstracted as an index, a reasoner does not have to explore the same deadend twice. Avoiding past mistakes increases both efficiency and accuracy (see Section 2.3.2)

3. *Interpreting Rules*: Sometimes, a knowledge engineer defines a rule's predicate at his/her peril. There may be no authoritative definition, the rule may have exceptions, the predicate may be deliberately ambiguous, or experts may differ as to the formulation. Cases can help to interpret the rule. By annotating a rule with its positive case examples or exceptions, the reasoner can draw reasonable conclusions about whether a predicate applies in light of competing arguments by analogy to past cases where the predicate did or did not apply (see Sections 4.3.1 and 4.3.2).

4. *Supplementing Weak Domain Models*: In real domains, available theory is often too weak to support a deductive analysis. There is no one right answer. The best that an expert system can do is to support a decision by apprising the decision maker of reasonable alternative answers based on the treatment of past cases. Comparing the problem in detail to selected past cases may focus the reasoner on fundamental differences that counsel an alternative approach to the current problem (see Sections 2.4, 2.5.1 and 2.5.3)

5. *Facilitating Explanation*: Examples can help an expert system explain its advise, a skill at which such systems are notoriously deficient. Typically, an expert system's explanation is organized around a trace of the rules that fired in generating a conclusion. Even when augmented by general principles or strategies, however, the traces either do not provide the right information or too much information and at the wrong level of detail. Tailoring general explanations to a specific problem-solving context has proven difficult, as well. A program that records case examples could use them to illustrate its analysis and advice. By examining the cases, the human user can assess whether the program's advice is appropriate. A system can illustrate terms, as well as define them, and demonstrate the intended application of a rule in a specific, similar context (see Section 4.4).

6. *Supporting Knowledge Acquisition and Learning*: In a simple sense, as case-based expert systems acquire and index cases, they learn from experience. As we will see, fully automating the process of acquiring and indexing cases presents some challenging learning issues. Nevertheless, expert systems have begun to reflect the significant uses of cases in human knowledge acquisition, learning and teaching (see Sections 2.3.2 and 2.5.2). Even the standard methodology for acquiring knowledge from a human expert and representing it as rules involves posing real and hypothetical cases to elicit and debug the rules. If those cases are retained, they can be used to provide illustrations and exceptions to the rules. They may also assist in revising the rules; the reviser should know the case situations with which the rule originally was intended to deal (see Section 4.4.3)

1.2. CASE-BASED REASONING IN AI

At this point in the development of CBR as a science, it may not be fruitful to attempt to set out a general theory of reasoning with cases. As enumerated above, the uses of cases are very diverse; even similar uses of cases differ widely across domains depending on how domain experts draw inferences from case comparisons and on the nature of the theory available in the domain. Instead of a general theory, in this paper, I examine five general paradigms of CBR and compare and contrast them in terms thirteen important criteria. As a prelude for discussing the paradigms, here are some useful definitions and observations about reasoning with cases.

1.2.1. *Definition of 'Case'*

A 'case' is a particular set of empirical circumstances presenting a problem for decision, solution or classification as an instance of a type. It has particularity, presenting the circumstances and situation of a discrete episode, action, person, or thing. It is empirical in that it deals with something that actually exists or happens. It presents a problem for decision, that is, circumstances constituting a matter for consideration and analysis resulting in an outcome. Finally, a case is an instance or example of the particular type of outcome, the category under which it has been classified, or the concept which the decision maker has applied. This definition's requirement that a case be 'empirical' needs amending to admit hypothetical cases. Hypothetical cases are problematic because, by definition, a hypothetical case has not happened. Hypothetical cases are interesting, however, in so far as they could happen; they are intentionally constructed to raise an issue in circumstances that might, in principle, be realized.

In practically all of the CBR systems I discuss, a case is represented as having a particular name, a set of empirical circumstances or facts, and an outcome representing the result of the problem for decision, solution, or classification it poses. Frequently, as part of its outcome, a case will have been deemed to be an instance of a type or concept and will be indexed accordingly. Depending on the kind of inferences the program draws from cases, a case may also record an explanation of the outcome, the method of solution, and the context in which solution decision steps were taken. Frequently, cases are linked to other cases, either directly or via the index.

1.2.2. *Case-Based Inference*

At its most basic, all case-based reasoning involves a kind of table look up. The table is an indexed store of cases which will be referred to as the CBR system's case database. Historically, the case database has also been called the CBR system's memory or its Case Knowledge Base (CKB), terms which focus on the need to organize and index cases in the database for purposes of retrieval. Much as we might look up telephone numbers of 'Attorneys' or 'Computers-Service and Repair' establishments in the index of the *Yellow Pages*, a CBR system 'looks up,' in the index of its CKB, past cases that bear on the current problem. Instead of being an alphabetical index of services, a CBR system's

index includes information, abstracted from cases, which represents the circumstances in which the case has some utility in problem analysis or solution. In other words, the index terms represent a case's relevance. Employing the index, the program retrieves and selects the most similar cases to the problem and applies them in a solution.

Start: Problem description.
A: Process problem description to match terms in case database index.
B: Retrieve from case database all candidate cases associated with matched index terms.
C: Select most similar candidate cases not yet tried.
  If there are no acceptable candidate cases, try alternative solution method, if any, and go to F.
  Otherwise:
D: Apply selected best candidate cases to analyze/solve the problem. If necessary, adapt cases for solution.
E: Determine if case-based solution or outcome for problem is successful.
  If not, return to C to try next candidate cases.
  Otherwise:
F: Determine if solution to problem is success or failure, generalize from the problem, update index accordingly and Stop.

Fig. 1. General scheme for a case-based reasoner.

A general process for reasoning with cases is provided in figure 1. Although CBR systems display wide variations in design – not all CBR programs perform all of these steps, some programs implement others steps and some of the steps, particularly the last one, raise difficult technical problems – they usually are variations of this scheme.

Case-based reasoning starts with a description of the problem. The program processes the description to match terms in the index (A), retrieves the indexed cases (B), and selects the most similar candidate cases that have not yet been tried as the basis of a solution (C). If there are no acceptable candidate cases, the program should try any available alternative method for solving the problem. Otherwise, having selected the best case or cases, the program attempts to use them to create a solution or comparative analysis of the problem (D). This often involves adapting the solution from a selected case to the problem's constraints. Next, the program determines whether the case-based solution is a success or failure (E). This step may involve the program's simply seeking the user's feedback; otherwise it presupposes some computational way to assess a solution's distance from the goal or to determine that too much time has been expended on adaptation. If a failure, the program should try the next best candidate cases as possible solutions (Return to C). Otherwise, the program should go on (to F) to determine whether the problem has been solved by either a case-based or alternative process (success) or whether there is no solution (failure). In either instance, it may be valuable to add the new problem to the CKB. If a success, the newly solved case may be used to solve problems. If a failure, the attempted solution may be avoided. Adding the new case to the CKB entails abstracting from it some features for indexing, for instance, by constructing an explanation of why it is a success or failure. Possibly, new terms may have to be added to the index.

CBR programs employ two basic kinds of case-based inference in solving or analyzing a problem at step D: (1) Constrained Search or (2) Comparative Evaluation.

A past case may constrain the search for a solution by providing a solution template. In this mode, a program maps a past solution onto a problem. The past solution may have been constructed painstakingly from scratch by hand or by a computationally expensive method. Once constructed, however, the solution may make future problem solving more efficient. In general, if the index adequately captures the applicability conditions of the past solution, the database contains a sufficiently similar case, and there are computationally reasonable means to adapt the past solution to the current constraints and check the solution's correctness, case-based problem solving may be more efficient than solving the new problem from scratch.

Alternatively, the CBR program may draw a case-based inference by comparative evaluation. A reasoner evaluates a current problem by comparing and contrasting it with evaluated past cases. Comparisons with specific cases serve as the basis for an argument that the problem should be assigned a particular value. Symbolic or numeric values may be assigned. The value may be an outcome justified by an analogy to a past case or by a 'disanalogy' to a past case with a different outcome. Alternatively, the value assigned to the thing may be a dollar amount extrapolated from comparing it to similar previous cases of known value.

Some domain tasks require combining constrained search and comparative evaluation. For instance, a design program could employ design cases as both templates to adapt to new constraints and as datapoints with which to evaluate the cost of a new design.

1.2.3. *CBR and Fundamental AI Concerns*

In light of the case-based reasoning process, one may see how CBR relates to the fundamental AI concerns of search, representation, and control, and to other approaches to reasoning in AI. CBR is intended to make its biggest contribution in the matter of search; theoretically a case-based approach minimizes search by means of its index to relevant past cases. Ideally, a more or less weakly guided search through a space of operators and bindings is replaced by a kind of table look up; the table is the database of indexed cases. The matching and retrieval of cases and selection of the most similar cases (A, B, and C) are intended to retrieve a case that presents very nearly a solution to the problem.

In practice, search is not entirely eliminated. A retrieved case may be relevantly similar to the problem without being identical. Search is reintroduced in adapting the retrieved case to the constraints of the current problem (D), a process that may involve a more or less weakly guided search through a space of adaptation operators and bindings. For those interested in deploying CBR systems for commercial manufacturing and design, probably the most important research problem in CBR is determining how to control that adaptation search, heuristically or otherwise.

The major knowledge representation issue in CBR is how to represent cases and design an index in such a way that a program can assess relevance effectively (steps B and C). Features of cases that are relevant to its outcome should be incorporated into an index, but which features are they? Are they functional or structural features or

something else? How does one define and assess the relevant similarities and differences among cases? When is a match close enough? If it is only a partial match, are the differences relevant? How can the best of a number of competing more-or-less similar cases be selected?

Coping with the representation issue entails striking some balance between generalization and particularity. Since almost all case-based reasoners employ indices, generalizations are almost always present and serve two functions, as: (1) terms indexing cases and (2) measures, weights or indications of a feature's significance in assessing case relevance. Usually, cases are indexed by feature types, solution types, goals, or other concepts which have been abstracted from groups of cases or from the explanations or solutions of individual cases. Although abstracting such concepts may employ inductive or statistical methods (and some degree of automation has been achieved as discussed below), often it falls to a human expert to determine the predictive features of a case and the circumstances in which the case will have utility. The human expert may also be asked to provide some indication of the features' weights, represented either symbolically or numerically. (If weights are represented numerically, the system may revise them according to the feature's utility in past problem solving).

Though such generalizations are essential for constructing an index and finding relevant case, inevitably the generalizations may mislead a reasoner: it is impossible to state all of the conditions of a case's applicability. Features which generally are decisive may not be in the circumstances of a particular problem. Similarities or differences which seem irrelevant when described at one level, may be profound when described at another level or for a different purpose.

Relevance, in other words, depends highly on context. The question of how and when to take context into account is one of the most important in CBR research. Human experts are facile at recognizing the importance of particular features in context, determining when a problem should be treated as an exception to a general rule or drawing conceptual analogies between cases that are superficially quite different. CBR program's strategies for switching focus appropriately from generalizations to a case's particular facts and back again are but crude approximations. The computational problem of dealing with context is compounded as case representations become more complex and the number of cases becomes large. Nevertheless, systems designers may still come to some accommodation by selecting tasks appropriate to the program's often limited ability to take context into account.

Other control problems of current interest in CBR include deciding whether to resort to cases or to some other solution process (Step C in Figure 1. One could imagine making this decision at other points), ensuring that adaptations of a retrieved case converge on an acceptable solution (step D, the search control problem referred to above), deciding whether to apply and adapt a retrieved case to the problem or to keep looking for a more similar case (adaptation can be computationally very expensive), and assigning credit and blame in attempting to explain why a case represents a success or failure (F). Learning relevant features and appropriately indexing a case depend on a robust capability for credit assignment.

1.2.4. *CBR and Other AI Reasoning Modes*

CBR and Analogical Reasoning: CBR is closely linked to analogical reasoning. Comparing a problem and case leads to an analogical inference: since they are similar in certain respects, the problem should have the same outcome as the case. Probably not all analogical reasoning is case-based. For instance, although it is a debatable point, analogical reasoning tests, in which a student is asked to select the object D which is related to C as A is to B, do not seem to me to be case-based. Typically, a Miller's analogy question involves not reasoning from a body of possibly relevant cases, but only from two cases, which are relevant by definition, and the unstated relationship between them.

In addition, probably not all case-based reasoning is analogical, particularly the more statistically oriented kind described below. Indeed, some researchers on analogical reasoning commonly insist that analogical reasoning involves transferring information across some kind of domain boundaries. Much AI research in analogical reasoning has focused on drawing analogies across domains, for instance, from fluid mechanics to electricity. Although some CBR approaches involve such cross-domain transfers, most do not. In deference to this viewpoint, I will try to reserve the term 'analogical reasoning' for situations involving a transfer of information from one domain to another. In typical AI work on analogy, it is also the case that a reasoner maps the structure of a problem solution in one causal model to a problem in the other domain; structural consistency (isomorphism) in the problems is critically important, as is the level of abstraction of description of the problems, since it affects pattern matching. Some CBR approaches map structured solutions across cases in this way, but, as discussed below, there are other ways of comparing cases beside structurally.

CBR and Deductive Reasoning: CBR is usually contrasted with logical deductive or rule-based reasoning in which *modus ponens* is the inference mechanism. For one thing, CBR uses different inference mechanisms: constrained search or comparative evaluation. For another, CBR is a kind of nonmonotonic reasoning. Assumptions and conclusions based on generalizations may have to be abandoned once the reasoner takes into account the specific facts of a past case in comparison to the problem. For instance, if a problem matches not only a rule's antecedents but also matches a case exception to the rule, the rule's conclusion may have to be abandoned.

CBR is an alternative to deductive or rule-based reasoning for domains that lack strong analytic models. A domain theory may be said to be strong to the extent that rules or explanations can be applied deductively. To the extent that the rules cannot be applied deductively, cases conceptually indexed and employed to fill in the gaps may be thought of as an alternative to a strong domain theory.

Comparative evaluation with cases does not assume a strong domain theory; indeed, its primary utility is as a method for assigning a value reasonably despite the lack of a strong domain theory. (As we will see, however, comparative evaluation with cases does assume some domain theory). For instance, there is no theory for deducing the value of real estate, but there are some systematic means for estimating real estate values based on

comparing the property to similar recently-sold properties, accounting for any differences in the properties' features, like recent improvements, and correcting for appreciation (or depreciation) since the sale date.

In fact, CBR may complement logical or deductive reasoning. As discussed below, reasoning with a rule's exceptions can improve the performance of a rule-based reasoner. Where rule antecedents are ambiguous, CBR furnishes a method for evaluating whether the predicate is satisfied or not by arguing from cases. To the extent that the rules cannot be applied deductively, cases conceptually indexed and employed to fill in the gaps may be thought of as complementing the strong domain theory.

A constrained search case-based approach may be compatible with either a strong or weak domain theory. Even where stronger analytic methods such as deduction are available, solutions may still involve a lot of search. Successful solution paths may be compiled and stored as cases and used as templates for solving new problems, shortcutting the problem-solving search. Where only weak methods for solving a problem are available, constrained search may still be useful; the case records the application of the weak methods in a compiled solution that may be applied again.

CBR and Inductive Reasoning: The goal of inductive reasoning is to abstract decision rules from positive and negative instances of a concept. The rules which may then be applied deductively to solve or classify problems. Cases, of course, may be the positive and negative examples from which the program learns decision rules. The induction process may involve lots of examples or only one and employ statistical, model-based, or explanation-based methods. Ultimately, however, its goal is to eliminate the cases from which the rules were derived.

The goal of CBR is quite different: to create an indexed body of cases from which to retrieve relevant cases that can be applied to the analysis of a problem via constrained search or comparative evaluation. Constructing an index may involve some inductive reasoning in a broad sense. For instance, a cluster of features induced from a set of cases may be employed as an indexing term as may a concept derived from an explanation-based analysis. But the index terms are used to retrieve a case whose solution may then be applied to a problem analogically.

Even a purely inductive method, however, whose goal is to derive a set of decision rules, may benefit from integrating a case-based component. If the cases were retained, they could be used as examples for explaining the resulting rules and as justifications for the rule's existence. In realistically complex domains, it is possible that no induction process could totally reify the task into rules. Some cases would need to be retained as counterexamples to a rule. Others might be used to support interpretations of the rule through a process of comparative evaluation. Still others might serve as test cases when a rule needs to be modified. As new cases come along and necessitate changes in the rules, presumably, it would be desirable to have access to the cases from which the rules were derived originally.

2. Five CBR Paradigms with Examples

For pedagogical purposes, CBR programs can be grouped into five paradigms, each with a representative program. Each paradigm corresponds to a typical use of cases in a domain task. The paradigms are not mutually exclusive. In fact, each of the representative programs evidence aspects of one or more of the other paradigms. Nevertheless, the typology serves to focus on some important kinds of reasoning with cases and the computational approaches to modeling them. Most of the programs selected are Ph.D. dissertation projects, so that this synthesis serves also as a survey of recent academic work on CBR. The descriptions of the programs are intended to convey an intuitive understanding about how the programs work. For more detailed information, the reader should refer to the cited papers. The paradigms are:

Statistically-oriented: Cases are used as data points for statistical generalization. The case-based reasoner computes conditional probabilities that a problem should be treated like particular previous cases. Stanfill's and Waltz's MBRTALK is a good example.

Model-based: Cases are examples explained in terms of a theoretical model of the domain task. Given a new case, the program determines if a past explanation applies. Phyllis Koton's CASEY program illustrates the paradigm.

Planning/design-oriented: Cases are instantiated plans that satisfy goals or avoid goal conflicts; they record a past problem's solution and are used as templates to map the solution on to a new problem. Katia Sycara's PERSUADER and Kris Hammond's CHEF programs are typical.

Exemplar-based: Here, cases are exemplars of frequently ill-defined concepts; they supplement a definition of the concept. Ray Bareiss' PROTOS is a good example.

Adversarial or Precedent-based: Cases are precedents employed to justify arguments how to decide a problem. My own program, HYPO, and Karl Branting's GREBE illustrate this paradigm.

2.1. STATISTICALLY-ORIENTED CBR: MBRTALK

The first paradigm, statistically-oriented CBR is especially good for tackling domains with very weak domain models. It involves employing cases to compute statistical probabilities that a problem should be treated like a previous case. This paradigm is important, but not because of any obvious application to legal problems. Its major drawback is that it only supports numerical case comparisons, not symbolic ones; presumably this would disqualify it from most legal applications. Nevertheless, the paradigm introduces an important relevance measure, the overlap measure, and the method's relative simplicitly facilitates thinking about two complicated ideas: (1) taking context into account in

assessing relevance (Of all the paradigms, this one best supports context-sensitive similarity assessments and generalizing dynamically directly from cases.) (2) employing parallel processing in support of CBR.

The MBRTALK program illustrates this approach to CBR (Stanfill & Waltz 1986; Stanfill 1987). MBRTALK 'pronounces' novel words. An input to the program is a word. The program outputs a 'pronunciation' of the word consisting of phonemes and an indication of where the stress should fall. MBRTALK's database comprises letters and pronunciation for each of 4438 English words. Each case consists of a record of how a particular letter in one of those words is pronounced. A case's 'features' are the letter values and locations of seven predictor fields, comprising the letter itself, the previous three letters and the subsequent three letters. The case's outcome comprises its two goal fields representing the pronunciation of the letter (i.e., a phoneme like the 'a' in bar or a soft 'g' as in germ), and the stress assigned to the letter (Stanfill & Waltz 1986, pp. 1218f).

MBRTALK's aim is to classify a letter in the target word (the 'target letter') by retrieving the 'best matched' cases from its database of pronounced words. The basic method is:

(1) Calculate the dissimilarity between the target word and each case in the database;
(2) Retrieve the n most closely matched cases;
(3) If the pronunciations of the target letter for all n cases are identical, predict a pronunciation for the target letter;
(4) Otherwise, order the possible pronunciations in terms of likelihood (Stanfill & Waltz 1986, pp. 1219–1222).

The most important step is the first one, calculating the dissimilarity between the target and each case in the database. To perform the classification, the program has a highly developed computational definition of 'dissimilarity' involving two basic metrics, the weighted feature metric and the 'value difference' metric (and a correction term). Ignoring the correction term, MBRTALK performs the first step as follows:

For each case r in the database:
For each predictor field f whose value does not match the target word's:
  Compute the weighted feature metric and the value difference metric.
  Associate a penalty with field f equal to the product of the two metrics.
Compute the sum of the penalties for all of the fields and assign this as the case's dissimilarity from the target word (Stanfill and Waltz 1986, pp. 1219–1222).

The weighted feature metric is a variant of a similarity metric commonly encountered in CBR programs known as the overlap metric. Basically, in the overlap metric, one counts the number of features where a target and recorded case differ. Here that metric would count the number of predictor fields for which the target word and case have different values, that is, letters. That metric gives a rough measure of how different the target word is from a case.

Some features are more important than others, of course. The weighted feature metric assigns weights to the different features according to how they constrain the pronunciation of a letter in the target word. For instance, an 'f' in the target letter position con-

strains the corresponding phoneme to either 'f' or '–' (i.e., silent). If the target letter is an 'a' the phoneme is constrained to four possibilities: as 'a's are pronounced in bar, all, or about. A target letter 'g' can be pronounced as in globe ('G'), German ('J'), rough ('f') or although ('–'). On the other hand, if the third letter after the target letter in question is an 'e', it hardly constrains the way the target letter is pronounced at all. Thus, 'f', 'a' or 'g' in the target spot are relatively good predictors while an 'e' in the number three spot is a poor predictor of how to pronounce the target letter. If the goal is pronouncing the 'g' in 'gypsum', for example, words that do not share a 'g' in the target spot are significantly more different than words that do. This may be obvious to us, but it is implemented computationally via the weighted feature metric. MBRTALK computes the weighted feature metric as follows: For all cases that have the same letter in the same location as the target word (for at least one location in the target word), it calculates the frequencies of each possible pronunciation of the target letter as revealed in those cases. Using the square root of the sum of the squares of these frequencies, it computes a weight representing the power of each letter of the target word to constrict possible pronunciations of the target letter (Stanfill & Waltz 1986, p. 1219). The top of Figure 2 shows the dissimilarity ranking of various cases in the database from the target word 'gypsum' as measured by the overlap and weighted feature metrics.

Overlap and weight feature metrics

Pronunciation	Case	Disimilarity
J	gyps	0.0000
g	gale	0.1832
g	gang	0.1832
g	gash	0.1832
g	gall	0.1832
J	gin	0.1832

Value difference metric

Pronunciation	Case	Dissimilarity
J	gyps	0.0000
g	gesu	0.0196
J	gemi	0.0361
J	geni	0.0369
g	gewg	0.0420
J	gin	0.0806

Fig. 2. MBRTALK Example: How to pronounce the 'g' in 'gypsum'. The top shows case dissimilarity rankings as measured by the overlap and weighted feature metrics. At the bottom, the soft 'g' cases become less dissimilar when the value difference metric is taken into account.

Having determined which predictive feature has the greatest weight in determining the outcome (remember that a feature consists of a letter value and its position in the target word), the program next restricts the database to cases sharing letters in those positions. With respect to pronouncing the 'g' in 'gypsum', the weightiest feature is the value of the target letter, itself, and the program restricts the database to cases sharing a 'g' in the target letter position. The program then determines plausible values for the goal by examining the cases in the restricted database. Of all the cases that share a 'g' in the target letter spot, the most likely pronunciations are a soft or hard 'g' and, since there is a case that matches 'gypsum' almost exactly (i.e., 'gyps'), the program treats a soft 'g' sound as the more plausible (Stanfill & Waltz 1986, p. 1221). The next step is to apply the value difference metric.

The value difference metric is a refinement designed to measure the power of a particular letter in a particular location to distinguish among a small number of possible pronunciations. Some following letters have a greater power to distinguish among a small number of possible pronunciations of a target letter than others. If the choices are between a hard and soft 'g', for instance, the letters 'y','e', and 'i' are different from all others. If a 'g' is pronounced as 'J', it usually is followed by 'y', 'e', or 'i'. A hard 'G' usually is followed by all other vowels, although sometimes by 'y', 'e', or 'i'. Where the goal is to determine how to pronounce the 'g' in 'gypsum' and the most plausible pronunciation thus far is a soft 'g', a case that has an 'a' or 'o' in the spot following the 'g' should differ more greatly from the target word than one that has 'e' or 'i' in the following spot. The role of the value difference metric is to capture this difference. It calculates the differential effect of letters in selecting among plausible pronunciations of a target letter. MBRTALK computes the value difference metric by again calculating frequencies of a given pronunciation of the target given the various possible letter values in the other position (Stanfill & Waltz 1986, pp. 1221f). (The program also makes a correction for the effects of combined predictors). The bottom of figure 2 shows the ranking of cases when the dissimilarity measure includes the more refined value difference metric. Note that the soft 'g' cases percolate to the top.

Among CBR programs, MBRTALK is noteworthy for a number of reasons (see Figure 3). Characteristically for statistically-oriented CBR, MBRTALK generalizes directly from cases. There are no rules. Although the information captured by the weighted feature and value difference metrics corresponds to some intuitive rules of pronunciation, the rules themselves are not expressly represented in the program; they are implicit in the data and revealed, in effect, as the program computes the metrics directly from the data. Since the approach generalizes statistically and does not presuppose a strong causal model on which to base assessments of weights or similarity, the approach is ideal for domains that do not have such models.

MBRTALK's elegant computational definition of dissimilarity takes context into account in assigning weights to features. The program computes the weights dynamically each time the program attempts to pronounce a new word. Its computation of dissimilarity is tailored to the new problem's specific facts. Since the weights are not represented

explicitly or implicitly in a set of rules or an *a priori* hierarchy, they are not static. The calculation of weights automatically accounts for any new cases which have been added to the database.

MBRTALK's database of over 4000 cases holds something of a record among CBR systems. On the other hand, cases in this domain of word pronunciation are quite simple, offering none of the richness one expects from a library of legal cases or engineering designs. Interestingly, each new problem is compared to all of the cases, and each case is indexed, in effect, by all of its features.

Calculating all of those frequencies requires an enormous amount of computation. In MBRTALK, the computations are conducted in parallel using a Connection Machine (Stanfill & Waltz 1986, p. 1222), an impressive application of parallel processing. Fortunately, parallel computation is becoming more reasonable in cost, a development that should greatly benefit CBR.

MBRTALK's use of numerical weights to compare cases adequately supports accurate classifications in the hard domain of English pronunciation. In an experiment in which the program tried to pronounce 100 novel words, human judges regarded 47 of MBRTALK's pronunciations to be correct and 21 to be slightly mispronounced (Stanfill & Waltz 1986, p. 1224). In other domains, like law, a failure to support symbolic comparisons among cases would be a serious limitation. In assessing whether numerical comparisons are sufficient, accuracy is only one concern. The other is explanation. In most legal domains, even accurate answers need to be explained and justified. Any numerical approach defeats explanation because too much information is collapsed into the weightings. In MBRTalk, even though one knows what the numbers mean and how they are computed, there is no way to compare cases in terms, say, of the origin of the words or the rule of pronunciation a word exemplifies, information which might help to assess one's confidence in a program's result. The need for such information may not be evident in the domain of word pronunciation, but it certainly is in other domains where symbolic comparisons among cases are essential. The other paradigms provide techniques supporting symbolic reasoning among cases. It is also questionable whether statistical generalizations are powerful enough, even for the domain of pronunciation. An alternative approach to this domain that employs rules and case exceptions is discussed below at Section 4.3.2.

+ Computational definition of dissimilarity is elegant.
+ Program generalizes directly from cases; it has no rules.
+ Feature weighting is highly sensitive to context.
+ Approach does not assume strong model of domain.
+ Program, has a large case database.
− Dissimilarity metrics require lots of frequency computations.
− Program requires parallel processing capability to be efficient.
− What do the numbers mean? Program does not support symbolic case comparisons.
− Statistical generalizations may not be powerful enough for many domains.

Fig. 3. Statistically-oriented CBR: MBRTALK's benefits and costs.

2.2. MODEL-BASED CBR: CASEY

In this paradigm, diagnosed cases are explained in terms of a theoretical domain model. Given a new case, the program determines if a past case's diagnostic explanation applies. Phyllis Koton, a graduate student of Peter Szolovits at MIT, designed the CASEY program as a Ph. D. dissertation (Koton 1988a, b, c).

In the model-based approach, the focus is on symbolic comparison of cases. In selecting among partially matched cases, the program reasons symbolically about the significance of differences between the problem and case in terms of the model. In addition, the cases are used to compile a complex analysis and to increase computational efficiency by eliminating the need to regenerate the complex analysis from scratch. Naturally, the model-based approach assumes that there is a model. In fact, this approach to symbolic reasoning about features and cases requires a strong causal model of the domain task. If there is no such model at hand, this paradigm will not work.

CASEY's task is to analyze descriptions of patients with heart disease and to produce a diagnostic explanation of the patient's heart disease symptoms. The program has three main knowledge sources: a database of 45 diagnosed patient cases, a set of evidence principles, and a detailed causal model of heart diseases embodied in a second program, HEART FAILURE, which also diagnoses heart diseases but does not use a case-based method. Each of the patient cases is represented as a set of features and a solution. The features comprise some 40 signs, symptoms, test results, observations in patient histories, and current therapies. The solution is a diagnosis, a causal explanation for the patient's symptoms, and recommendations for therapy. Each case is indexed by the symptoms and conditions that were explained in the case (Koton 1988b, p. 257).

In overview, CASEY processes a new patient description as follows:

1. CASEY finds the candidate case most similiar to the new patient description and then evaluates the significance of any differences between the retrieved case and the new patient. If the differences can not be reconciled, it rejects the match and considers the next most similar candidate case until one is accepted.
2. If no case is accepted, CASEY invokes the HEART FAILURE program to solve the new patient case from scratch.
   Otherwise, it adapts the solution from the accepted case to fit the new patient. This modified solution is a diagnostic explanation of the new patient's condition.
3. CASEY stores the new patient solution in its case database, appropriately indexed by the symptoms and conditions explained (Koton 1899c, p. 31).

Again, the most important step involves determining similarity, performed here in the first step. In more detail, CASEY:

a. Employs its causal model to obtain a list of all possible conditions explaining any features of the new patient.
b. Retrieves each case having any of the new patient's features or possible conditions.
c. Orders the retrieved cases by the number of new patient conditions and features matched minus case conditions not matched.
d. For the highest ranking case, CASEY attempts to justify the match. That is:
   1. If any features explained in the retrieved case by a condition is not matched in the new patient, the case

can still match if the condition is not ruled out by a new patient feature or if it explains another new patient feature.

2. If the case lacks any patient symptom, the case can still match if the symptom can be explained by some condition in the case or is unrelated to any case condition (Koton 1988c, pp. 36–44).

Even though case D and patient N have different symptoms, D still matches N:

(1) The condition FIXED HIGH OUTLFLOW RESISTANCE explains D's murmur of AS and also explains other symptoms of N.
(2) D's EKG = LV and N's EKG = LVH. It is OK, though, because LV HYPERTROPHY accounts for both.
(3) Patients' heart-rates (96 and 90) are different but in same qualitative region, so difference unimportant.
(4) D's arterial pressure high but N's is not (107 and 99.3). Since feature is not accounted for in D's case and N's pressure is normal, difference doesn't need explanation.
(5) Orthostatic change is absent in D and unknown in N, so assume that it is absent in N.
(6) D has unstable angina while N has unstable and within-hours angina. The condition UNSTABLE ANGINA explains explains both of them.
(7) D has no syncope but N has syncope on exertion. Nevertheless, the condition LIMITED CARDIAC OUTPUT which accounts for D's unstable anginal chest pain also accounts for N's syncope on exertion.
(8) N has aortic calcification but D has none. Nevertheless, since only one condition accounts for N's calcification, AORTIC VALVE DISEASE, that condition can be added to the explanation of N's symptoms without calling into question D's applicability (Koton 1988a, pp. 264–265).

Fig. 4. Justified match example from CASEY.

The key step is justifying a match, which can be illustrated with the example in Figure 4 of the output of CASEY's match procedure. The program is attempting to determine if case D is close enough to new patient N to match. It employs a set of evidence principles to reason about whether the differences matter (Koton 1988c, pp. 43–47).

In justifying the match, the program attempts to reconcile or explain away the differences in D's and N's symptoms or conditions. In (1), patient D has an extra symptom not in N, but it is explained in D by a condition that also explains other of N's symptoms. In (2), (6), and (7) different symptom magnitudes or test values in the patients are reconciled because patient D's condition is consistent with both. In (3), different test values are treated as qualitatively similar. In (4), D has an abnormal feature, but since N is normal in that regard and since the feature was never explained in D, the program concludes it need not be matched. In (5), the program assumes that an unknown in patient N, which was absent from D, is also absent from N. In (8), the program decides that it can add a condition to explain patient N's symptom. Since only one condition could account for the symptom, adding it would not affect the applicability of the explanation of D's condition (Koton 1988a, pp. 264–265).

As the example illustrates, CASEY reasons symbolically about the differences and whether the differences matter or can be reconciled or ignored. This is in marked contrast to the approach of the statisically-oriented CBR paradigm.

CASEY's causal model of the domain and its evidence principles enable it to reason symbolically in this way. The causal model, a causal inference network, links each condition node to (1) findings (i.e., features) for which the condition accounts and (2) other conditions caused by the condition. The links have associated probabilities, as well (although CASEY does not make use of them). CASEY and HEART FAILURE both

make use of the same causal model, but in different ways. CASEY employs the model to retrieve cases and in the justified match process. It also uses the model in adapting a case's explanation to a new patient (Koton 1988c, pp. 48–52).

HEART FAILURE, on the other hand, uses the model to solve a problem from scratch (Koton 1988c, pp. 23–30). Given a list of findings for a new patient, it propagates the list backward to conditions that cause them. HEART FAILURE then finds the most probable set of conditions that 'cover' the findings using causal, probabilistic and heuristic reasoning. HEART FAILURE can even diagnose multiple diseases in a single patient. The program's diagnostic facility comes at the expense of very high computational complexity, however (Koton 1988c, p. 30).

CASEY realizes the promise of Case-Based Reasoning to speed up complex analyses by employing past solutions to pattern a solution for the new problem. In CASEY, a case compiles HEART FAILURE's complex analysis of a patient. The case records the patient's diagnostic explanation and the circumstances under which that explanation can be reused, namely the features and conditions that were explained. The explained features and conditions index the case and capture the circumstances under which that case is useful in the future. CASEY does not take nearly as much computational effort to produce a diagnosis as HEART FAILURE and its diagnoses are of comparable accuracy (subject to some qualification described below). CASEY always examined fewer states than HEART FAILURE by one to three orders of magnitude (Koton 1988b, p. 260). CASEY integrates CBR and another reasoning method, model-based reasoning. Once the model-based reasoner (i.e, HEART FAILURE) has generated the corpus of cases, CASEY can apply this compiled knowledge in a way that demonstrably increases efficiency.

CASEY'S ability to reason about differences allows it to assess partial matches symbolically. The circumstances under which a case can be applied are generalized by the justified match procedure and the causal model. The justified match extends the range of a case's applicability in light of the causal model to situations not yet encountered. The case is useful not only when the new patient evidences exactly the same features and conditions but also in the wider range of patients evidencing features and conditions different from the case but where the match can be justified.

In summary (see Figure 5), CASEY is important because it symbolically compares cases. It deals with partial matches by reasoning about the effect of the differences. Its cases compile the complex analyses produced by an alternative reasoning system and can produce comparably accurate analyses while demonstrably increasing efficency. The program demonstrates how to build causal explanations of new problems from past explained cases. CASEY's causal model informs its assessments about the significance of similarities and differences, in effect, substituting for the lack of feature weights. Fortunately, the domain has a strong causal model, so the reliance on a strong model is not a problem.

Perhaps the main limitation of the CASEY program involves the question of whether CASEY's solutions are always as accurate as those produced by HEART FAILURE. Specifically, HEART FAILURE picks the more likely of two or more reasonable explanations of a patient's problem. CASEY, however, only sees the selected explanation. Even though other new patient facts make the rejected explanation more likely, CASEY

sticks with the best previous solution (Koton 1988c, pp. 91f). Possibly, this problem may be dealt with by augmenting CASEY's representation of a case to include alternative explanations. In other domains, it may not pose a significant objection. There may also be circumstances where one is willing to sacrifice accuracy, for instance, to get quick reasonable answers in an emergency.

+ CASEY symbolically compares cases; it deals with partial matches by reasoning about the differences' effects.
+ Cases compile complex analyses. The program demonstrably increases efficiency.
+ A strong causal model informs the weighting of features.
+ CASEY demonstrates how to build causal explanations from explained cases.
− Reasoning about features and cases requires a strong causal model.
− CASEY does not store or apply alternative reasonable explanations.

Fig. 5. Model-based CBR: CASEY's benefits and costs.

2.3. PLANNING/DESIGN-ORIENTED CBR

In the planning/design-oriented paradigm, cases are instantiated. They record a past problem's solution and are used as templates to map the solution on to a new problem. Here the focus is on indexing cases according to the planning goals satisfied in the case or the conflicts among planning goals that the case resolves or avoids. Retrieval is a matter of finding the prior case that shares the most goals with the problem and avoids anticipated conflicts. The retrieved plan must be adapted to solve the new case, a task that may require solving new problems. Also, the planning successes and failures should be recorded and indexed appropriately so that in the future the planner can reuse successful plans and avoid the failures. Two systems are characteristic of this paradigm, Sycara's PERSUADER and Hammond's CHEF programs.

2.3.1. *Persuader*

Katia Sycara, a Ph. D. student of Janet Kolodner at Georgia Tech, designed a program, PERSUADER, that adapts past solutions to meet the constraints of the current problem (Sycara, 1987, 1989a, b).[2]

[2] PERSUADER extends work done on an earlier program, MEDIATOR, designed by Robert Simpson as his dissertation project (Simpson 1985; Kolodner et al. 1985). (As a DARPA project officer, he later was instrumental in securing funding and recognition for CBR research.) MEDIATOR's task is also to settle disputes by applying past solutions. The disputes focus on international political, military and economic issues as seen from a nontechnical, 'common sense' viewpoint. MEDIATOR has a rich set of knowledge sources including a case database of past solved disputes, methods to infer the disputants' goals, and libraries of mediation plans and of recovery procedures. Each case is represented by the problem that gave rise to the dispute (i.e., the parties' conflicting goals and conditions), the plan that was generated to resolve the dispute, feedback on execution of the plan including any failures, the reason for the failures and the next steps to recover from the failure (Simpson 1985, pp.18–45). MEDIATOR is important because it uses cases in multiple ways to solve problems: to identify wrong goal inferences, choose mediation plans, instantiate an abstract mediation plan and recover from failed plans. Perhaps the main criticism of MEDIATOR is that it has no very robust method for adapting past solutions to current problems.

PERSUADER's domain is collective bargaining; its task is to resolve disputes between union and management. The inputs are descriptions of a dispute or rejected compromise including the disputants' conflicting goals. The outputs are a proposed or revised compromise resolution and argument that the compromise should be accepted based on a past solved union dispute. The knowledge sources include a database of previous negotiations and settlements and methods for modifying proposals in light of financial, political and bargaining unit information. PERSUADER's case representation of a past negotiation comprises the disputants' goals and beliefs, the labor dispute context, arguments, proposals and counter proposals, rejections and the final agreement.

The cases are indexed in hierarchical discrimination nets whose nodes are either generalized episodes or individual cases. Each generalized episode has a norm, a collection of features shared by all of the cases indexed under that episode. Individual cases are indexed under the episode by their particular features that differentiate them from other cases under the norm. Typically, the system designer and case enterer manually provide the initial generalized episodes, although some work has been done on having a program maintain and revise them.[3] Retrieving a case relevant to a problem is a matter of determining the appropriate generalized episode for the problem and then traversing the discrimination net to find the case that shares the most differentiating features with the problem.

In PERSUADER, there are generalized episodes for types of disputes (e.g., STEEL-FRAMES-INDUSTRY disputes) and types of impasse issue resolutions. For each generalized episode under which a case is indexed, it is subindexed by the unique differences it has from the generalized episode and from the other cases indexed under that episode (e.g., characteristics of the disputants such as geographical location or subject matter of the impasse) (Sycara 1989b, pp. 246f). See the top of Figure 6 for a sample discrimination net for contract disputes in the steel frames industry. Four disputes are shown all involving steel frames manufacturers and unions. The disputes differ, however, in terms of geographic location and types of provisions included in the contract. In traversing the discrimination nets, PERSUADER prefers cases more similar to the problem as measured by various orderings over the features. For instance, the bottom of Figure 6 also shows a hierarchical ordering of industries from which the program would conclude that a settlement involving a company in the air frames industry is more similar to a problem involving a steel frames manufacturer than an air transportation settlement.

Given a problem dispute and a statement of the disputants' conflicting goals, the program:

1. Proposes a 'ballpark' settlement in the following way:
   (a) It accesses past cases where similar disputes were resolved, selects a dispute that has the most similar features (such as kind of industry or geographical location) and adopts the solution from that case as the basis of a proposal.

[3] Following the custom in the Schank school, such an organized case database in referred to as 'memory' and the retrieval of a relevant case as 'reminding'. Janet Kolodner invented this memory organization in the CYRUS program (Kolodner, 1983a, b). CASEY also employed a variation of this kind of discrimination net (Koton 1988c, p. 22).

(b) The program employs general modification rules to modify the proposal. The rules adjust the past solution to account for the new company's financial situation, the structure of the bargaining unit, political considerations and the economic context (Sycara 1989b, p. 248).
(c) If no case is available, it generates a ball park solution plan from scratch.

2. Evaluates the settlement plan by subjecting it to its internal critic or by proposing the plan and seeking agreement and feedback from the disputants (as represented by the user). If the disputants do not agree to the plan, the program generates arguments based on past cases where the union and management of the case agreed to such a plan. Offering these arguments, the program seeks the disputant's agreement again.
3. If the disputants agree to the proposed settlement, the program succeeds and updates memory with the success (Sycara 1989b, p. 247).
4. If the proposed settlement plan is not acceptable and there is still time, the program modifies the plan based on a retrieved case or from scratch or, if no modification would be effective, generates a new plan by returning to the first step. If there is no time, the program deems the plan a failure and updates memory.

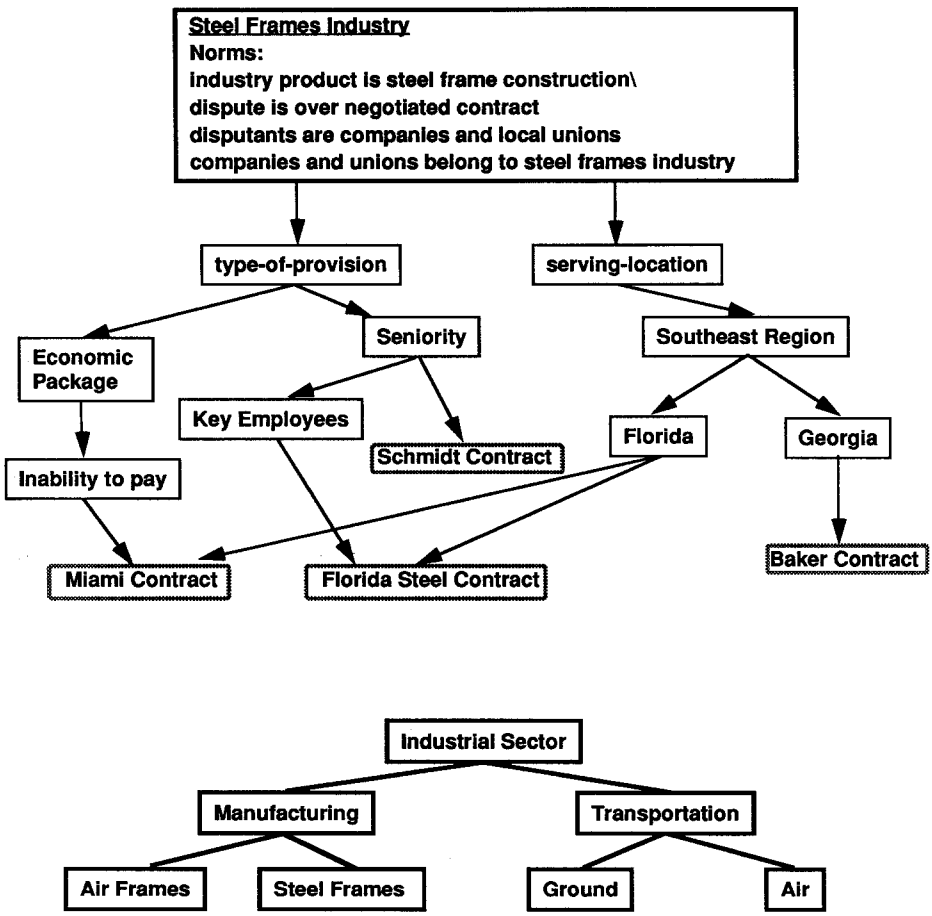

Fig. 6. Top: Hierarchical discrimination net in PERSUADER. Bottom: Hierarchical feature ordering.

For instance, here is an example of how PERSUADER deals with a typical dispute between Thompson Co., a Georgia air frames manufacturer, and its union. The union

wants a 15% increase in wages, 7% increase in pensions, no subcontracting and strict seniority. The company wants no increase in wages or pensions, unlimited subcontracting and no seniority. Searching its memory for AIR-FRAMES-INDUSTRY cases, the program finds none, but switches to a more general indexing category, STEEL-FRAMES-INDUSTRY, finds four cases and selects one, the Baker case, since it involves not only a similar industry but the same geographic area. PERSUADER adopts its provisions, a 10% increase in wages and subcontracting for limited time periods. PERSUADER then determines if any of the input constraints have not yet been dealt with and finds that the dispute over seniority provisions has not been addressed. Since there is no seniority provision in the Baker contract, the program does an intersection search on the categories, STEEL-FRAME-INDUSTRY and SENIORITY, and retrieves one case, the Schmidt case. The Schmidt contract gives moderate seniority, so the program combines its provisions with those of the Baker contract to form the initial Thompson contract.[4]

PERSUADER makes two kinds of adaptations. The program makes general adjustments to the proposed deal to account for different prevailing circumstances in the current problem; its domain specific heuristic rules correct for a company's position in the industry and for differentials in wages, geographic area and plants. The program can also modify a proposal to address the disputants' specific objections and in response to its own internal critic which anticipates difficulties using a model of the disputants' likely preferences (Sycara 1989b, p. 248). It has six domain specific heuristics for modifying settlement plans to try to obtain agreement: change the value of an issue (e.g., the monetary value of an issue), pass a settlement cost on to a third party, postpone a decision on some issues, make an argument invoking authority to persuade agreement, extend the current agreement or submit to binding arbitration (Sycara 1987, p. 144). Associated with each heuristic are preconditions which must be present for the heuristic to apply.

For instance, spurred by the internal critic's review of the settlement, the program first undertakes some general adjustments to the initial Thompson contract and then makes more specific case-based adjustments. The program adjusts the Thompson contract based on industry differentials and area differentials, setting a 12% increase in wages, 5% increase in pensions, subcontracting for limited time periods, and moderate seniority. Since Thompson Inc. had losses of 3% in the past three years, the program searches memory under the index categories, PROFIT-LOSS and AIR-IND-SLUMP. Not finding any cases, the program applies a heuristic rule, 'Reduce increases by half percentage of losses', setting the increases at 10.5% for wages and 3.5% for pensions. Since the internal critique indicates that Thompson Inc. cannot afford the economic package, the program searches memory under the index categories ECON-PKGE and INABILITY-TO-PAY, finds three cases, and selects the second one, the Miami contract since it involves a similar product, geographic area, and economic package issue. The solution in the retrieved case was to 'pass the extra cost to consumer'. The program checks that plan's

[4] This example is drawn from one presented by Katia Sycara in two tutorials on Case-Based Reasoning at AAAI–91 in Anaheim, California and IJCAI–91 in Sydney, Australia. It is similar to the example in (Sycara, 1989b, pp. 248f).

preconditions. Since Thompson's market is insensitive to product price changes for airplane frames, it adopts the plan and outputs the proposed resolution.

The modifications are not over, however. From the user's feedback, the company objects to the proposed seniority language saying that it limits its flexibility with respect to key employees. PERSUADER searches memory under index categories FAILURE, SENIORITY, and KEY-EMPLOYEES and finds two impasse-resolving cases. It selects one, the Florida Steel Contract, because the case involves the same industry, geographic area, and job classification and adopts that case's repair plan to 'except key employees' after finding that the plan has no preconditions.

Interestingly, PERSUADER has an Improvement Criterion to help ensure that a contemplated repair is likely to converge on an acceptable deal. At least some of its adaptations are guided by the overall improvement criterion, according to which the repair should increase the satisfaction of the rejecting agent by a greater degree than it might decrease the satisfaction of the agent who previously agreed to the solution (Sycara 1989b, p. 249). The program employs various utility functions to measure a disputant's satisfaction.

PERSUADER's strengths (see Figure 7 for a summary of benefits and costs) are that it draws on past cases to design an acceptable compromise for a current problem and can assemble pieces of a plan from various cases. It can also evaluate the appropriateness of a plan or provision and adapt it to satisfy new constraints. PERSUADER also has fallback methods for generating provisions from scratch if there are no cases. This work points toward an adversarial use of cases; PERSUADER's arguments in favour of a proposal cite past disputes in which the parties agreed to similar conditions. On the negative side, PERSUADER tends to oversimplify the design problems. It probably is a gross oversimplification for the program to employ mathematical utility functions to measure the disputants' satisfaction with proposed modifications. Realistically, how can such data be obtained? One also wonders how realistic the representations of the negotiations are and whether they accurately reflect subtle differences between negotiation contexts. A lawyer looking at the negotiations would regard them as grossly oversimplified. Also, how good is the program at maintaining consistency among disparate pieces of a plan? In any negotiated settlement, the terms are highly interdependent because the settlement is a compromise. Parties have agreed to sacrifice some goals because others have been achieved. Realistically, how can pieces of such a settlement be imported to new settings without a sophisticated model of the compromise as a whole?

\+ PERSUADER draws on past settlements to design and justify acceptable compromises.
\+ Repairs converge to guarantee satisfaction of both sides.
\+ Program has fallback methods for generating contracts from scratch without CBR.
\+ Planner puts pieces of cases together to solve problem.
\− Are negotiations realistically complex?
\− How realistically does the program accommodate subtly different negotiation contexts?
\− How good is it at correcting cases for global changes?
\− How good is it at maintaining consistency among complex disparate pieces of a compromise solution?

Fig. 7. Planning/design-oriented CBR: PERSUADER's benefits and costs.

PERSUADERS's memory organization as descrimination net is very suggestive. Intuitively, generalizing concepts in terms of shared features and indexing individual cases by their differences from each other seems like a good idea. Potentially, such a memory could be efficient and self-organizing. New cases are specially indexed only to the extent they are different from existing cases. There is the potential for reorganizing the index automatically. As new cases are added, features may migrate to or from the norm. On the other hand, questions remain whether such a memory organization and retrieval mechanism would bog down as the number of concepts, cases in memory, and the complexity of their representations increase. The hierarchical feature orderings reflect a static ordering of the importance of features and do not support context sensitive relevance assessments. Thus, the discrimination net does not realistically reflect the relative significance of different features that cases may share with the problem. It seems plausible that an unrelated industry contract provision from out-of-state still may be very relevant in a particular context. Some kind of justified match process, as in CASEY, or other method for reasoning about the significance of differences in a specific context is required.

2.3.2. *Chef*

The second example of a planning/design-oriented CBR program is CHEF. Kris Hammond designed CHEF as a Ph.D. dissertation project at Yale for his advisor, Roger Schank (Hammond, 1986a, b, 1987, 1989). CHEF's domain is cooking; its task is to plan recipes. The inputs to the program are a set of goals or constraints for the prepared dish. For example, the user may specify that the recipe should be for a stir fry dish with chicken and snow peas or a souffle with strawberries. The program's output is a recipe satisfying the constraints. The program tests its proposed recipes by simulating their outcomes; the simulator determines whether the recipe is a gastronomical failure and generates an explanation of why it failed. The main knowledge sources include a case database of some number of past recipes, a set of rules for the simulator, rules for predicting failures in advance, a set of strategies for repairing failed plans and a set of rules for adapting plans. Each case is represented as the set of steps in the recipe, the goals satisfied by the recipe and the goal conflicts resolved. The system's indexing vocabulary consists of goals that the recipes satisfy and goal conflicts resolved in the recipes (Hammond 1989, pp. 32–62; 1986a).

CHEF's planning process works as follows:

1. CHEF analyzes the input goals to anticipate goal conflicts. If a conflict is anticipated, an additional goal to avoid those conflicts is added to the constraints.
2. The program retrieves the past plan that satisfies as many goals as possible. If a literal match for a goal cannot be found, the program chooses among partial matches by referring to an a priori goal hierarchy.
3. CHEF modifies the retrieved plan to satisfy goals not yet achieved and tests the recipe in the simulator.
4. Successful plans and failures are stored. If the plan fails, using the trace of the simulation, the program builds a causal explanation of the failure and uses it to:
   a. Apply repair strategies.
   b. Identify the features in the inputs that caused the failure.
5. The program indexes the repaired plan as a new case under the goals satisfied and the goal conflicts resolved by the recipe (Hammond 1986a, p. 268). It also updates the knowledge that allows it to anticipate failures.

An example illustrates CHEF's use of a past plan to solve a new problem. Assume that the input goals were to create a stir-fry dish that included chicken and snow peas. First, CHEF analyses the constraints to anticipate failures (using, in effect, a rule-based analysis implemented with a network). Here, it anticipates that the recipe will fail because the chicken adds liquid to the stir-fry mixture resulting in soggy vegetables. The program adds to the constraints a goal to avoid this failure by resolving the conflicting goals. CHEF searches the index for the plan that satisfies the greatest number of the constraints. The best matched plan turns out to be the 'Failed Beef and Broccoli' plan where the vegetable was too soggy. The Beef and Broccoli plan was repaired using a 'Split and Reform' strategy in which the stir-fry was conducted in two steps so that the meat and vegetables were fried separately. Previously, CHEF had stored the repaired plan under the fact that the plan avoided the problem of the liquid in the stir fry causing the vegetables to get soggy. CHEF modifies the retrieved plan to apply to the new problem. It substitutes chicken for beef, snow peas for broccoli and proceeds (Hammond 1986a, pp. 268–271; 1989, pp. 32–48).

CHEF has the distinction of learning from failures. The program is said to learn if it can analyze the failure and index it appropriately so that the failure can be avoided in the future. This kind of learning involves a problem of credit assignment. It is instructive to examine why CHEF can learn in the context of a specific example. Assume that the simulator shows a plan failure. The goal was to make a souffle that tastes like berries, but the batter has fallen. The simulator is rule-based and returns a trace of the inference steps that led to a failure (Hammond 1986b, p. 557; 1989, pp. 32–48):

Goal: Make souffle taste like berries ==>
Step: Pulp strawberries ==>
Side effect: Adds extra liquid ==>
Failure of Condition: Imbalance between whipped stuff and liquid ==>
Failure: Batter falls

The next step on the way to learning from the failure is to explain the failure. Basically, CHEF steps backward through the simulation trace to the step of pulping the strawberries. That was the step that led to the undesirable side effect that caused the condition failure. CHEF has a number of strategies to repair an imbalance failure in a plan. It may avoid the side effect (use strawberry preserves), recover (drain pulped strawberries), take a concurrent step (add flour) or adjust the balance (add more egg-white). It selects a strategy and repairs the plan (Hammond 1986b, p. 558; 1989, pp. 39–43).

Having corrected the failure, next CHEF must learn from it. That means, CHEF must index the repair appropriately so that it can be recalled on relevant future occasions. CHEF indexes the repaired plan under the goals satisfied and failures avoided. It indexes the saved souffle under 'Avoids failure that chopping fruit causes liquid imbalance and batter falls.' CHEF also can abstract from the failure two tests to anticipate similar failures in the future. The two tests it generates to apply to inputs are, 'Is the item a fruit?' and 'Is the item a liquid spice?' Since chopping strawberries caused the liquid, by climbing an is-a hierarchy from strawberries to fruit, it hypothesizes that chopping fruit causes

liquid. (Query what happens with bananas.) The program can also ask what else releases liquid in light of what it knows about the properties of ingredients and predict that liquid spices also cause liquid and threaten to bring the batter down (Hammond 1986b, p. 559; 1989, p. 36).

CHEF's indexing plans under the goals satisfied and the goal conflicts avoided seems to be a natural scheme for indexing collections of designs. It is less clear that the scheme would work for highly complicated plans with many subgoals and lots of interdependencies in a design's components. More recent case-based planning systems also index past plans by features of the planning situation's initial state as well as by features of the context in which planning decisions were made (see Section 4.6).

CHEF can learn because it is able to analyze the failure and assign credit or blame to the step that caused the failure. In general, credit assignment can be computationally very expensive. Here the assignment problem is manageable because the explanation of the failure is contained in the simulator's inference chain output. By definition, the plan can only fail in ways the simulator knows about. Since the explanation trace is short and simple and the failure has only one cause, the program knows exactly how far up the chain it should go in identifying the cause. It is not difficult to imagine more complex inference chains where identifying the exact cause of a failure would be far more difficult computationally.

+ Planner 'learns' from failed plans.
+ Plans (recipes) are indexed under goals satisfied and goal conflicts resolved.
+ Program dynamically indexes cases as it solves problems.
− Approach requires a strong causal model for explaining failures and credit assignment.
− Program assumes each case has only one, simple explanation.
− Program assumes cases are fully worked out examples. It maps solutions onto problem.

Fig. 8. Planning/design-oriented CBR: CHEF's benefits and costs.

CHEF's accomplishments in problem solving, explaining failures, credit assignment and learning (see Figure 8) necessitate the existence of a strong causal model. The simulator's rules, for instance, are part of that model. The program assumes that each case is a fully worked out example that has only one, simple explanation and that failures have unique, easily identifiable causes. In dealing with complex design domains, these assumptions would have to be abandoned or modified, presenting interesting research issues.

2.4. EXEMPLAR-BASED CBR: PROTOS

In the fourth CBR paradigm, cases are exemplars of concepts. Problem solving in this paradigm is cast as classification. The cases help to classify new problems as instances of concepts. Often, the concepts cannot be satisfactorily defined logically with a set of necessary and sufficient conditions. An expert can, however, explain why a case exemplifies a concept. These explanations may be recorded with the case and supplement the definition of the concept. Not only do they help to illustrate the concept; they play an

integral role in representing, applying and explaining the concept. Ray Bareiss' PROTOS exemplifies this paradigm (Bareiss et al. 1988, Bareiss, 1989, Bareiss et al. 1989).

Ray Bareiss, a former Ph.D. student of Bruce Porter at the University of Texas, designed PROTOS as a doctoral dissertation project. PROTOS' domain is clinical audiology, the study of diseases of the ear. Its task is to classify patients with hearing problems. The knowledge sources include 86 exemplars of 16 diagnostic categories of clinical audiology. The categories and exemplars are organized in a category structure (Bareiss 1989, pp. 11–12).

PROTOS' category structure is designed to deal with the fact that, in reality, many concepts cannot be defined with a set of necessary and sufficient conditions. Instead, the program assumes that one knows whether such concepts apply through comparison with exemplars (Bareiss et al. 1988, p. 552). In PROTOS' category structure, concepts are indexed by features, either directly or indirectly, via case exemplars (see Figure 9). A feature is linked to the concept if all instances of the concept have the feature. A feature is linked to an exemplar if the feature is relevant to why the exemplar is an instance of the concept but not all of the instances of the concept have the feature. These links are called remindings and can be assigned weights representing how strong the link is. In addition, cross links among the exemplars show how they differ, that is, show features significant to the classification that one exemplar has but the other does not.

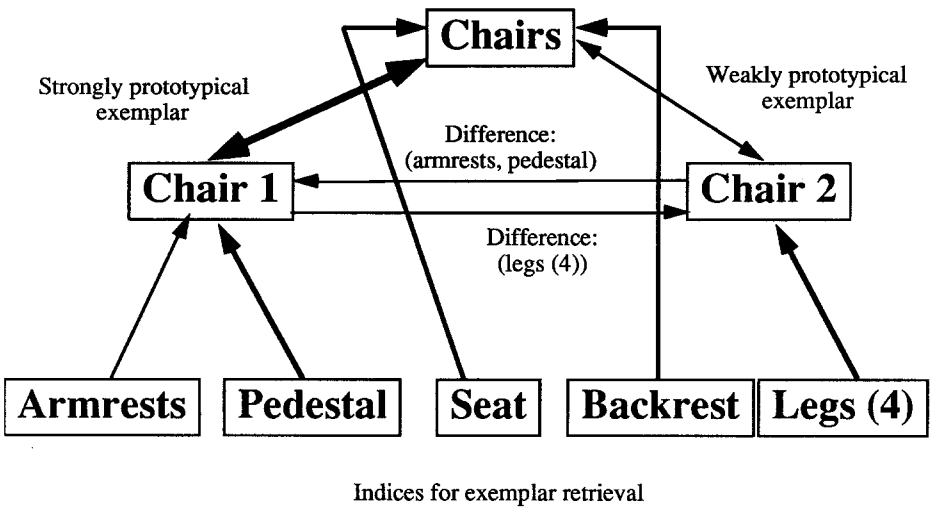

Fig. 9. PROTOS category structure.

This arrangement finesses the problem of defining the concept in terms of necessary and sufficient conditions. It would, for example, be impossible to define the concept 'Chair', but one could provide exemplars of chairs. In Figure 9, all chairs have seats and backrests. Chair 2 also has four legs and Chair 1 also has armrests and a pedestal. As indicated by the lack of a direct link from feature to concept, not all chairs have legs,

armrests or pedestals. The difference links between Chairs 1 and 2 indicate that the former does not have legs and the latter has no armrests or pedestal.

The various links among features, exemplars and categories in PROTOS' category structure can be assigned weights reflecting the prototypical quality of the exemplars and the features' importance. The weights are assigned initially by the teacher and revised by the program automatically and in consultation with the teacher. PROTOS is unique among the CBR paradigm programs in that it interacts with a human expert teacher whose role is to critique PROTOS' exemplar-based classifications and provide feedback which PROTOS uses to revise and update its category structure.

More specifically, the links from categories to exemplars are weighted according to how prototypical the expert regards the exemplar and revised by how frequently the exemplar has been used successfully in the past to classify a new problem. Chair 1 has been deemed to be more prototypical than Chair 2 in Figure 9. The links from features to exemplars or from features to concepts can be assigned weights to represent, in light of the expert's explanation, how important the features are in explaining why the exemplar is an instance of the concept. If a teacher's explanation indicates that the feature is central to the explanation, a higher weight is accorded to the link (Bareiss et al. 1988, p. 554). There are predefined relations and qualifiers for expressing degrees of centrality to an explanation. The relations include 'consistent with', 'causes', 'co-occurs with', and 'has part'. Qualifiers include 'usually', 'sometimes', and 'occassionally'. Heuristics for estimating weights of links are associated with these relations and qualifiers (Bareiss et al. 1989, p. 268). A feature which 'causes' an exemplar to be classified as an instance of a concept will be assigned a greater linkage weight than one which merely is 'consistent with' the classification. Later, PROTOS may revise the weights in light of the expert's critique of its decisions or to reflect that some exemplars have proven more useful in making classifications than others.

In PROTOS, the explained exemplars help to classify new problems in the following way:

1. The features of the new case trigger remindings (i.e., connections via category structure links) to all of the concept classifications to which any of the features is linked, either directly or via an exemplar.
2. The program determines the best preliminary classification using heuristics to sum up the weights associated with the direct and indirect linkages.
3. PROTOS selects the most prototypical exemplar of the classification and constructs a match between it and the problem. PROTOS examines the problem features that do not match the exemplar to determine if the match can be improved. Given the unmatched problem features, the program follows difference links to neighboring exemplars to test if any of them would better match the problem. If so, PROTOS substitutes the better matched exemplar.
4. PROTOS presents the match between the selected exemplar and the problem to a human expert tutor as an explanation justifying the problem's classification and asks the tutor to accept or reject it.
   If the expert does not accept the explanation, the program automatically queries the tutor for a corrected explanation or modification of its existing category structure to correct the error. For instance, if according to the expert, PROTOS misclassified the new problem, the program may add difference links between it and the exemplars to which it was mistakenly matched. PROTOS returns to step 2.
5. Once the new problem is correctly classified, if the new problem strongly matched the prototypical exemplar, PROTOS merges it with the exemplar. Otherwise it adds the problem as new exemplar and decreases the measure of prototypicality of the old exemplar, accordingly (Bareiss et al. 1988, p. 551; Bareiss 1989, pp. 13–24, 36–54).

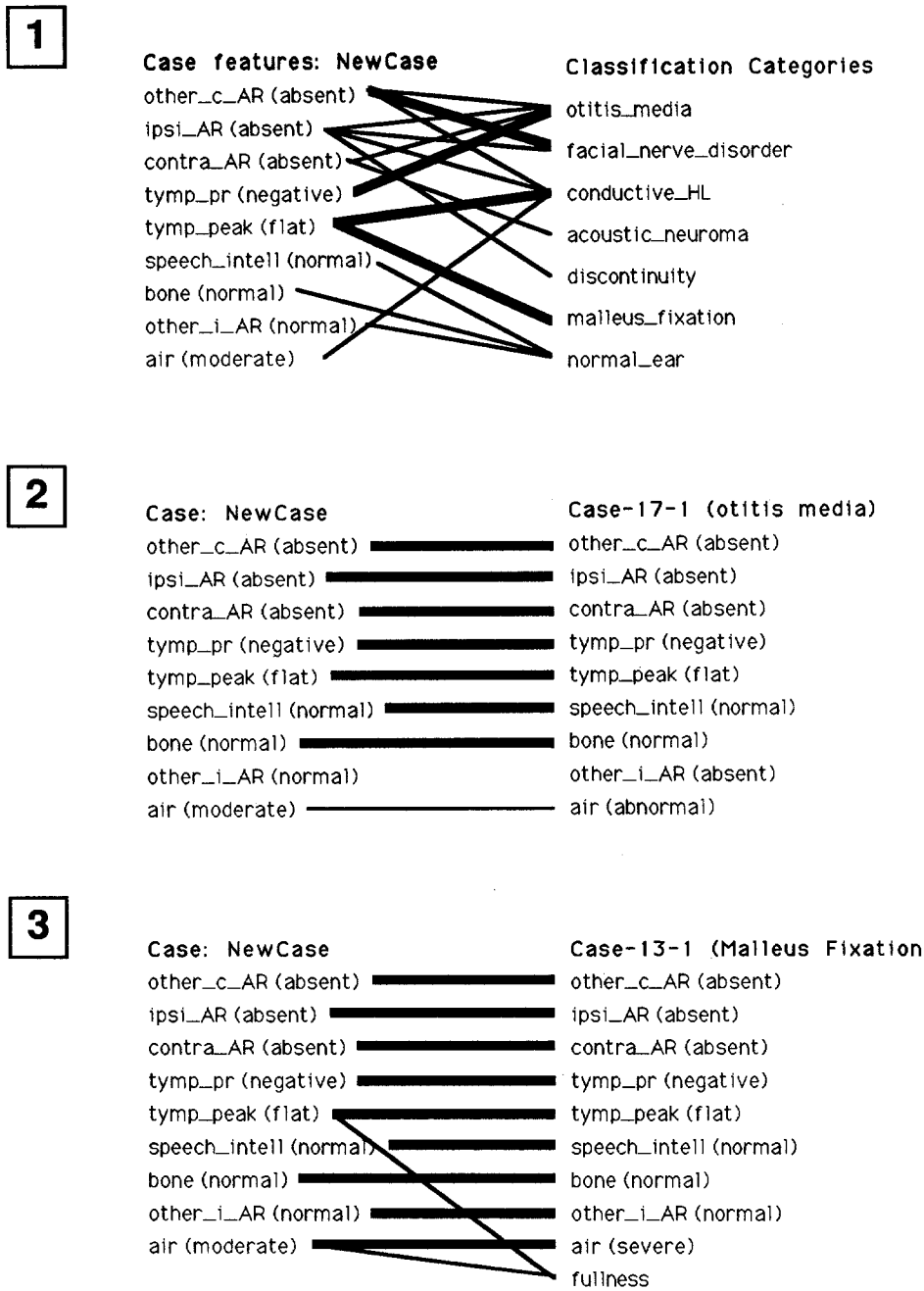

Fig. 10. PROTOS example.

The example in Figure 10 shows how PROTOS classifies a new patient's hearing problems (New case). In part 1, the New Case's features have been matched to all possible classifications. PROTOS applies its heuritics for picking the best matched classification.

It sums multiple matches to a category, passes the match totals to any specializations of a category (drawn from a taxonomy of clinical audiological conditions) and checks for any common specializations for multiple categories. Here, it sums up the four matches to 'otitis-media' and to 'conductive-HL'. There also are strong matches to the categories 'facial-nerve-disorder' and 'malleus-fixation'. Comparing the relative weights of the various matches, PROTOS chooses the most strongly matched category, otitis-media (Bareiss et al. 1988, p. 556).

The New Case is not finally classified as an instance of any category, however, until it is shown to match a prototypical exemplar of the diagnostic category. First, PROTOS retrieves Case-17-1, the most prototypical exemplar of otitis-media, matches up the features of the new case, and presents the proposed match to the human expert (see Figure 10, part 2). Let's assume that the expert rejects the classification, however, and insists on considering alternatives. As a result of the rejection, PROTOS reduces the link weights that lead it to select the category otitis-media. This is PROTOS' method of dealing with credit assignment. With the approval of the expert, it increases the weight of the unmatched feature 'other-i-AR (absent)' between the New Case and Case-17-1 so that this mistake will not be repeated. Returning to step 2, it tries the next best classification, malleus-fixation, and matches the New Case to Case-13-1, the most prototypical example of that category, as shown in Figure 10 (part 3), a classification with which the expert agrees (Bareiss et al. 1988, p. 556).

As the statement of the process indicates, PROTOS combines classification and knowledge acquisition. In theory, as it is run on more and more training cases with a human expert, it will have acquired enough exemplars, links, and weights to classify new problems correctly. As indicated in steps 4 and 5, in the course of attempting to classify new problems, it supports automatically acquiring new knowledge about the domain in the form of new exemplars, new linkages in the category structure between features, exemplars, and concepts, and revised weights.

As a first kind of automated knowledge acquisition, if a new case is dissimilar from an exemplar, PROTOS retains it as a new exemplar. In the above example, the new case is so similar to Case-13-1, that PROTOS merges the two rather than record the new case separately (Bareiss et al. 1988, p. 556).

Second, PROTOS may ask the teacher to relate features between a new case and an exemplar that are not matched. For instance, when it first tried the match to Case-13-1, the new case's air (moderate) and Case-13-1's fullness feature were not matched (see Figure 10, part 3). On querying the expert, the teacher provides the explanations that 'air (moderate) has generalization air (abnormal) which has specialization air (severe)' and that 'tymp-peak (flat) and air (moderate) are usually sufficient for fullness.' PROTOS updates the match, accordingly, and adds these generalizations to its knowledge about the domain. PROTOS incorporates the explanations into its category structure by adding or deleting links and nodes and assigning appropriate weights. In future classifications, the program can invoke these generalizations in matching a problem and exemplars (Bareiss et al. 1988, p. 556).

Third, when PROTOS mismatches a new case to an exemplar, it installs difference

links between the incorrect and the correct exemplars. Here, for instance, it originally erroneously matched the new case to the otitis-media Case-17-1. With the expert's approval, PROTOS draws a link between Cases 17-1 and 13-1 noting the differences. This may affect PROTOS in a future attempt to improve a match in step 3. Suppose that PROTOS happens to be considering a match between a new case and Case-17-1. If it discovers problem features that do not match and those features are the same as indicated in the difference link to Case-13-1, the program can follow the difference link to the other exemplar to test if it would better match the problem (Bareiss et al. 1988, p. 558).

Finally, PROTOS may ask the teacher to explain the relation between a case feature and a classification. For example, the teacher may explain that 's-neural (profound, 2k) sometimes [is] consistent with Cochlear which has specialization Cochlear Age which has specialization Cochlear Age Noise.' PROTOS adds this links to its category structure, suitably weighted to reflect the weakness of the evidential connection. See (Bareiss et al. 1989, pp. 268, 270).

PROTOS deals with a domain theory of intermediate strength; categories in this domain, as in so many others, cannot be defined in terms of necessary and sufficient conditions. PROTOS shows how case exemplars can supplement the meanings of categories in such domains. Indeed, in so far as it supports automated knowledge acquisition of exemplars and explanations, it supports acquiring and representing a domain theory (see Figure 11 for a summary of PROTOS's benefits and costs).

+ Exemplars supplement classifiers' meanings.
+ PROTOS classifies new problems by symbolically comparing them to exemplars in light of its category structure.
+ It assumes an intermediately strong domain theory and supports acquiring new knowledge about the theory from a human teacher.
+ PROTOS represents one aspect of paradigmatic cases, prototypicalness.
+ Program has been empirically evaluated and copies are available to researchers.
− Weights and prototypicalness are represented numerically.
− Approach assumes that exemplars have no alternative explanations.

Fig. 11. Exemplar-based CBR: PROTOS's benefits and costs.

PROTOS supports symbolically comparing problems and cases in light of its accumulated explanatory theory. The cases that get compared symbolically are of a special sort: prototypical cases which I will treat as one sense in which cases may be paradigmatic.

Reasoning with paradigm cases is important in a variety of domains, and there are a variety of senses in which a case may be a paradigm. PROTOS' representation of a prototypical case consists of a list of feature values which an expert asserts is prototypical of a certain classification, an assertion whose weight is revised up or down with the utility of the case in classifications. In other domains (like law and practical ethical reasoning), one would need a richer representation of paradigm cases. A paradigm case might be one that experts regard as a classic (e.g., a paradigm example of an ethical or legal principle) and frequently invoke in scholarly discussions of a concept, conferring upon it the mantle of consensus and authority. Or a paradigm case may present a particularly dramatic conflict

among principles, a real paradox. Although these senses of a paradigm case are not in PROTOS, the program does show how one sense, prototypicalness, can be used to classify new cases and in case comparisons. Interestingly, PROTOS generalizes prototypical cases by modifying their weights. If a new case is very much like an exemplar, they are merged and the exemplar's prototypicalness is increased. The particular facts of the merged case are lost, however.

Within its domain, PROTOS apparently does a good job of classification as demonstrated by empirical tests (Bareiss et al. 1989, p. 273). Copies of the program have been made available to researchers (contact either Ray Bareiss or Bruce Porter). PROTOS' negatives are slight. Its use of numeric weights limits the program's ability to make justifications symbolically. It cannot make arguments for or against classifying new problems. The classifications are either right or wrong according to the teacher. Also the paradigms are classifiable as instances uniquely of one category or another. This may be reasonable in the audiology domain, but in other domains, one would expect even exemplars to have alternative interpretations or explanations.

2.5. ADVERSARIAL OR PRECEDENT-BASED CBR

In the final paradigm, cases are precedents employed in arguments by analogy to justify assertions about how to decide a problem. Since analogies typically can be drawn to a number of precedents which may have different outcomes, justifications based on these competing precedents conflict. A reasoner taking one side's viewpoint tries to discredit analogies to opposing precedents by distinguishing the precedent (focusing on relevant differences) or citing other precedents (or hypothetical cases) as counterexamples. By evaluating the best arguments pro and con and selecting the stronger, a decision is sometimes reached. Two programs approach this paradigm differently. My own program, HYPO, pioneers a Dimensional approach, while Karl Branting's GREBE employs structure-mapping.

2.5.1. *Dimensional Approach: HYPO*

I designed HYPO as my Ph.D. dissertation project at the University of Massachusetts where Edwina Rissland was my adviser (Ashley, 1991, 1990, 1989; Ashley & Rissland 1988b, 1987). HYPO's task is to analyze legal disputes involving trade secrets law. The inputs to the program are descriptions of the dispute represented in a Legal Case Frame language. The program's outputs are 3-Ply Arguments for and against each side in the dispute (plaintiff and defendant) citing their best cases and suggested hypotheticals to strengthen or weaken a side's argument. HYPO's knowledge sources consist of a Case Knowledge Base of thirty legal cases indexed by thirteen Dimensions. Each Dimension represents a factor, a general collection of facts that strengthens or weakens the plaintiff's argument. There also are heuristics for evaluating arguments and posing hypotheticals. Beside the case frame language, there is a language of Factual Predicates for testing whether Dimensions apply to a dispute. Each case is indexed by the Dimensions that apply to it (Ashley 1990, pp. 35–43).

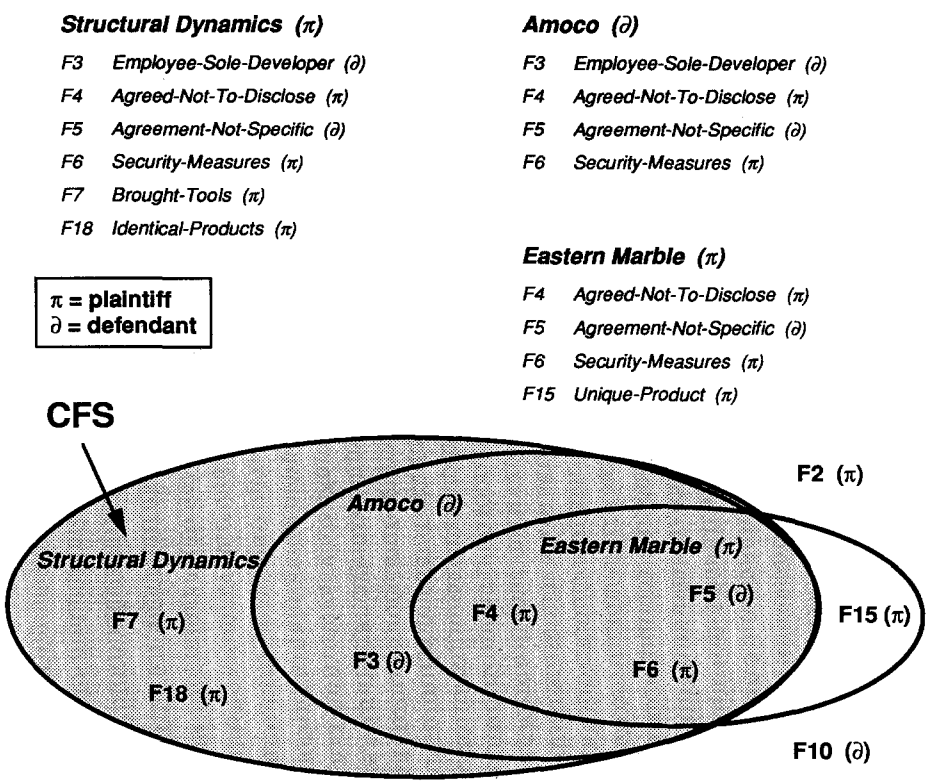

Fig. 12. Venn Representation of Legal Problem: The figure represents a problem situation called the *Structural Dynamics* case as a set of six factors, some of which favor the plaintiff (π) and others the defendant (δ) on a claim for trade secrets misappropriation. Also shown are two cases, each of which shares some factors with the problem, one of which was won by plaintiff and the other by defendant: *Eastern Marble* (π) and *Amoco* (δ). The text describes how the cases can be employed to justify assertions about how the problem's conflicting factors should be resolved.

The focus (and challenge) of work in this paradigm is to identify and model ways that past cases can have a persuasive force in an argument about how to decide a current problem. In the HYPO model, precedents provide a basis for arguing how to resolve competing factors. In a trade secret case, the plaintiff claims that the defendant has gained an unfair competitive advantage by gaining access to the plaintiff's confidential product-related information. Typically, a trade secret dispute is likely to involve a collection of competing factors, some favoring the plaintiff and some the defendant. Legal experts are likely to agree that these factors are important in deciding the case but often they disagree as to how important each factor is or how the conflict should be resolved.

For instance, Figure 12 shows a legal case called the *Structural Dynamics* case as a collection of six factors, four of which favor plaintiff; the other two favor defendant. Attorneys versed in trade secret law are likely to agree that in the *Structural Dynamics*

case, the following factors favor plaintiff's claim that the defendant misappropriated its trade secrets: the plaintiff and defendant entered into a nondisclosure agreement (f4), plaintiff took various security measures (f6), defendant's employee, who used to work for the plaintiff, brought various product development tools to his new employer (f7) and the defendant's product turned out to be almost identical to plaintiff's (f18). They will also agree that other factors in *Structural Dynamics* favor defendant: defendant's employee, when he used to work at plaintiff, was solely responsible for initiating and developing the plaintiff's product (f3) and the nondisclosure agreement did not specifically refer to the product about which the dispute has arisen (f5).

Agreement stops there, however. Legal experts, generally, will not agree how the conflicting factors should be resolved. Indeed, the advocates in a case are paid to disagree about how to resolve the factors and to find persuasive arguments to back up their opposing views.

Unfortunately, the law provides no mathematical function or authoritative weighting scheme to tell legal experts how to resolve the competing factors and decide who should win. In law, the question of assigning weights to factors and other features of cases is controversial. Attorneys may agree that one factor is usually more important than another, but recognize that within certain contexts the opposite may be true. It depends on the individual problem situation, so any weighting scheme must be context sensitive; attorneys worry that commiting themselves prematurely to a weighting sheme may cut off lines of inquiry that could lead to convincing arguments. Legal weighting schemes must also be authoritative. Judges and other authorities do not employ numerical weights in justifying decisions. In general, statistical arguments summarizing the effect of past cases, though they may be useful for predicting outcomes, are not regarded as persuasive. The weighting scheme must also support symbolic comparative evaluation. Judges expect adversaries to compare a problem situation symbolically to past cases and to draw inferences from the comparison in symbolic ways. Two sides draw analogies to competing alternative cases, attempt to distinguish or otherwise explain away the other side's cases, and point out how some cases refute conclusions based on other cases to which they are counterexamples. Attempting to reduce the comparison of cases to numerical weights obliterates the information needed to make such comparisons.

Intuitively, attorneys believe that past cases resolve competing factors. Attorneys deal with factor weights by citing precedents to show how competing factors were resolved in the past. That use of precedents can be illustrated in Figure 12. For instance, one could base an argument that the competing factors f4, f5 and f6 in *Structural Dynamics* should be resolved in favor of the plaintiff as they were in the *Eastern Marble* case. In essence, that is the persuasive force of the *Eastern Marble* case in the context of this problem. Plaintiff's advocate would cite *Eastern Marble* and draw the analogy in terms of the shared factors.

That persuasive force of a precedent is diminished to the extent that the case may be distinguished from the problem or there are counterexamples. For instance, there was an extra factor in *Eastern Marble*, justifying an outcome for plaintiff, that was not present in *Structural Dynamics*, namely that the product was unique among competitors (f15). That

is a valid distinction between the two cases that weakens *Eastern Marble's* persuasiveness. In response to plaintiff's point citing *Eastern Marble*, defendant's advocate would make this distinction.

Counterexamples also weaken the persuasiveness of a precedent. If an advocate were to cite *Eastern Marble* for the plaintiff, his/her opponent could trump the point by citing the *Amoco* case as a counterexample. *Amoco* shares every factor with the problem that *Eastern Marble* does, plus an extra pro-defendant factor (f3) implying that *Amoco* is more relevant to the problem of resolving the competing factors *in Structural Dynamics* than *Eastern Marble*. In legal parlance, *Amoco* is more on point than *Eastern Marble* relative to *Structural Dynamics* and had the opposite outcome.

In summary, to the extent competing factors are resolvable in law, advocates resolve them by symbolically comparing past cases to the problem, asserting that the conflict should be resolved in the same way as a precedent and responding to the assertion by distinguishing the cited case or citing counterexamples.

HYPO models this adversarial process and generates such arguments. Given a problem situation represented in its Legal Case Frame language, HYPO:

1. Analyzes the problem situation (referred to as the Current Fact Situation or 'cfs') to determine the Dimensions that apply and those that are near misses. Each dimension represents a factor.
2. Retrieves relevant cases from the CKB. Specifically, it retrieves every case indexed by any Dimension that applies to the problem.
3. Selects relevant cases that are most on point (i.e., most relevant) to the problem. As described below, all retrieved cases are sorted into a Claim Lattice like that of Figure 13 for this purpose.
4. Selects the most on point cases that are best for each side to cite. Each of these will be cited in a 3-Ply Argument as the basis of a point for the side.
5. Computes distinctions between the best cases and the problem. The distinctions are used in a 3-Ply Argument on behalf of the side's opponent to respond to the point by showing relevant differences between the cfs and the problem.
6. Identifies counterexamples to the best cases. The counterexamples are also used in a 3-Ply Argument on behalf of the side's opponent to respond to the point.
7. Evaluates and summarizes the overall argument. If any side is the only side that can cite cases for which there are no 'trumping' counterexamples (defined below), that side has the stronger argument.
8. For each side's best cases, generates a 3-ply Argument citing the case for side 1. This is the first ply or the 'point'. In the second ply or the 'response', it distinguishes the cited case and cites counterexamples on behalf side 1's opponent, side 2. In the third ply, the 'rebuttal', the program rebuts the response by distinguishing any counterexamples from the problem.
9. Generates hypotheticals to strengthen or weaken arguments (Ashley 1990, p. 36).

A Claim Lattice is a data structure used by HYPO in step 3 to order cases in terms of how relevant they are to a problem and provides a useful diagram for interpreting the cases in terms of arguments. Figure 13 shows a Claim Lattice[5] involving the same cases presented in the Venn diagram, Figure 12, and two additional cases. The root node represents the problem situation and the factors that apply to it. Here the cfs (i.e., current dispute) is the *Structural Dynamics* case and the same six factors apply, four of which favor the plaintiff

[5] This Claim Lattice was generated with a reduced CKB to make a point about tutorial uses of CBR in Section 2.5.2. For ease of reference, the factors have been numbered and distinguishing factors have been shown. For examples of Claim Lattices as actually drawn by HYPO, see (Ashley 1990, pp. 56, 130f, 136).

(π) and two favor the defendant (δ). The body of the Claim Lattice contains four cases: three of which were won by the plaintiff and one by the defendant. Each case is relevant to the problem because it shares some subset of factors in common with the cfs. The *Amoco* case, for instance, shares factors f3, f4, f5 and f6 with cfs and was won by the defendant (δ). *Eastern Marble* was won by the plaintiff (π) and shares only factors f4, f5 and f6 with the cfs (Ashley 1990, pp. 55–57). It has one factor not shared with the cfs, factor f15.

In the Claim Lattice the relevant cases are ordered in terms of how the set of factors they share with the cfs compares with those of the other cases. Those with the most inclusive sets of shared Dimensions (measured not by comparing numbers of Dimensions but by determining which sets are proper subsets of other sets) are closer to the root node and treated as more on point. *Eastern Marble* is less on point than *Amoco* because its set of factors shared with the cfs, {f4, f5, f6}, is a proper subset of that of *Amoco*, {f3, f4, f5, f6}. Similarly the pro-plaintiff *Schulenburg* case is less on point than the *Analogic* case which was also won by the plaintiff. *Amoco* is neither more nor less on point than *Analogic* (not because each shares four factors with the problem but because neither set of shared factors is a proper subset of the other). Of the four cases, *Amoco* is defendant's most on point case (and the best case for it to cite). *Analogic* is plaintiff's most on point case and best case to cite. A best case to cite is a most on point case that has at least one Dimension that favors the side for which it is cited (Ashley 1990, pp. 128–147).

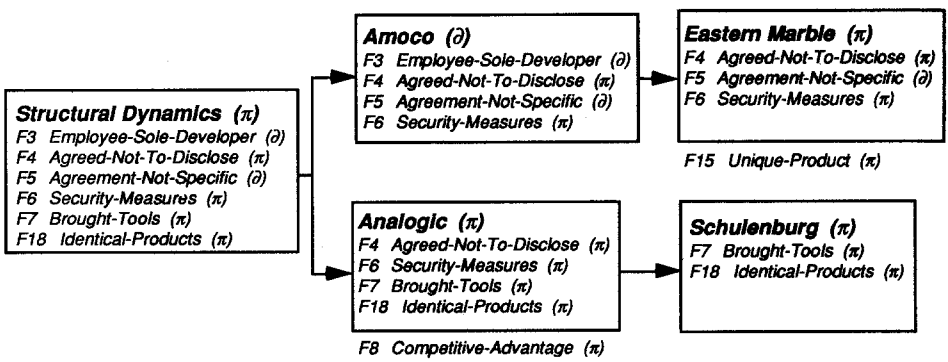

Fig. 13. A Claim Lattice

HYPO interprets the Claim Lattice as an argument how to decide the cfs (in steps 4 through 8). It generates arguments citing each of each side's best cases. Representing the defendant in the cfs, it makes a strong argument that the defendant should win by drawing an analogy to *Amoco*, the defendant's best case to cite. In making this point, HYPO draws the analogy in terms of all the factors (each represented by a Dimension) that *Amoco* shared with *Structural Dynamics*. The argument asserts that the conflicting factors in the cfs should be resolved just as they were in *Amoco*:

[1] Where: Employee defendant was the sole developer of plaintiff's product [f3]. The nondisclosure agreement did not specifically refer to plaintiff's product [f5].

Even though: Plaintiff and defendant entered into a nondisclosure agreement [f4]. Plaintiff adopted security measures [f6].
Defendant should win a claim for trade secrets misppropriation. Cite: *Amoco*.

The program argues to the contrary on behalf of the plaintiff by drawing an analogy to the *Analogic* case. It prefers citing a case that is more on point. For instance, it avoids citing *Schulenburg* for the plaintiff in favor of citing *Analogic* which is more on point:

[2] Where: Plaintiff and defendant entered into a nondisclosure agreement [f4]. Plaintiff adopted security measures [f6]. Plaintiff's former employee brought product development information to defendant [f7], and Plaintiff's and defendant's products are identical [f18].
Plaintiff should win a claim for trade secrets misappropriation. Cite: *Analogic*.

HYPO responds to such arguments by citing counterexamples and distinguishing cases (Ashley 1990, pp. 62–79). For instance, when HYPO cites *Eastern Marble* for the plaintiff, it responds by citing *Amoco* as a counterexample, a case that is more on point than *Eastern Marble* and with the opposite outcome. It points out the extra pro-defendant factor (f3) that *Amoco* shares with the cfs over *Eastern Marble*, the employee defendants in *Amoco* and *Structural Dynamics* were the sole developers of plaintiff's product. In response to argument [2] citing *Analogic*, HYPO has no counterexample to cite but it can distinguish *Analogic* on behalf of the defendant. It points out that *Analogic* has a factor (f8) that make it stronger for the plaintiff than *Structural Dynamics*. This is a relevant distinction, since these unshared factors can be used to suggest that the result in *Analogic* may have more to do with the fact that the defendant gained a competitive advantage than with the factors it shares with the cfs. It also points out that *Structural Dynamics* has some factors (f3 and f5) that favor defendant, which were not present in *Analogic*.

Finally, HYPO poses hypothetical variations of the problem that would strengthen or weaken a side's argument (in step 9). The hypotheticals apprise the user that it would be desirable to prove, or prevent the opponent from proving, certain additional facts which would lead to new case-citing arguments or responses. HYPO looks for Dimensions that are near-misses with respect to the cfs (i.e., all of a near-miss Dimension's prerequisites are satisfied except one which is designated in advance as being central to the Dimension's meaning). If the near-miss Dimension indexes a case that would strenghthen a side's argument, for instance, by turning an 'as on point' counterexample into one that is more on point than a case cited for the opponent, HYPO suggests varying the problem so that the near-miss Dimension applies (Ashley 1990, pp. 84–86). The program can demonstrate the effect of the change on the 3-Ply Argument (Ashley 1990, pp. 82–84).

HYPO also compares cases 'along' Dimensions, a kind of reasoning not evident in either Figures 12 or 13. Some Dimensions have numerical ranges to capture a sense in which a case may be a more or less extreme example of a factor (Ashley 1990, pp. 107–126). For instance, a defendant may save a larger and larger percentage of the plaintiff's development time and costs (the competitive advantage Dimension, factor f8). Secrets may have been disclosed to more and more outsiders (the secrets-disclosed-to-outsiders Dimension, factor 10). These Dimension magnitudes do not measure the Dimension's 'weight', but they do provide a sense in which one case is stronger or

weaker for the plaintiff along a Dimension. Such magnitudes are useful in distinguishing cases, for example, pointing out that in one case there were many more disclosures to outsiders than in another, or in finding counterexamples that show that even extreme values of a factor do not necessarily change the result (boundary counterexamples) (Ashley 1990, pp. 69–70, 145–146).

HYPO provides yet another example of how to reason symbolically about differences. Reasoning with Dimensions allows HYPO to determine whether a difference is significant in context (Ashley 1990, pp. 167–174) and shows that reasoning with Dimensions about partial matches is possible even though there is no strong domain theory. In Figure 12, for instance, *Eastern Marble* is a partial match: it differs from the cfs because it has factor f15 but does not have f3, f7 and f18. In the context of making an argument for defendant, HYPO treats the lack of f3 as significant because that is the extra factor that turns *Amoco* into a trumping counterexample. HYPO also treats f15 as important because it is a relevant difference between *Eastern Marble* and the cfs from the defendant's viewpoint; f15 does not prevent HYPO from citing *Eastern Marble* for the plaintiff but HYPO treats it as a distinction in responding for the defendant: there was an extra reason why plaintiff won in *Eastern Marble* that did not apply in the cfs.

HYPO's model can tell when a difference does not matter enough in the context of an argument to amount to a distinction. Suppose there were another case, call it *Western Marble*, which also shared factors f4, f5 and f6 with the cfs and also had one factor that the cfs did not have, just like *Eastern Marble*, but that factor was not pro-plaintiff f15 but pro-defendant f20. Even though it is a difference between *Western Marble* and the cfs, factor f20 is not a distinction from the defendant's viewpoint. If HYPO cited *Western Marble* for the plaintiff and then distinguished it from the defendant's viewpoint, HYPO would ignore f20. Indeed, calling attention to f20 would not be in the defendant's best interest. It makes *Western Marble* even more persuasive for plaintiff because it shows a case where the factors f4, f5 and f6 were resolved in favor of plaintiff despite the existence of a pro-defendant factor not present in the cfs.

Reasoning about differences represented by Dimensions allows a program to select among partially matched cases. Suppose plaintiff's advocate has to choose between citing either of *Eastern Marble* or *Western*. Both cases are equivalently on-point since they each share the same set of factors with the cfs. By reasoning about the effect of the differences on the argument (i.e., either pro-plaintiff f15 or pro-defendant f20), however, the program could reasonably prefer *Western Marble* as a better case to cite for plaintiff than *Eastern Marble*.

In summary (see Figure 14), using Dimensions, HYPO symbolically compares a problem and cases in generating arguments about who should win. It makes the strongest arguments for each side in light of the ways in which past cases have resolved conflicting factors. It does not assume a strong causal model of the domain; all it assumes is that one can identify factors that tend to strengthen or weaken a side's argument and represents them with Dimensions. It assesses relevant similarities in context; whether a case is cited and how the program describes the case depends on the facts of the problem, the role of the case in an argument (as part of a point, counterexample, distinguished case, or hypo-

thetical) and the facts of all the other cases in the CKB. It does not assume one right answer to a problem, but provides alternative reasonable answers. It also combines cases in arguments and poses meaningful hypotheticals.

+ HYPO compares cases symbolically with Dimensions.
+ It does not assume a strong domain theory.
+ HYPO assesses relevant similarities in context.
+ The program does not assume one right answer, and it combines cases in arguments.
+ HYPO poses meaningful hypotheticals.
– Factors are treated initially as equally weighted.
– The program does not deal with rules or causal explanations.
– It assumes legal cases have binary outcomes.

Fig. 14. Adversarial CBR: HYPO's benefits and costs.

On the other hand, HYPO assumes each case has a binary outcome. Even in law that is an oversimplification.[6] It does not deal with rules or causal explanations, both of which are important in legal reasoning . It captures only one way of dealing with the problem of factor weights. (It treats a factor as important to the extent that it makes a case or counterexample more on point than any other case). HYPO does not capture, however, other ways in which legal experts purport to deal symbolically with the problem of assigning weights to factors, such as by defining a factor's importance in a legal rule or by invoking general principles to justify a factor's importance.

2.5.2. *Tutoring with the HYPO model*

In recent work, my Graduate Research Assistant, Vincent Aleven, and I have been designing a tutorial program, based on the HYPO model, to teach students to argue with cases. Our Case/Argument Tutorial program, CATO, is intended to give law students practice in making arguments. The system will present exercises, including the facts of a problem and a set of on-line cases, and instructions to make, or respond to, a legal argument about the problem. The student/user will have a set of tools to analyze the problem and fashion an answer comparing it to other cases. The program will respond to a student's arguments, for example, by citing counterexamples, and provide feedback on a student's problem solving activities with explanations, examples and follow-up assignments (Ashley & Aleven 1991, 1992; Aleven & Ashley, 1992).

Initially, the system will present exercises to teach law students basic issues of arguing with cases, like: (1) How can one analyze a problem to determine which cases are relevent? (2) What are the minimum criteria for citing a precedent? (3) What are criteria for preferring to cite one relevant precedent over another, including determining which is more on point? (4) which is less distinguishable? (5) which cannot be trumped by the opponent? (6) How can one respond to points by distinguishing the cases cited and citing

[6] For instance, the case representation does not take into account the procedural posture of the case (Berman & Hafner 1991). Also, cases may involve decisions of more than one issue.

counterexamples? and (7) How can one pose hypothetical variations of a problem that would improve an argument for one side or another?

Each tutorial exercise is embodied in an 'Argument Context', a collection of a problem situation and cases that is interesting pedagogically because it raises one or more of the above issues. Argument Contexts are drawn from an on line case database. Using an Argument Context, CATO will pose a problem to the student, like 'Of the following cases, pick the best case to cite for the plaintiff in this situation.' In providing an answer, if the student has missed something, like the fact that the student's selected case was won by the wrong side or is not relevantly similar to the problem or has a more relevant counterexample, the tutorial program will respond accordingly, using information contained in the Argument Context. The system will also drive home the lesson with follow-up exercises based on new Argument Contexts.

The reader has already seen an Argument Context in Figure 13. This Argument Context is embodied in a Claim Lattice. The tutoring program generated it because its cases relate to one another in such a way that one could use them to teach issues 2, 3 and 5. That is, the Argument Context represents a pedagogical opportunity because, in order to find the best case, one has to:

(a) know the minimum criteria for citing a case (Issue 2),
(b) know that the more on point a case, the better it is to cite (Issue 3),
(c) recognize the trumping counterexample, and realize that untrumped points are better than trumped ones (Issue 5).

More specifically, although *Analogic*, *Eastern Marble* and *Schulenburg* are all citable for plaintiff, *Analogic* is the best precedent. It is better than *Eastern Marble*, because *Eastern Marble* can be trumped (by *Amoco*), whereas *Analogic* cannot be trumped (Issue 5). *Analogic* is better than *Schulenburg* because it is more on point, and according to Issue 3, a more on point precedent is better. *Analogic* is better than *Amoco*, because *Amoco* cannot be cited on behalf of the plaintiff – it does not satisfy the minimum requirements (Issue 2).

CATO will be able to teach these basic issues in the context of specific examples. Here is a sample of what, we hope, the tutorial program will be able to do with the information in the Argument Context of Figure 13.

Tutor: The current fact situation is: *Structural Dynamics*. Suppose you are plaintiff's attorney. Which of the following cases would you cite: *Schulenburg*, *Amoco*, *Eastern Marble* or *Analogic*?
[Suppose, after employing various system-provided tools to read and analyze the cases, the student replies, 'Cite *Eastern Marble*'. The system would respond:]
Tutor: If you cite *Eastern Marble* for the plaintiff, defendant will trump your point by citing *Amoco* as a more on point or trumping counterexample.
In *Eastern Marble*, where the defendant employee entered into a nondisclosure agreement (f4) and plaintiff took security measures (f6), plaintiff won even though the agreement was not specific (f5).
All those factors were present in *Amoco*, but there, plaintiff lost where it was also true that the employee was the sole developer of the information (f3).

A basic difference between HYPO and CATO's Argument Context Generator module involves the flexibility of queries (Aleven & Ashley 1992). Queries to HYPO required the user to specify a case to use as a problem situation. For instance, the user specified

Structural Dynamics as the cfs and HYPO returned a Claim Lattice like that in Figure 13. By contrast, queries to the new Generator module can ask for sets of cases that satisfy certain more abstract constraints. The constraints may include a set of issues which the Argument Context should raise as well as other limitations on the factors or cases involved. This means, in the example, that the user can request all sets of five cases which are related to one another in such a way that they raise Issues 2, 3 and 5. The program returns the Claim Lattice in Figure 13 and any other five case sets that satisfy the constraints. That kind of retrieval is impossible for HYPO.

The new flexibility is feasible because we have adopted an express representation of concepts and relations that were only procedurally represented in HYPO. Using LOOM, a large knowledge representation system which has a query language similar to first order predicate logic (MacGregor 1988), various relevance concepts and relations from HYPO are defined in terms of logical expressions, including the concepts: shared factor, relevantly similar, citable, more on point, most on point, best case to cite, trumping counterexample, and untrumped best case. These relations are employed, in turn, to define other, more complex relations, including queries for Argument Contexts. The query that generated the Argument Context in Figure 13, for instance, invoked three of these concepts. In English, it says:

Retrieve cases ?cfs, ?c1, C2, ?C3, ?C4 such that:
?c1, ?c2, and ?c3 are citable for plaintiff
?c2 is more on point than ?c3
?c4 is a trumping counterexample for ?c1
?c2 and ?c3 are in a different branch than ?c1 and ?c4

Note that it is a query for five cases, including one case to serve as the cfs, satisfying various relations that insure the retrieved Argument Context will raise Issues 2, 3 and 5.

Taking advantage of our flexible query language, it has been easy to define criteria for examples of obvious utility in explanations. Already, the module can produce examples of *ceteris paribus* comparisons: two cases with different outcomes whose facts differ only with respect to the presence of one particular factor. A student or the program could use such cases to demonstrate that the factor is legally significant. The program can also generate 'cover the bases' examples: Argument Contexts where a side can cite cases to counteract each of an opponent's strengths while making minimal use of his own strengths (Ashley & Aleven, 1992).

In general, it is a difficult and time-consuming exercise for a human being to think up Argument Contexts. While agreeing that the five-case Argument Context would be a useful exercise in a first year legal methods course, an instructor at the University of Pittsburgh School of Law estimated it would take hours or days using traditional or on line legal research methods to come up with five cases so related. From a database of 26 cases, our module was able to generate the Argument Context of Figure 13 and 57 others in 5.9 seconds pursuant to a query like the one described above. The program enables filtering and ranking Argument Contexts according to various criteria of pedagogical utility. We hope that this flexible, speedy retrieval will enable the tutorial program to

present follow-up exercises based on new Argument Contexts which it generates dynamically on line.

We have used the program module to generate a variety of Argument Contexts and employed them in a preliminary experiment to ascertain whether they are, in fact, useful in tutoring argument skills. We divided a group of five first year law students into 'experimental' and 'control' groups. All students took a one hour, written pre-test to provide a baseline indication of their case-based argumentation skills. I then manually conducted a seventy-five minute tutorial session for the experimental group. During this session, I employed as examples only Argument Contexts generated by our program. Finally, all of the students took a post-test. The questions in the pre- and post-tests also employed only examples generated by the program. Both tests were scored on a five point scale according to whether the student had taken advantage of argument-making opportunities implict in the problem. The results showed a clear difference between the groups: both groups did equally well at the pre-test, but on the post-test the experimental group students performed substantially better than those in the control group (Aleven & Ashley, 1992). The results, and protocols of tutoring sessions with second year law students, tend to confirm the utility of Argument Contexts in teaching students to analyze problems, cite analogous cases, distinguish them, prefer more on point cases, frame queries for useful cases to cite, and respond with trumping counterexamples.

2.5.3. *Structure-mapping Approach: GREBE*

Karl Branting, another of Bruce Porter's doctoral students at the University of Texas, designed a program called GREBE which employs explained legal precedents and structure-mapping to classify and decide new problems (Branting & Porter 1991; Branting 1991a, b). GREBE analyzes workmens' compensation problems by employing explanations from exemplary precedents. The inputs to the program are fact situations; the outputs are arguments explaining why an individual is entitled or not to workmen's compensation under Texas law. GREBE's knowledge sources include fifty-seven statutory, common law and common sense rules, 132 semantic rules, sixteen legal precedents, four paradigm cases and twenty-one hypothetical test cases, all involving workmen's compensation awards (Branting 1991b, p. 24).

GREBE's cases represent not only collections of facts to which a judge has assigned an outcome but also some parts of a court's explanation of its reasoning. Semantic networks of relations among objects represent both the facts and the explanation. With respect to the explanations, a case enterer has drawn links from certain of the facts, the so-called criterial facts, to the legal predicates whose application the facts support (Branting 1991b, pp. 29–39). Criterial facts are those which the judge regarded as sufficient for establishing a legal conclusion. These and other links in the case explanation are labeled to show the roles they play such as criterial facts, explanations of intermediate conclusions, antecedents of statutory rules, or consequents. For example, take the *Vaughn* case, parts of which are pictured in Figure 15. Relations among objects in a semantic net represent the criterial facts that (1) an employee (Vaughn) had intense hunger, which (2) impedes

his duties as the driver of a sulphur truck and that (3) having food at a restaurant decreases his intense hunger. Labeled links in the network represent that, together, criterial facts (1), (2) and (3) are sufficient to conclude that having food was 'reasonably essential' to Vaughn's job duties. That conclusion, along with the fact that Vaughn had to travel to a restaurant to get food, explain a conclusion that the traveling 'was in furtherance of employment', a technical legal requirement of statutory rules determining whether an injury sustained in the course of the employment was covered by workman's compensation. The consequent of this and other similarly explained conclusions is that the company is liable to Vaughn. Like the paradigm explanations in PROTOS, these explanations (Branting calls them EBE's or Exemplar-Based Explanations) do not purport to state the necessary and sufficient conditions for a conclusion but only the facts that this decision maker deemed important for the conclusion (Branting 1991b, pp. 29–39).

Given a proposition about the problem, such as an assertion that an employee's injury is covered by workmen's compensations, GREBE attempts to create an explanation justifying the proposition. The proposition is characterized as a predicate relation, 'worker's compensation liability' and corresponding arguments including the employer, employee, injury and activity. GREBE's recursive basic algorithm sets out to find an explanation that the predicate applies given the arguments. It employs both rules and precedents to substantiate the classification. The rules are applied in a backward chaining manner. Best-matching precedents are employed to provide arguments by analogy that a predicate applies or not. Each analogy is based on a mapping onto the problem facts of the structure of the precedent's explanation of why the predicate does or does not apply (Branting 1991b, pp. 25–29).

In the process, both the rule-based and case-based methods identify additional subpredicates that require explanation. GREBE's algorithm is applied recursively to find explanations of these additional predicates. Some of them are rule antecedents which need to be established. Others are missing criterial facts which, if present in the problem, would tighten the analogy to a precedent (Branting 1991b, pp. 25–29). The recursive algorithm constructs its overall explanation for the input predicate from the explanations produced for the various sub-predicates.

Similarity in GREBE is measured by the proportion of criterial facts missing per total criterial facts in a match between a problem and a precedent, another variant of the overlap metric (Branting 1991b, p. 53). GREBE attempts to match the problem to all of the precedents using a structure-mapping algorithm. It proceeds in a best-first fashion, however, so that candidate mappings are pursued only so far as they lead to a best match. (The best-first mapping is modified so that a user-specifiable number of 'best-matches' are returned.) There is also a match-improvement process in which the program, having kept track of the criterial facts that could not be matched, attempts to infer them using GREBE's basic algorithm (Branting 1991b, pp. 42–49).

In presenting its explanations, GREBE qualifies its analogies by pointing out missing criterial facts and also draws analogies to contrary cases (Branting, 1991b, pp. 69–72). For instance, in the *Jarek* problem where an employee, Jarek, suffered an injury while

walking several blocks to tell his wife that his employer had asked him to work late that night, GREBE states that the fact that Jarek's need to tell his wife did not impede his duties for his employer is a relevant difference from the *Vaughn* case, referred to above, where the employee's need for food would have impeded his duties (see fact (2) in Figure 15) (Branting 1991a, p. 823) and from the *Janak* case where the employees' need for ice water impeded their ability to drill for oil (Branting 1991b, p. 72). The impediments were criterial facts in the *Vaughn* and *Janak* cases, not matched in the *Jarek* case, which means a weaker analogy.

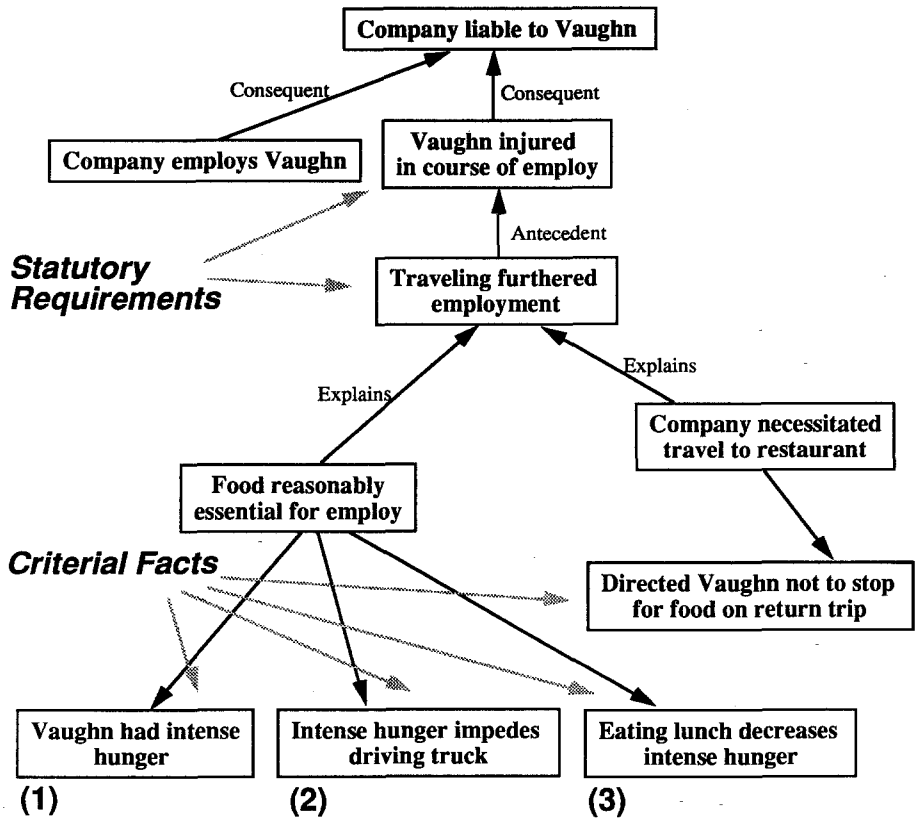

Fig. 15. Part of semantic network representing the *Vaughn* case in GREBE (simplified).

GREBE's strengths (and weaknesses) are summarized in Figure 16. Chief among them are its ability to compare cases symbolically with structure-mapping and to integrate into its reasoning legal rules and courts' causal case explanations. Structure-mapping enables it to compare cases symbolically without assuming a strong domain theory. GREBE knows that a similarity is important in an analogy because it relates to a criterial fact in the explanation of a precedent. Conversely, it knows that a difference between two cases is important because something that was a criterial fact in the precedent is not matched.

On the other hand, relying on the presence or absence of structurally matches to signify relevance has limitations. It assumes that explanations which map isomorphically do, in fact, apply to matched problems and confer explanatory importance on the matched facts regardless of other facts in the problem which may not match. Unlike CASEY, GREBE has no causal inference network with which to assess such other facts.

There is another reason why GREBE would seem to require a more explicit representation of the significance of facts and their absences. GREBE knows that permitting the employee's trip is different from directing it, but does not appear to be able to explain why the fact that the employer in *Vaughn* directed, while the employer in *Jarek* only permitted, the employee's trip makes the *Vaughn* case stronger that the *Jarek* case on the issue of 'furtherance of employment' (Branting 1991a, p. 831). Similarly, GREBE knows that the fact that Jarek's need to tell his wife was a relevant difference from the the need for ice water in the *Janak* case. In the latter, failing to satisfy the need impedes the work, but nothing in the *Jarek* case matches that impediment. GREBE concludes that the analogy is weaker, but it does so only because the impediment relation was a criterial fact in the *Janak* case that is not matched in the *Jarek* case. GREBE does not, I believe, have any other knowledge about why the impediment strengthens the conclusion in the *Janak* (or *Vaughn*) case (Branting 1991a, pp. 823, 831). It is hard to see, for instance, how the program could modify a case to present a more or less extreme example of an impediment. That kind of knowledge simply is not there.

Since structure-mapping is complex and match-improvement may involve mappings to other precedents as well, GREBE's algorithm is computationally expensive (Branting 1991b, p. 42). Structure-mapping also places a premium on consistently representing the explanations of cases (to facilitate matching). Small superficial differences in the way semantically similar facts are asserted could defeat a match. It is not clear whether GREBE enforces any standards in case representation.

GREBE was tested in a dramatic experiment. Law students were asked to write a brief analysing a problem within GREBE's domain of expertise employing traditional and on-line legal research methods. An expert then graded their answers and GREBE's, finding that the quality of the analyses was comparable (Branting 1991b, pp. 89–93).

+ GREBE compares case symbolically with structure-mapping.
+ It does not assume a strong domain theory.
+ It does not assume one right answer, but combines cases in arguments and counterarguments.
+ GREBE deals with legal rules and a court's causal explanations.
+ It uses parts of cases in constructing explanations.
+ The program has been evaluated empirically.
− It employs computationally expensive structure-mapping.
− It places a premium on representing cases the same way but provides little guidance.
− Relevant differences are absences of structural matches.

Fig. 16. Adversarial CBR: GREBE's benefits and costs.

3. Comparing the CBR Paradigms

The five Case-Based Reasoning paradigms and their representative programs illustrate diverse ways to reason with cases. AI research has progressed at least this far (and, indeed, farther as indicated in the next section). The question of which approach is best depends on application requirements such as the kind of case-based inference to be performed and the extent of analytic theory available for the domain and task. Each program makes different assumptions about a domain theory and presents different benefits and costs. Comparing the paradigms may spur insights on how to approach specific domains and tasks, for instance, in law-related applications.

The paradigms can be compared along the criteria summarized in Figure 17. Although there are other ways to compare CBR systems (e.g., in terms of memory design or implementation decisions), these criteria highlight the paradigms' significance as alternative ways to conceptualize and implement case-based tasks. The criteria can be grouped into five general topics: domain/task assumptions, relevance assessment, explanation and argument, cases and meanings, and knowledge acquisition. To facilitate the comparison, I will not document the assertions about each program but rely instead on the program descriptions in the previous section.

3.1. DOMAIN/TASK ASSUMPTIONS

Case-Based Inference Type: The paradigm programs fall into the two basic camps in terms of the type of case inference they support: those that employ cases (1) to constrain search by mapping a solution or (2) for comparative evaluation. CASEY, CHEF, and PERSUADER employ cases primarily to constrain search by mapping a past solution onto a problem. CASEY maps an explanation, CHEF a recipe (i.e., plan), and PERSUADER maps settlements. By contrast, MBRTALK, PROTOS, HYPO, and GREBE employ cases as a means for comparative evaluation. Their case comparisons serve as the basis for an argument that the problem should be assigned a particular value. GREBE is a kind of bridge between the two camps since it employs structure-mapping in service of comparative evaluation.

Domain Model Strength: Each paradigm program assumes some kind of analytical domain/task model for implementing its index or some aspect of its reasoning. The programs differ in terms of how strong a model they assume and the way the model is worked into the program's inferences.

In that a strong analytical model is one that supports deductive reasoning, CHEF assumes the strongest model of any of the programs. CHEF's simulator's cooking rules enable it to deduce whether a recipe is a gastronomical success or failure, without which capability the program could not assign credit or blame for the failure. (CHEF also employs what might be considered a weaker domain model of planning including goals, goal hierarchies and modification heuristics.) CASEY comes next. It justifies matches with the HEART FAILURE program's causal inference network which alone is sufficient to diagnose conditions using a combination of deductive and heuristic reasoning.

1. *Case-Based Inference Type*: Does the program use cases to constrain search by mapping a solution or for comparative evaluation?
2. *Domain Model Strength*: Does the domain task have, and does the program assume, a strong analytic domain model? If not, how does the program compensate for the lack of a strong model?
3. *Generalization of Cases*: How does the program generalize cases for purposes of indexing and feature weights?
4. *Partial Match Assessment*: How does the program reason about the significance of differences among cases?
5. *Context Sensitivity*: How sensitive to context are the programs in their generalizations and case relevance assessments?
6. *Arguing Pros and Cons*: Does the program assume a problem has one right answer or does it argue about alternatives?
7. *Explaining with Examples*: Dose the program explain its advice with case examples? Does it compare the relevance of cases symbolically? How does it explain a feature's significance?
8. *Combining Cases*: In creating a solution, does the program combine contributions from various relevant cases?
9. *Annotating Concepts*: Does the program employ cases to provide or supplement meanings for terms and concepts?
10. *Paradigmatic Cases*: Does the program treat cases as 'Classic' or paradigmatic examples?
11. *Case Interpretation*: To what extent does the program support varying interpretations of past cases?
12. *Automated Case/Index Acquisition*: does the program support automated entry of new cases and new index terms?
13. *Learning*: Can the program learn from cases?

Fig. 17. Thirteen criteria for comparing CBR paradigms.

At the other extreme, MBRTALK assumes only a very weak model of the domain of word pronunciation. Its dissimilarity metrics statistically approximate feature weights whose effects might otherwise be supplied by a stronger analytic model including pronunciation rules (for an example of the latter, see Section 4.3.2).

The other paradigms fall between these extremes, probably in the following order of strength of assumed domain models: PERSUADER, PROTOS and GREBE, and HYPO. Like CHEF, PERSUADER employs a weaker planning model including goals, solution strategies and adaptation heuristics to assess and adapt relevant cases. To the extent that PERSUADER puts solutions together drawn from multiple cases, it also needs some domain theory for insuring that solutions are compatible and for adapting a solution to a problem's constraints. To the extent that it also solves problems 'from scratch' when similar cases are not available, it implements an even stronger domain model. PROTOS eschews a strong analytic model defining necessary and sufficient concept conditions in favor of its category structure. The combination of exemplars, expert-provided classification explanations and heuristic weights substitutes for a strong model. (PROTOS can also acquire and augment its domain model by interrogating the expert.) Similarly in GREBE, the structured explanations of precedents and the facilities for mapping them on to a problem substitute for a strong model. HYPO assumes a relatively weak model of the domain, Dimensions, in order to comparatively evaluate cases. Dimensions capture domain specific information about strengthening or weakening a classification argument without defining necessary and sufficient criteria.

3.2. RELEVANCE ASSESSMENT

Generalization of Cases: In all of the paradigms, a case is meant to have significance beyond its own facts and to apply to a wider range of problem situations. A case can only be used to solve a new problem if it can be retrieved and modified successfully. A case's range of applicability depends on the way it is indexed and on the program's modification strategies; both implicitly generalize the case. The paradigms achieve such generalization in very different ways:

- In MBRTALK, each case's every feature is, in effect, an index; the program statistically generalizes feature weights directly from all of the cases to refine the overlap similarity measure.
- In CASEY, classifications and features that explain a patient classification; are the indices. The justified match determines how generally to apply a past case's explanation.
- CHEF's and PERSUADER's indices comprise goals, goal conflicts and plan features. Feature significance depends on whether prior cases resolve conflicting goals. Heuristic strategies for adapting past case solutions generalize their applicability. PERSUADER also stores explicitly generalized cases in its discrimination net.
- Dimensions are HYPO's index. Legal experts generalize collections of facts that tend to strengthen or weaken claims. In an argument that the side favored by a Dimension should win, the Dimension's symbolic weight increases to the extent that one can cite an analogous case with few distinctions and no counterexamples.
- In PROTOS and GREBE (like CASEY), indexing features come from the expert's explanations of case classifications. PROTOS's heuristic feature weights reflect two aspects of prototypicalness, the feature's significance in the expert's explanation and its utility in previous classifications. In GREBE, generalization is achieved by structurally mapping explanations (and parts of explanations) from precedents to new problems. PROTOS explicitly generalizes cases when it merges exemplars.

Match Assessment: In assessing similarity, each program selects the 'closest' case (or cases) with the highest similarity 'score' based on some maximizing metric applied to the generalized features referred to above. The programs must still decide, however, whether even the closest case is close enough. If the generalized features have been well selected and the database has a representative set of cases, the closest case should be relevantly similar to the problem. But even the closest case may still only partially match the problem.

The CBR programs illustrate some ingenious mechanisms for symbolically assessing partial matches. Some of their methods account for differences as well as similarities and evaluate those differences symbolically. CASEY's justified match attempts to explain away differences in terms of its strong theoretical model. Partial match assessments in PROTOS, HYPO and GREBE work despite the lack of a strong analytical model. PROTOS employs the heuristic feature weights to assess unmatched facts; its difference links lead to other plausible exemplars. HYPO's arguments distinguish partially matched cases by explaining their opposite outcomes in terms of unshared Dimensions that justify different results. GREBE assesses the quality of partial matches by noting missing criterial facts and attempts to improve the match using rules and other precedents to infer linkages.

Context Sensitivity: The very fact that the paradigm programs compare problems to specific cases suggests a more context sensitive approach than applying rules. Typically

in AI programs, rules are applied to any situation they match regardless of extra problem facts which would lead an intelligent reasoner to question the rule's application and regardless of the existence of counterexamples, exceptions or the need for rule modifications. This may be a context sensitive approach if the rules are very good rules (i.e., the rules have few exceptions and the rule's conditions bar the rule's application where it would be inappropriate). Such rules, however, may not be available. In domains like law, there are practical limitations to how refined the rules can be; the rules' conditions cannot identify and exclude all exceptions.

Comparing the paradigm programs in terms of this criterion is difficult because the meaning of 'context' varies for each paradigm. In their generalizations and case relevance assessments, however, some are more sensitive to context than others. MBRTALK deals most directly with assessing the weights of features in a context sensitive manner; it recomputes the weights for every problem. The computation of feature weights is tailored to the specific new word to be pronounced and automatically takes into account any new cases that may have been added to the database. There are no rules that freeze conditions or become stale. In PROTOS, by contrast, the heuristic feature weights that capture prototypicalness reflect past classification utility and are not likely to be context sensitive. In HYPO, a Dimension's significance is somewhat context sensitive. Although the Dimensions are treated initially as having equal significance in a problem, the distribution of relevant cases in a Claim Lattice and the opportunities for argument they present make certain Dimensions symbolically more important than others. CASEY's justified match procedure affords it a measure of context sensitivity. In effect, it assesses the symbolic weights of mismatched features on a case-by-case basis in light of its overall causal inference network. The ability of PERSUADER, CHEF and CASEY to adapt past plans to new circumstances offers context sensitivity in a different sense. Even though the case is relevant in terms of more general criteria, the programs can tailor it to the particular problem constraints.

3.3. EXPLANATION AND ARGUMENT

Arguing Pros and Cons: In a case-based paradigm, explanation is likely to involve arguing about alternatives. None of the paradigm programs generates a proof to explain an answer. Instead HYPO and GREBE argue a conclusion's pros and cons from opposing viewpoints and their explanations draw comparisons to competing cases and counterexamples. (PERSUADER also creates arguments urging parties to adopt a settlement although it is not clear if it argues both sides).

For these CBR programs, arguments pro and con are a strategy for dealing with ill-structured problems in domains and tasks whose analytical theories are too weak to yield 'the' logically correct answer. For any proposed answer, these programs assume that there will be competing reasonable arguments. They also treat the competing arguments expressly. They do not collapse these reasons into some hard-to-explain numerical confidence measure. In the traditional expert systems approach, at best, alternative answers are ranked in terms of confidence factors or likelihood (e.g., MYCIN presented all alternative answers ranked by confidence measures). The reasons pro and con a

particular answer are subsumed into a numerical ranking. Rarely are counterarguments expressly considered, much less presented to the user.

Even where the domain theory is strong, considering alternatives can be a valuable explanatory tool. Ideally, planning and design-oriented CBR programs should facilitate browsing through alternative design solutions and enable users to appreciate the designs' trade-offs of pros and cons. In the course of problem solving or in response to user feedback, PERSUADER and CHEF serially present alternative solution plans until the user or the program's critic expresses satisfaction; a parallel presentation of alternatives would be more helpful for a design assistant. Presenting alternative diagnoses is also important. Apparently, CASEY did not present alternatives that HEART FAILURE, the non case-based program, would have presented.

Explaining with Examples: All of the paradigm programs can explain their advice in terms of specific comparisons between the problem and one or more past cases as examples. Since examples are an important component of explanations (widely ignored in AI), this capability is significant. The programs that generate arguments (HYPO, GREBE and PERSUADER) or that present proposed matches to a teacher (PROTOS) focus particularly on incorporating cases into explanations.

In a case-based paradigm, good explanations require a capability for symbolically comparing cases. Of the five paradigm programs, only MBRTALK compares cases in purely numerical terms: measuring the degree of overlapping features and adjusting with statistically computed weights and corrections. As we have seen, the other programs all symbolically compare cases. They differ, however, in terms of how well they can explain why a relevant case matters. Although they draw an analogy between the problem and the case to justify a conclusion about the problem, they may not be able to explain why the feature justified the outcome of the case. Explaining the significance of a feature requires relating the feature to something else, something more general, such as a causal model of the domain. The programs with weaker models can only relate a feature to still other examples. Programs with stronger analytic models such as CASEY and CHEF can explain significance in terms of their models: CASEY's causal model of heart disease or CHEF's causal model of cooking. By contrast, PROTOS and GREBE can only explain a feature's significance by its presence or absence in a prototypical exemplar or explained precedent. The Argument Context generator of CATO can construct some interesting examples to explain a feature's (i.e., factor's) significance such as a *ceteris paribus* comparison, and HYPO could pose hypotheticals to exaggerate factors, but neither can relate the features to broader theories of the domain.

3.4. CASES AND MEANINGS

In a number of the paradigms, cases play a special role in representing meanings, a useful feature for domains such as law.

Annotating Concepts: Cases can represent a concept's meaning. Of the paradigm programs, PROTOS employs cases most explicitly to supplement the meanings of its

classification concepts. The category structure's linkage of concepts to exemplars plays an intimate role in PROTOS' classification process. HYPO's cases supplement the meaning of the concept 'trade secrets misappropriation'; they show how competing factors have been resolved. For a similar use of HYPO's Dimensions to supplement the meanings of concepts associated with the terms of a statutory legal rule, see CABARET, discussed below.

Paradigmatic Cases: Paradigm cases often play a very useful role in reasoning. Not just instances, they are examples distinguished by their prototypicalness, thematic purity, special clarity, quality of posing a classic dilemma, wide backing or authoritative sanction. Although none of the paradigm programs makes as robust use of paradigm cases as human reasoners do, PROTOS' undeniably reasons with prototypical exemplars.

Case Interpretation: A past case may have more than one meaning, interpretation, or explanation. None of the paradigm programs has a robust ability to support different meanings for the same case, but there are intimations. PERSUADER, which employs cases for different purposes, ascribes more than one meaning to a case. A precedent in HYPO typically has more than one interpretation. HYPO describes cases differently depending on the context of the argument; it emphasizes the features that are salient to making the best use of the case in a particular argument role given the facts of the problem and the other relevant cases.

Combining Cases: A single case is not always the right unit of information. In creating a solution, some of the paradigm programs combine information from various relevant cases. For each domain task, it is important to ascertain whether the cases that compete for selection as 'relevant' may contribute useful information even though they are not picked as the most relevant. For instance, PERSUADER draws settlement provisions from various past cases to piece together an overall solution. HYPO's arguments and hypothetical suggestions combine information from the most relevant pro and con cases, counterexamples, and nearly applicable cases that could affect the argument. GREBE's arguments also combine positive and negative examples. MBRTALK combines cases in predicting pronunciations. By contrast, CHEF, CASEY and PROTOS pick the best matched case and do not combine information from the contenders (although PROTOS may follow a difference link to another exemplar).

3.5. KNOWLEDGE ACQUISITION AND LEARNING

Automated Case/Index Acquisition: Some of the programs support automated entry of new cases. MBRTALK, PERSUADER, CHEF, CASEY, and PROTOS all support recording new problem-solving episodes as reusable successes or failures. Such cases are 'authoritative' in so far as experience shows they solve a problem (or failed to) as authenticated by the user (MBRTalk, PERSUADER, PROTOS) or by the program's internal critic (CHEF and CASEY). PROTOS provides the most highly developed facilities for

automatically acquiring new indexing information directly from the expert. The capacity to revise indices automatically also exists to varying degrees in PERSUADER and CHEF. In HYPO, the program assists a user to add new problem situations to the program's Case Knowledge Base under the Dimensions that HYPO determines apply. Since a problem is not an authentic 'case' unless it has actually been decided by a court, such cases are stored as 'hypothetical cases'. Other aspects of HYPO's processing, like the arguments generated, are not stored.

Learning: The discussion of the paradigms raises an important question: to what extent can one expect that CBR programs will be able to learn indexing terms automatically? CHEF is the only one of the paradigm programs to demonstrate a classic kind of machine learning in its facility for analyzing failures and indexing them appropriately. Credit assignment is manageable because the explanation of a failed case is simple and correct by definition (the recipe can fail only if the simulator says it has failed). The indexing vocabulary is already provided in the failure explanation and concept hierarchy. In this and other instances where an explained case can be analyzed to determine its important features in light of the explanation, indexing terms may be learned automatically.

Where indexing terms are, fundamentally, clusters of features, it is also likely that machine learning algorithms to perform induction will suffice. From a body of positive and negative case examples of a concept, such algorithms can identify clusters of features that are determinative.

Existing induction algorithms, however, do not necessarily pick the best features for solving a problem nor can they invent new terms to apply to cases. For that, one must turn to human experts whose learning abilities are not likely to be captured in AI programs for some time. For instance, HYPO's Dimensions are gleaned from scholarly discussions of trade secrets law. For a program to learn Dimensions automatically, it would need the background and analytical acumen of a legal scholar. This is not to say that inductive or statistical methods could not review a body of suitably represented cases and draw conclusions about which facts or factors are important. But even to create a form for summarizing the facts of legal cases so that statistical analyses can be conducted requires an exercise of sophisticated legal scholarship. It may be that automating some aspects of acquiring expert knowledge of the explanatory importance of facts, along the lines of PROTOS, is the best that AI will do in 'learning' index terms, and it is no small achievement.

4. CBR's Implications for AI and Law

Legal philosophers have long regarded reasoning with cases as fundamental to understanding and applying law (Levi 1949; Llewellyn 1930, 1989, Radin 1933; Burton 1985). Under the doctrine of *stare decisis*, precedents impose constraints on the freedom of a judge to decide a case. Whatever the actual boundaries of those constraints, common law judges and attorneys justify conclusions by citing legal precedents and seek to explain and persuade by comparing current disputes to past cases. As illustrated in Section 1.1, in

diverse aspects of legal practice, common law and civilian attorneys alike reason with past problem-solving episodes. Since cases are so important to law and AI researchers aspire someday to model legal reasoning adequately, case-based methods are essential.

CBR techniques can contribute significantly to modeling legal reasoning and to the design of practical programs to assist in legal practice and education. Some of CBR's potential contributions are realizable in a relatively short term; others are intuitively interesting approaches to tough intellectual problems that will require considerable research time and effort. Sections 4.3 through 4.7 describe these potential contributions in six areas. The discussion draws on the Case-Based Reasoning paradigms and methods, previously described, and points to promising new AI CBR research. The six areas in which CBR can contribute are:

1. assisting rule-based programs to reason about statutory predicates.
2. improving problem-solving performance of rule-based and other reasoning methods.
3. improving explanation in legal expert systems.
4. contributing to the design of intelligent legal data retrieval systems.
5. improving legal document assembly programs.
6. contributing to cognitive modeling of legal reasoning and other fields.

4.1. ILL-STRUCTURED LEGAL PROBLEMS: TWO APPROACHES

In urging a CBR approach, I am mindful that rules are essential tools in almost any legal domain. AI models of law need to come to grips with statutory, regulatory and court-made rules. Although legal expert systems need to reason about authoritative legal rules, to the extent a system attempts to reason deductively with statutory or regulatory provisions represented directly in terms of logical rules, substantial roadblocks arise. There are, at least, five reasons why logic turns out not to be a natural medium for capturing legal rules, statutes and regulations:

(1) Logical ambiguity: Layman Allen has demonstrated that ambiguity caused by legislative failure to specify the scopes of logical connectors such as 'and', 'or', 'if', and 'unless' result in numerous interpretations of even simple statutory texts (Allen & Engholm 1978; Allen & Saxon 1987).

(2) Semantic ambiguity: As is well known, the terms employed in statutes and regulations are not well-defined. Some are deliberately left ambiguous by the legislature. Don Waterman recognized the problems of accommodating ambiguous technical legal terms in a rule-based system (Waterman & Peterson 1980, 1981). The argument has been pressed at length in (Gardner 1987; Berman & Hafner 1986).

(3) Conflicts among legal rules: Not only are legal rules incomplete, they are inconsistent. If a system that performs logical inferences with legal rules has inconsistent rules, it can logically prove any conclusion (Berman & Hafner 1986).

(4) Unstated conditions: Although courts must decide whether a rule applies, the conditions of a rule's application are not stated. Among the unstated conditions are that applying the rule would not be unconstitutional or not contravene legal principles (Berman & Hafner 1986; Allen & Saxon 1987).

(5) Problems of expressiveness versus efficiency: Statutes employ negative conclusions and counterfactual conditionals (e.g., 'would have [become a British Citizen] but for his having died or ceased to be a citizen... [by] renunciation.') If one could specify all of the ways under a statute that one could "fail to show 'not P'" or all the ways a conclusion would have held but for an event, one could handle negative conclusions and counterfactual conditionals. For complex statutes with lots of conditions, this may well be too hard to do manually or require the reasoner to make assumptions about the legal text which are not authoritatively justified. Theorem provers could perform the former automatically, but they are too inefficient (Sergot et al. 1986).

In short, a paramount difficulty of building legal expert systems is the fact that, typically, legal domains lack authoritative strong analytic models that support deducing answers to problems. By 'authoritative', I mean having the imprimatur of a law-making body such as a legislature or court. Since logically representing legal rules and reasoning with them deductively are problematic, many legal problems are inherently ill-structured. Legal rules clearly play a vital role in the process of analyzing and deciding a problem, but that role is not capable of being modeled in terms of logical deduction. Expert system designers need to transform ill-structured legal problems into computationally better structured ones, a transformation for which two basic approaches have been developed.

4.1.1. *Heuristic Rule-Based Approach: 'Deep' Models*

The first technique has been to design non-authoritative bodies of heuristic rules to facilitate reasoning about the legal rules and their application to fact situations. An expert defines heuristic rules to deal with the legal domain's lack of a strong analytic model in the hope that the conclusions deduced from the non-authoritative rules will correspond closely enough to those that human experts obtain with authoritative rules. In constructing expert-level advice-giving systems, production rules have been employed, more or less successfully, to capture legal 'rules of thumb' and other heuristic legal knowledge (Susskind 1987, Capper & Susskind 1988). In a classic system whose primary function was to advise on settlements, Waterman included some heuristic rules, derived from experts, to deal with statutory terms that were ill-defined or indefinite, such as 'reasonable and proper', 'foreseeable,' 'responsible for use of product,' and 'incidental sale' (Waterman & Peterson 1981, p. 4). The expert would provide successively refined rules, in effect, defining a term, or, at least, summarizing its past uses (Waterman & Peterson 1981, p. 26).

The creation of 'deep model' legal expert systems takes even farther this heuristic approach to dealing with a legal domain's lack of an authoritative strong analytic model. An expert creates a comprehensive theory about a legal domain, the so-called deep model. The rules of the theory are then implemented as production rules in an expert system. The system's rules are intended to capture the expert's heuristic knowledge about the domain and about how the domain's authoritative legal rules (and other sources of authority such as leading precedents) will be applied to problems (Smith & Deedman 1987).

The deep model approach relies on the ingenuity of the legal expert in transforming ill-structured problems into well-structured ones. Essentially, the expert creates a strong analytic model where none may have existed before, but a model whose authoritative source is only the expert. The heuristic rules explicate the important statutory concepts and introduce new concepts which, in the expert's view, are helpful in analyzing problems and predicting outcomes. Through the introduction of these new terms and use of various knowledge representation techniques, the expert, in effect, transforms ill-structured problems into ones that are solvable deductively.

As the problem domain is pounded and bent into a well-structured mold, some corners need to be rounded. Certainty factors may be one of the knowledge representation techniques employed for this purpose. Certainty factors purport to enable an expert to opine that a rule applies with a specified level of confidence rather than with certainty. This facilitates combining confidence levels of various pieces of a complicated analysis (i.e., numbers are easy to combine) and provides criteria for ranking alternatives. The facility of certainly factors has a cost, however, since their sources and meanings are rarely straight forward, they are difficult to explain, and symbolic information is lost as they are combined.

Whether a deep model approach works depends on how one defines 'works'. The program's predictive abilities are, in theory, evaluable. If the expert is clever, the model may work in the sense that it does a good job of analyzing problems and predicting correct outcomes. Whether a deep model system works in the sense that it can explain its results or convince users of their reliability is questionable, as argued below. The deep model knowledge is not represented in a form conducive to formulating good explanations.

4.1.2. *Case-Based Approach*

The second approach to transforming ill-structured into better structured legal problems involves reasoning with cases. If one cannot deduce from authoritative sources whether a legal rule or term is satisfied, one may justify assertions that it is satisfied or not by comparing the problem to cases where a court authoritatively determined whether the rule or term was satisfied. Waterman anticipated this approach, as well. In order to deal with the problems of semantic ambiguity of technical legal terms in a rule-based system, he suggested two techniques that employed cases: (1) Let the system decide whether the statute term is satisfied by comparing the problem with case instances and picking the best case. (2) Let the user decide whether the statute term is satisfied but make it an informed decision by presenting case instances of how the predicate was decided in the past (Waterman & Peterson 1981, p. 26).

The case-based approach attempts to make up for the lack of an authoritative strong model by modeling weaker methods for drawing and supporting legal conclusions. A symbolic case-based approach imposes enough structure on a domain to support case comparisons and case-based inferences. These symbolic comparison, inference and justification methods correspond to (at least some of) the methods employed in actual

legal arguments. As discussed below, this enables such programs to generate explanations that look like real legal arguments.[7]

In imposing some structure on problems, each of the paradigm case-based programs discussed above assumes the existence of a partial domain theory for analyzing problems by comparing cases and adopts a complementary case representation language to enabale the comparisons. When cases and problems are represented in the appropriate language, the program can compare and interpret them in terms of the partial theory. The programs differ in the extent to which the theories are expressly or only implicitly represented and in their degrees of completeness. We have already seen some of the techniques employed to structure case comparisons to facilitate problem solving: a causal inference network as in CASEY, PROTOS' explanation-related heuristic weights, HYPO's Dimensions (and CABARET's application of them to legal predicates, discussed below), GREBE's structure-mapping explanations, and CHEF's and PERSUADER's failure-driven repair and modification techniques.

Unlike a heuristic rule-based approach, typically, adopting a case-based approach falls far short of transforming legal problems into well-structured ones for which 'right' answers can be deduced. Any realistic matching process for legal cases probably yields a set of candidate 'best matched' cases with conflicting outcomes. Frequently, there are a number of ways of defining the 'best case' and alternative measures of relevance. But the problems are better structured in so far as comparing the problem to relevant cases helps to inform a judgement decision. Ideally, an adversarial case-based process leads a decision-maker to work through the competing analogies, focusing on important similarities and differences among the candidates, considering alternative rationales, assessing the strengths of competing arguments and testing alternative conclusions.

While I recognize that no case-based program realizes this ideal, the programs listed have made initial but substantial strides. Although there are significant open issues, I believe that work on Case-Based Reasoning is more likely to address and solve the interesting questions about law and legal reasoning than a heuristic rule-based approach. Whatever the strengths of other methods, case comparisons have important, complementary roles to play in assisting programs to reason about legal rules and to improve explanation and performance. They also can help systems designers and researchers address tasks for which production rule systems are not likely to be sufficient such as information retrieval, document assembly, and designing accurate cognitive models of legal reasoning.

Of the two approaches to dealing with ill-structured legal problems, I concede that the heuristic rule-based approach is far easier to implement and to get 'up and running' by virtue of its simple inference mechanism, *modus ponens*. (Of course, coming up with rules for a deep model may be a scholarly legal task of no small magnitude. Refining the rules to insure consistency and effectiveness is also time consuming.) By contrast, even the inference methods in CBR must be invented. Part of the challenge of case-based approaches

[7] Conceivably, a CBR legal program might employ statistical methods for analyzing factors and summarizing case trends with good predictive results but an inability to generate traditional symbolic legal arguments (see e.g. Henderson, Jr. & Eisenberg 1990; Eisenberg & Henderson, Jr. 1992).

has been to discover inference mechanisms by examining how experts compare cases to draw conclusions. To date, production rule systems also have been the most successful AI technology for fielding applications in the legal area. Recent reports of ruled-based legal expert systems are very encouraging (see, for example (Greenleaf et al. 1991)). Every AI/Law researcher who has stumbled across an in flight magazine advertising a production rule system for personal income taxes has felt heartened. Successful deployment of rule-based expert systems in law bodes well for the field of AI and Law. Like other areas of AI, we need to demonstrate successful applications to encourage financial support from the public and private sectors.[8]

Promising as they are, heuristic rule-based approaches are not a panacea, and it is worth cataloging reasons why AI and Law should not put all its hopes into production rule systems. As already noted, they deal with the problems of semantic ambiguity and direct logical representation of legal rules by employing heuristic rules to predict the net legal results. Heuristic rule-based systems, however, have weaknesses. They do not adequately explain their advice, deal with situations where there are conflicting heuristic rules or competing sets of rules nor do they present reasons for *not* believing their conclusions. As a result of these weaknesses, rule-based system users cannot adequately determine whether the advice is reasonable or whether they reasonably can rely on the advice. They risk liability if they do rely on it, a situation that poses problems for them and for the makers of the legal expert systems. Ultimately, I fear these problems imperil the chances of successfully fielding such systems. At the very least, these weaknesses imply that rule-based approaches present research issues of no less difficulty than case-based approaches. With respect to a number of these problems, CBR offers solutions or, at least, directions.

4.2. PROBLEMS WITH HEURISTIC RULE-BASED LEGAL EXPERT SYSTEMS

4.2.1. *Explanation Deficiencies*

Legal expert systems that employ heuristic, rule-based approaches, including deep models, present problems. Their explanation and justification techniques are limited. Attorneys employ cases to interpret, apply, circumvent, explain and justify the application of legal rules to specific facts. Collapsing this information into the heuristic rules

[8] While it is true that CBR is largely still a field of academic research, especially in the legal domain, CBR programs have begun to be fielded in commercial settings such as 'Help Desk' applications. The programs enable customer service personnel to draw on a database of cases in determining how to handle clients' problems. CASCADE, a Help Desk program designed to assist engineers respond to reports of operating systems crashes, is reported in (Simoudis & Miller, 1991).

A case-based design program, CLAVIER, has been assisting designers to configure lists of parts to be cured in an autoclave oven. Reportedly, the case database has grown to 250 cases, and the system has been used daily at Lockheed (Mark 1989).

Recently, CBR system-building shells have been fielded in commercial settings by Inference Corporation, Cognitive Systems, Inc., and Esteem Corporation. Although it is too early to assess how successful these tools may be, their commercial deployment shows that CBR is in the process of becoming one of AI's basic methodological techniques.

of a deep model expert system is expedient but fails to account for justification and explanation. Attorneys cannot justify conclusions solely by citing the opinion of an expert, even the author of a deep model. For justifying and explaining conclusions, they need other authorities such as cases. In addition, such systems do not deal adequately with alternative analyses. There are, after all, other experts with differing views. As described below, cases can play an important role in dealing with alternatives.

In classic works, Swartout and Clancey criticized the inability of production rule systems adequately to explain their advice. Swartout complained that production rule systems were unable to justify their actions or reasoning paths because the principles of the domain were not represented:

> [C]urrent explanation methodologies are limited to describing what they did (or the reasoning path they followed to reach a conclusion), but they are incapable of justifying their actions (or reasoning paths). By justifictions, we mean explanations that tell why an expert system's actions are reasonable in terms of principles of the domain - the reasoning behind the system (Swartout 1983, p. 287).

Similarly, Clancey complained that although MYCIN could generate an explanation consisting of a trace of the program's backchaining through its rules, the explanations were deficient. The rule traces often provided too much detail without focussing on important aspects of the reasoning path. They also failed to relate the rules to familiar patterns of reasoning one has encountered previously and premises that one readily accepts (Clancey 1983, pp. 226–7), as good explanations would.

Clancey and Swartout both sought to remedy these problems of explanation in expert systems by explicitly representing the 'missing' knowledge.[9] Other researchers have sought to enable expert systems to elaborate previous explanations and respond to follow-up questions (Moore & Swartout 1989, 1990a, b; McKeown et al. 1985). In solving the problem of explanation, however, AI needs to focus on domain specific techniques for explaining advice and persuading the hearer to take it. In the legal domain, Clancey's 'familiar patterns of reasoning' to which explanations need to relate include explaining with cases as examples, drawing illustrative comparisons to other cases and persuading by showing how plausible counterarguments fail.

Surprisingly, given the importance of examples as explanatory tools in human discourse, there has not been a great deal of work in harnessing examples to improve rule-based explanations. In the area of intelligent tutoring, where examples are central to explaining tasks and concepts to be taught, researchers have made some progress on employing examples as part of the interactions between program and user. See, for example (Lesgold et al. 1987; Collins & Stevens 1982; Aleven & Ashley 1992; Ashley & Aleven 1992). Indeed, the fact that a production rule system does not represent

[9] Clancey offered a number of solutions, including relating heuristic rules to: (1) causal medical 'principles or generalizations' that can be used to justify the rule (Clancey 1983, p. 232), and (2) strategic principles that justify the rule's application in the diagnostic inquiry (Clancey 1983, pp. 238, 240). These strategic 'principles for reasoning about evidence' include principles like, "Common (frequent) causes of a disorder should be considered first' or 'If nothing has been observed, consider situations that have no visible manifestations,' (Clancey 1983, pp. 238, 240).

knowledge in a form adequate for generating tutorial explanations led Clancey to develop his critique. Some classic work on example generation in a non-tutoring context is (Rissland 1981, Rissland & Soloway 1980).

In fairness, production rule systems in law have been developed in which rules index sample cases and may even provide hypertext interfaces allowing users to browse through court opinions (see, for example (Greenleaf et al. 1991)). Potentially, this is an important development. My goal for such systems, however, would be to enable systems to guide users to cases relevant in the context of the problem at hand. Statutes may have a lot of case annotations, most of which may not be relevant to the user's problem. To the extent that the program has a model of case relevance, it could guide the user to the best case examples and counterexamples of its advice. This would be a robust example of CBR in the service of explanation.

4.2.2. *Conflicting Rules and Alternative Analyses*

In general, rule-based legal expert systems also do not deal with conflicts among rules or competing sets of rules and there has not been enough emphasis on providing alternative analyses. Despite the importance in legal reasoning of analyzing problems from multiple viewpoints, arguing multiple sides of a legal question and symbolically comparing alternative answers, legal expert systems tend to seek only one answer. Although I have already mentioned this issue in connection with the problem of accommodating conflicting legal rules, it remains a tough problem for heuristic rule-based systems. I do not maintain that it is impossible for rule-based expert systems to present alternative analyses of a problem. Indeed, MYCIN did cover alternatives; it performed a complete, backward-chaining search of the problem space and ranked the alternative therapies in terms of certainty factors (Shortliffe 1984). Generating alternatives and dealing with competing rule sets has not, however, been a focus of legal expert systems development.

This is a pity because, in general, legal experts disagree about which rules of thumb are approprite for summarizing any given area of law. Sometimes the disagreements are minor but, frequently, major theories of a statute or rule vie for acceptance. Though they may disagree with an opposing theory, legal experts often must concede that it has some merit. At least some legal authorities agree with it. In a given situation it may lead to a different conclusion from the one they would have reached, but a reasonable (or, at least, 'reasoned') conclusion nonetheless.

One may object that a legal expert's ability to assess confidence in a conclusion in light of all the alternative reasonable conclusions is reflected in his or her formulations of the heuristic legal rules. Two problems remain, however: whether the expert system can be as sensitive to context as the human expert in assessing confidence in a conclusion in light of alternatives and whether the system can adequately explain and justify its preference for one conclusion over another.

The question of context sensitivity is an empirical one that should be addressed in

evaluating Deep Model expert systems. Where an alternative theory is not even represented in the program's rule base, the program will be insensitive to contextual cues in a problem that warranted an alternative approach.

Even if alternatives are represented, there is still a problem of whether the system can justify and explain its preference for one conclusion over another. I am sceptical of knowledge representation techniques, such as certainty factors, which purport to summarize the expert's confidence in an analysis and provide criteria for ranking alternatives. What do the resulting numbers really mean? Can they be explained? They convert a question that humans resolve symbolically into a number of dubious meaning and subvert the possibility that the system can explain and justify its reasoning symbolically (see, for example, the critique in (Henrion et al. 1991, p. 81).

Some designers of rule-based legal expert systems have addressed the problem of presenting alternative analyses. In *Models of Legal Decisionmaking*, Waterman and Peterson provided manually drawn figures depicting traces of rules that fired in analyzing two versions of a product liability problem and a final figure illustrating the effect on the analysis of assuming an alternate definition of strict liability. Not only do the figures present overviews of the rules that fired and their relationships to one another, but they demonstrate the effects of changes in the formulation of a rule (Waterman & Peterson 1981, pp. 18–20). Figure 18 shows how a change in the definition of strict liability changes a former result. Similarly, Anne Gardner's graphic presentation of her program's analyses of offer and acceptance problems illustrated the effects of adopting different rules. Under Williston's rule an acceptance that varies the terms of the offer is ineffective. Under the rule of UCC 207 it is an acceptance plus a proposal to modify the contract (Gardner 1987, p. 173).

Designers of legal expert systems should take up the goal of building systems that generate such diagrams automatically. Significantly, the creation and editing of graphs like Figure 18 is an intelligent activity closely related to devising effective explanations. The graphics designer exercises editorial decisions in determining which parts of the rule trace to emphasize. As the authors report, '[t]he figure does not trace the inference process all the way to the facts of the case; it omits some rules that lead to conclusions listed at the left margin, as well as other rules that provided premises for those shown in the figure (Waterman & Peterson 1981, footnote 24)'. Like Clancey and Swartout, the graphics designer seeks a basis for determining which rules and details to leave out to better focus an explanation, here presented graphically.

I think these attempts are promising, not only because they recognize the importance of dealing with conflicting rules and alternative analyses, but because they also suggest an important role that cases may play in enabling a legal expert system to deal with alternatives. If the program, at least, could cite cases as counterexamples to its conclusions in a given context, it would greatly increase a user's ability to assess the reliability of the system's advice, as described more fully below. One could imagine adding to Waterman's figure, graphic pointers to the cases that support or detract from the intermediate conclusions or to cases where the alternative analyses were considered and a decision among them made.

4.2.3. *Is Reliance Reasonable?*

The deficiencies in representation of statutory sources, explanation and presentation of alternatives in heuristic rule-based legal expert systems raise a serious problem: can users reasonably rely on their advice? Although there are applications where these infirmities may not matter or can be finessed, in general, they do matter. Ultimately the question of legal liability in connection with expert systems will turn on whether a user may reasonably rely on the advice of the expert system without independently verifying its correctness. Whether reliance is reasonable will turn on a number of things, including the feasibility of independent verification, the evaluation of the system, the nature of the foreseeable injuries, the user's sophistication, the manufacturer's representations, disclaimers, warranties, and warnings, and, significantly, the system's ability to explain and justify its advice (Nimmer & Krauthaus 1986).

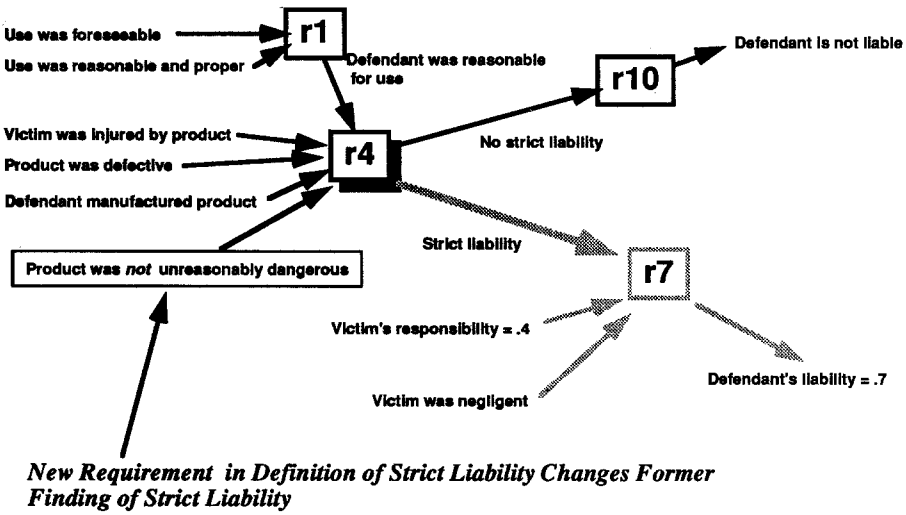

Fig. 18. Manually drawn graphic representation of alternative rule-based analyses.

If rule-based systems do not provide adequate explanations of their advice, it does not seem likely that they will be deployed very widely in legal domains where the possibility of liability for giving wrong advice is an acute problem. If the system can explain its advice only by showing a trace of rules and inferences that lead to its conclusion, but cannot focus the user on the crucial steps or important assumptions or justify the appropriateness of applying a crucial rule, a user may not be reasonable in relying on its advice.

If the system furthermore cannot apprise the user of alternative answers based on different assumptions about the 'right' rule or the 'right' interpretation of the concept, that would compound the issue of liability even more. To the extent that a legal expert (or a knowledge engineer) assumes the responsibility of selecting one interpretation as 'the' interpretation of the law, presumably he or she assumes responsibility for any negative

results of a user's reliance on a decision maker that does not compare alternative interpretations. The user cannot reasonably rely on such advice, and that would seem to expose the user, the system designer, and even the expert to the risk of liability. This would be especially true if a crucial rule turns out to be controversial. Suppose experts disagree as to how the rule should be formulated or the rule employs a deliberately ambigous statutory term. If the system designer has arbitrarily selected one formulation of the rule or one interpretation of a concept for which there is a range of interpretations, that would not justify reasonable reliance on the system. Indeed, it is a little surprising that a legal expert would agree to have his or her approach to an area of law ensconced in a program's rule base where it will be mechanically applied to reach but one conclusion without a hint that an alternative analysis may also be valid.

One may object that experts who write legal treatises often adopt 'the' interpretation of a legal rule, but a book is different from an expert system. With a book, the application of the information to a specific set of circumstances is left to the reader. Of all a book's information, the reader judges for himself which applies to a specific problem and what its impact is. An expert system, on the other hand, purports to tailor advice to the user's specific problem and circumstances. The system selects and applies the knowledge, and, in fact, forecloses the user from doing so. Foreclosing alternative interpretations involves a much more onerous responsibility on the part of the author of an expert system's rules.

Finally, if the system's explanations fail to be of the sort that human experts provide, for instance because they do not provide relevant concrete examples or argue alternatives or demonstrate consequences with hypothetical cases, reliance is even less reasonable.

In the absence of an adequate explanatory facility, reasonable reliance may turn on the extent of evaluation of the expert system, hardly a rosy prospect. An expert system may yield results comparable to human experts' for a set of test problems, but how difficult would it be for a plaintiff's attorney to show that the system does not have even the most basic common sense knowledge of the concepts with which it deals, that it does not have a clear notion of the kinds of problems for which it is not qualified, that it cannot deal adequately with noisy, missing, inconsistent data, and that it does not consider alternatives or test assumptions?

In other domains such as medicine, the problems of explanation, context sensitivity and evaluation posed by decision-making systems, including production rule systems, are profound (see Miller et al. 1985). Given the legal profession's genius for demonstrating that rules and logic are not as certain as they seem, it is not likely that the law will be more amenable to solving these hard research questions with heuristic rules. Adopting a rule-based approach may be a good way of scoping out a legal domain and designing a prototype that will yield 'quick' results, but it is not a 'quick fix' to designing practical legal expert systems. When it comes to solving such vital research questions, things are, at least, no worse for the Case-Based Reasoning community. CBR researchers have not solved the above-described problems either, but they are, at least, asking the right questions, making progress toward solutions to some of these same problems, and even providing solutions that can be integrated with rule-based systems. Morever, CBR approaches are more

amenable to the development of expert assistants, programs that assist practitioners to make informed decisions by accessing relevant information. Now I will present some of these positive arguments in favor of pursuing case-based approaches to law.

4.3. INTEGRATING CBR INTO HEURISTIC RULE-BASED SYSTEMS

Systems that deal with legal domains do not have to be purely rule-based or case-based but can integrate the two types of reasoning in order to deal with ill-defined predicates, handle exceptions to rules, and even improve the performance of rule-based systems.

4.3.1. *Representing Legal Predicates*

In order to take up either of Waterman's suggestions about employing cases to deal with ambiguous rules or terms, a system needs a model of case comparison. It needs a mechanism for comparing a problem and cases in order to justify a conclusion that the problem should be treated like a past case in so far as applying a legal term is concerned. The system could then select and present case instances either to guide the user's decision or make its own decision based on the (possibly competing) best cases.

The CABARET program moves farthest in the direction of meeting Waterman's sug gestions (Rissland & Skalak 1991; Rissland 1990; Rissland & Skalak 1989). In CABARET, ambiguous predicates from statutes are annotated with positive and negative case examples. The program can compare a problem with the cases and use them to justify an assertion that the predicate is satisfied. It has been implemented for a provision of the United States Internal Revenue Code dealing with the home office tax deduction, IRC Sec. 280 A (c) (1). This provision deals with whether a taxpayer can treat an office within his or her home as a deductible business expense. Inputs to the program are a fact situation and specification of viewpoint (taxpayer or IRS). The program outputs an argument in favor of the position that the home office is or is not deductible. The arguments cite cases justifying an assertion that various statutory predicates are satisfied or not (Rissland & Skalak 1991, pp. 839–887).

CABARET comprises rule-based and case-based modules and an agenda mechanism. The rules are derived from the relevant provisions of the tax code. An overall rule incorporates the statutes' frequently litigated predicates, each of which also has associated heuristic rules. The rule-based module chains forward from facts to confirmed goals or backward from desired goals to confirming facts (Rissland & Skalak 1991).

The case-based module is based on HYPO. As in HYPO, similarities and differences are defined in terms of shared and unshared Dimensions. CABARET's Dimensions are collections of facts that strengthen or weaken a claim of the IRS or defense of the taxpayer. Each statutory predicate is associated with a set of Dimensions that strengthen or weaken an argument the predicate is satisfied. There is a different set of Dimensions for each predicate. Each case is indexed by the Demensions that apply to it and records the general outcome of the case and the specific outcome regarding each statutory predicate decided in the case (Rissland & Skalak 1991, p. 865).

Integrating case- and rule-based reasoning presents a practical problem of control: deciding when to use cases and when to use rules. The designers of CABARET have identified control heuristics employed by the agenda mechanism to integrate the case- and rule-based modes of reasoning. The control heuristics include rules like the following:

Switch on failure: If one mode of reasoning fails then switch to the other mode.
Sanity check: Test conclusion reached by one mode with the other mode.
Open Texture: Use CBR on deliberately open textured terms.
Match statutory predicates: Find a case that has failed and succeeded on the same statutory predicates.
RBR Near-miss: If all of a rule's antecedents are established but one, use CBR to broaden the application of the rule with respect to the missing antecedent. There are a number of ways to do that:
Broaden Missing Antecedent: Use CBR to establish the rule's missing antecedent.
Broaden-01: Use CBR to show there are cases where the conclusion of the rule is true but the antecedent is not established (Rissland & Skalak 1991, p. 857).

For instance, in processing a taxpayer's (Weisman's) dispute with the IRS, the program fails in its rule-based attempt to conclude that the home office is the taxpayer's 'principal place of business'. The rule defining this precedent fails to fire because the case contains insufficient information for the program to conclude one of the rule's antecedents: whether the taxpayer discharged his primary responsibilities in that office. CABARET invokes the 'RBR Near-miss' and 'Broaden Missing Antecedents' heuristics and attempts to find a case where 'primary responsibility in home office' was satisfied. It succeeds in locating a past case, the *Drucker* case, and draws an analogy in terms of shared factors: In both cases 'the home office was necessary to perform the taxpayer's duties' and 'there was evidence as to the frequency of usage of the home office by the taxpayer' (Rissland & Skalak 1991, pp. 866–876).

Using an approach like that in CABARET, system designers can more justifiably place some of the burden on the user for deciding if a predicate is satisfied. The system could present the user with a set of alternative positive and negative case examples of an ambiguous predicate. The output can be tailored to the context of the user's problem by focussing on the cases that share factors (i.e., Dimensions) with the problem and highlighting the relevant similarities and differences. The system could organize the alternatives into arguments pro and con; conceivably it might be able to evaluate the competing arguments and make a recommendation. The user's choice would be informed by the specific alternatives and competing examples.

Production rule systems that merely annotate rules with cases could be improved by integrating some adversarial case-based ability. As in CABARET, a more complete integration of cases and rules goes beyond simply annotating legal expert system rules with cases. While better than nothing, such annotations enable only a superficial kind of case reference; activated rules lead the user to case summaries which a user can read (perhaps in a hypertext display). A more complete integration of case-based reasoning, however, involves designing a system that can focus on important features of the problem which in the system's analysis justified its conclusion, provide case examples and counterexamples, and enable the user to test the importance of those features. Dimensions in CABARET, for instance, enable a program to order retrieved cases in terms of whether

they are stronger or weaker examples of a rule predicate. Using relevance criteria developed in HYPO, a program can select the examples most relevant to the user's problem and tailor the descriptions of a recalled case to focus on the important differences between the problem and a case. Without the ability to focus and test, it is not likely that a user could assess the reasonableness of advice by perusing case summaries whose only connection to a problem are that they, too, involved a legal rule.

Fully integrating cases into a rule-based heuristic legal expert systems has not yet been accomplished on a large scale. CABARET has only a few simple heuristic legal rules. PROLEXS integrates cases and a rule-based component into an expert system advising on landlord/tenant law (Walker et al, 1991). The cases, however, tend to be simple, single issue examples. Although GREBE invokes cases to supplement rule-based analyses and vice versa, it lacks criteria for deciding when to pursue case-based or rule-based analyses, always attempting all avenues. See also (Vossos et al, 1991) which describes an attempt to employ concepts drawn from legal rules to improve case relevance assessment.

4.3.2. *Representing Exceptions to Rules*

A system that supports reasoning with rules and cases can test results of one mode of reasoning against the other (sanity checks in CABARET's parlance). A rule-based assertion that certain problem features are determinative could be cross-checked against decided cases.

Rule-based expectations that particular features determine the outcome of a problem may or may not be confirmed by past precedents. In law no one expects cases to be perfectly consistent with rules. Interpretations of rules are highly context dependent and change over time. There is, moreover, no reason why the two modes should always yield consistent results since they employ different methods for representing importance. In a heuristic rule-based expert system, features are important because some rule makes them so; the features appear in the antecedents of rules that have fired and lead to a conclusion. Knowledge engineers seek to formulate rules and define concepts that correctly reflect the importance of the features. By contrast, CBR programs represent the importance of features in ways other than by definition. In a case-based program, problem features are important to the extent they make the problem similar to relevant case examples and counterexamples where relevance, as we have seen, can be defined in a variety of ways.

A past case may be a counterexample to a rule in the context of the problem (i.e., a case where the antecedent features were present but the rule's outcome did not follow). If a rule-based analysis leads to a conclusion for which there is a close counterexample, the contrary case qualifies the reliability of the rule-based analysis and may be the basis of a reasonable counterargument. In purely heuristic rule-based systems, this important symbolic information would be lost or subsumed into a reduced certainty factor.

The ANAPRON program models reasoning with counterexamples in somewhat this way. Dealing with a pronunciation domain like that of MBRTALK, ANAPRON has a

set of rules for pronouncing names, but applies cases to determine whether it should follow the rules (Golding & Rosenbloom 1991). The consequents of the program's rules are called operators; they enable the pronunciation of a name to be predicted. A case is stored as a negative example of those rules it violates and as a positive examples of rules it confirms (Golding & Rosenbloom, 1991, p. 23). A case violates a rule if the rule says that an operator should apply but the case record says that a different operator applied.

ANAPRON applies rules, but employs cases to condition the decision whether to follow a rule that otherwise would apply. In selecting an operator to apply to a new problem the program first determines, in a forward chaining manner, which rules' antecedents are satisfied. Suppose Rule A's antecedents are satisfied and that Operator A is the consequent. Before actually applying Operator A, the program applies a case-based test. It searches through any negative case examples of Rule A for a compelling analogy. If it finds one, it applies the operator suggested by the compelling analogy. Otherwise it applies the rule's operator. If, for instance, Case X is an exception to Rule A (i.e., Rule A should apply but Operator X actually was applied), and Case X is a compelling analogy, then the program applies Operator X, not Operator A (Golding & Rosenbloom 1991, p. 23).

'Compellingness' of an analogy in ANAPRON has two components: one measures similarity of features and the other deals with the analogy's accuracy as a predictor. In effect, the program generates a rule based on the counterexample. The rule's antecedent includes all of the features shared by the case and the problem; the consequent is the operator applied in the case. Similarity is measured simply as the number of antecedents (i.e., shared features). Accuracy is measured by trying the rule out on existing cases in the case library, determining how well it succeeds in predicting the results, and comparing how much better the success rate is than classifying cases by pure chance. If the similarity or accuracy values are high enough relative to threshold values, the analogy is judged to be compelling (Golding & Rosenbloom 1991, p. 23).

ANAPRON's techniques may be adaptable to legal tasks. Indexing rules with positive and negative examples and comparing a problem to the examples before deciding whether the rule applies certainly are consistent with legal practice. Any attorney who consults a statute book or restatement of the law of a particular domain consults the rules' case annotations as well as attempts to determine whether a rule applies by its terms. On the other hand, ANAPRON's definition of 'compelling' analogies is not symbolic but numerical; it is unclear what the thresholds would mean in a legal context or how to compute them.

Conceivably, a heuristic rule-based analysis of a legal problem could be used automatically to constuct a data base query for relevant counterexamples. In a complex rule-based analysis, counterexamples are interesting to the extent that they are either very similar to the problem situation in a global sense or deal with crucial parts of the analysis. Checking each rule that fires for counterexamples, as ANAPRON does, probably will not adequately focus on important counterexamples. Instead, one needs some relatively

context-sensitive representation of the crucial parts of a rule-based analysis. Determining crucialness in a rule-based analysis is a question of credit assessment which we have encountered before. CHEF performs exactly this kind of credit assignment in determining the causes of cooking failures which it uses to repair and index recipes. The credit assignment problem is also at the core of the problem of rule-based explanation: how to focus the user on crucial facts and reasons embedded in a backtrace of activated rules.

The search for counterexamples involving crucial facts may present a more tractible credit assignment problem, however. I assume that legal experts from whom heuristic rules are derived would also be willing to opine as to which of a rule's antecedents are crucial to the rule's conclusion. By 'crucial', I mean, 'likely in the expert's experience to be controversial or controverted or to lead to counterexamples.' Expert opinions on 'crucialness' of facts can be acquired much as, in PROTOS, an expert's opinions about the importance of a fact to an explanation are acquired and represented. Such information may be used to constrain searches for counterexamples by focusing only on counterexamples dealing with crucial antecedents or their associated features. If a crucial antecedent does not lead to a counterexample (a relatively inexpensive test to ascertain), the system would move on to another crucial antecedent. Although expert opinions on crucialness may look like certainty factors, they have a much more limited and defensible use: to select among possible counterexamples those to call attention to. Rather than subsuming contrary information into numerical certainty factors, expert opinions as to crucialness could lead to counterexamples that symbolically represent contrary information.

4.3.3. *Improving Performance Through CBR*

Case-based methods may complement rule-based ones in a different sense; they offer a way to improve the performance of rule-based and other technologies. If rule-based approaches are, indeed, adequate, then there may be utility in compiling such solutions for use as patterns in solving new problems. One approach has already been discussed in CASEY: compile explanations generated with rules or other reasoning methods and reuse them. CASEY demonstrated dramatic improvements in efficiency by reusing explanations previously generated by the HEART FAILURE program. The approach presupposes that one could build a detailed causal model of the domain sufficient for a program to reason about partially matched cases to determine if the match is justified.

ANAPRON has shown that simply annotating rules with exceptions increases the accuracy of a rule-based system. In tests run for the domain of name pronunciation, Golding has shown that ANAPRON's accuracy compares favorably with commercially available pronunciation programs and that recording counterexamples to rules enhances the performance of the rule-based component (Golding & Rosenbloom 1991, p. 25f). His experiments also showed that by combining a rule-based and a case-based component, the system achieved higher accuracy than either the rule-based or case-based component alone could achieve (Golding & Rosenbloom 1991, p. 26).

4.4. IMPROVING EXPLANATION WITH CBR

Another benefit of pursuing case-based AI and law research is to improve explanation. Although CBR offers no 'quick fixes' to the complex issue of explanation, it does yield some insights into the problem of explaining in domains that lack strong theories. Case-based explanations may also enable users to assess the reliability of system-provided advice. As I discuss these benefits more fully, I will point to CBR research that has, at least, begun to probe ways in which systems can model explanatory uses of cases.

4.4.1. *Case-Based Explanations Compensate for Weak Models*

As we have seen, designing a rule-based heuristic or deep model is one approach to compensating for the absence in legal domains of authoritative strong analytic models that support deducing answers to problems. As Clancey discovered in MYCIN, however, representing expert knowledge in heuristic rules does not adequately support explanation. Though easy to generate as a by-product of the system's reasoning, a proof of an answer's correctness in terms of a model designed by a legal expert is not an adequate explanation. The reasons are stated above in Section 4.2.1.

A symbolic CBR program in law compensates for the lack of an authoritative strong domain model by modeling at least some of the case-based comparison, inference, and justification methods which legal advocates employ. Since the programs can answer the question, 'Why does the similarity (or difference) between two cases justify the same (or different) treatment?' in legally meaningful ways, they should be able to generate explanations that correspond more closely to actual legal explanations. HYPO and CABARET compare cases in terms of factors that strengthen or weaken a side's legal argument; they explain the significance of similarities and differences in terms of information represented in Dimensions and the case examples they index. GREBE points to the presence or absence in the problem of features that were significant in light of a court's rationale for a previous decision. Although not involving legal domains, some of the other CBR paradigm programs' case-based inferences correspond to authentic legal inference methods. PERSUADER relates conflict-causing features to conflicting goals and past resolutions, a natural medium for comparing contractual agreements. To explain the significance of similarities and differences, CASEY and CHEF apply relatively strong analytic models (the causal inference network and rule-based simulator, respectively) which might well be appropriate for tax problems and other legal domains with strong models. PROTOS's acquisition of expert classification explanations and measures of prototypicality might facilitate regulatory classification in administrative law contexts.

4.4.2. *Case Examples Facilitate User Reliance*

In ordinary discourse, case-based explanations, involving comparisons not necessarily much more complex than those referred to above, provide important benefits to the listener. They anchor abstract explanations in specific circumstances, focus a listener on

salient features of the problem and suggest how to test the limits of the advice. In short, they help the listener assess the reliability of the advice. CBR has, at least, begun to accommodate these features of expert/listener discourse into expert system/user interactions.

Case examples anchor an explanation in specific circumstances, enabling the listener to draw connections between abstract explanatory concepts whose definitions he may not know and concrete facts with which he may be more familiar.

The case examples provide focus and a basis for dialogue between listener and expert. Comparing the problem to case examples focuses the listener on salient problem features. When an expert compares a problem and examples he emphasizes shared features which he regards as particularly important and explains why they are salient in the context of his view of the problem. He may also contrast the problem with near misses or counterexamples to point out crucial differences that would justify different treatment.

Employing examples to focus a listener on important features helps the listener to assess the reasonableness of advice. Even if the listener cannot formulate a question in the expert's technical terms, in the specific context of a comparison among cases, he can ask why a particular factual similarity or difference does or does not matter. The expert may cite a counterexample close to the facts of the problem as a nontechnical way to caution the listener that advice lacks certainty. Alternatively, the expert can bolster a listener's confidence if there are no counterexamples or if the expert distinguishes any counterexample effectively. The expert may probe the listener to test whether an important fact really exists. On closer inspection, the listener may decide that the problem is not really like the example or has other features that distinguish it. The listener may draw his own conclusions from the fact that the expert regards two cases as similar. The analogy may suggest that the expert is misguidedly solving a problem different from the one the listener means or that the analogy's significance is subtle enough to have escaped the listener. In either case, the sooner the expert and the listener start talking a common language the better.

CBR programs have only begun to support this sophisticated kind of expert/listener interaction, but they are moving in that direction. PROTOS and CABARET, for instance, employ cases to elucidate for the user the meanings of technical classifications. HYPO, CABARET, GREBE, and PROTOS all make use of counterexamples. CASEY can focus on differences and confirm or deny their significance in light of its causal model. GREBE and CABARET can attempt to eliminate a difference by building an alternative bridge between facts and a conclusion based on an analogy to a different case. PERSUADER solicits user feedback on whether it has solved the right problem. PROTOS solicits clarifications from an expert to distinguish cases the system regards as nearly identical. HYPO edits case descriptions to focus the user on contextually salient features. Its hypothetical problem modifications are intended to probe the user for additional problem facts. In HYPO, pushing hypotheticals to extreme values along Dimensions or by adding Dimensions suggests how to probe the outside limits of a conclusion. The tutorial program, CATO, poses dialectical examples, collections of problems and cases intended to teach students the significance of similarities, differences, examples and counterexamples.

CBR has also just begun to model other aspects of expert ability to explain with examples. Human experts display a tremendous dexterity in fashioning explanations. Experts can switch explanation modes with ease, sometimes explaining with a case example, sometimes with an analytic proof, sometimes employing a case example in service of a proof or a proof in service of explaining an example. CABARET and GREBE both illustrate techniques for integrating proofs and cases in explanations. PERSUADER and its predecessor, MEDIATOR, demonstrate how a system can use the same case examples in different ways (Simpson 1985; Kolodner et al. 1985).

4.4.3. *Improving Case-Based Explanations: Rationales, Principles and Theories*

Further improvements in explanation, however, depend on improving ways to represent and reason with cases, in particular, dealing with rationales of cases, principles, and theories.

When a human explainer poses a case as an example to explain advice, he has in mind some rationale about the case. A rationale relates a case's facts and a conclusion through an explanation in terms of some principles which, in the decider's opinion, control the case. According to the rationale, the controlling principles make certain facts about that case important and warrant the result. The rationale's controlling principles are drawn from some more general domain theory. In posing the example, the explainer asserts that the rationale applies to the problem, as well as to the case. The case's relevance to the problem depends intimately on the rationale (and domain theory) with which the explainer offers it and also on the alternative rationales which might reasonably apply to the case. Given a different domain theory or rationale, the case may be irrelevant, relevant in a different sense or degree, or even a counterexample to the advice.

Human experts, especially in law, are good at working with case examples from the vantage point of differing theories and rationales. They not only know how to characterize a case in terms of a rationale or theory, but can shift rationales or theories and recharacterize a case from the new vantage point. They can find or construct examples to point out differences among theories and rationales and even to test them. Given an example they can come up with a rationale to explain it. Given a rationale, they can create an example which is problematic. McCarty provides a vivid example of how expert attorneys (Supreme Court Justices) generated competing hypothetical examples with which to characterize a dividend of corporate stock. Depending on their view of its taxable status, they called it a 'gain derived from capital' or simply a recasting of a capital share (McCarty 1980). Lakatos provides examples from a hypothetical tutoring discourse in which, given a conjectural definition of a polyhedron, students pose 'monster' counterexamples leading to monster-barring revisions of the definition (Lakatos 1976, pp. 7–53).

Instances of dialectical reasoning with case examples, principles and rationales are easy to find in law. Here are two, both drawn from the domain of illegal searches in criminal cases.

The first example focuses on the interaction between theories, rationales, and outcomes. In a lecture, Wilfried Bottke, a professor of comparative criminal law, wished to

make a point about the fundamental difference between German and U.S. law on illegal searches. Both countries have similar rules excluding illegally obtained evidence from criminal trials. The principles underlying the exclusionary rules differ, however. In Germany, the governing principle is to protect an individual's privacy rights, while in the United States, it is to protect the integrity of the police process. As a result, a problem may involve the same facts, may even be governed by nearly identical rules, and yet have widely divergent outcomes in the two systems. To the audience, the implications of the difference were abstract, until the professor presented a case example: Suppose a private investigator breaks into a defendant's home and obtains incriminating letters. Are the letters admissible? Even though the two system's rules are similar, the outcomes in this case differ. Under German law, the letters would not be admissible because the investigator, though not a policeman, invaded the individual's privacy in his home. Under U.S. law, the letters would be admissible since no police action was involved. The different outcomes depend on and illustrate the different principles underlying the rules.

Legal reasoners regularly employ hypothetical examples to test rationales. For instance, how far can a rationale that focuses on a lack of police action be pushed? The second example comes from the oral argument of the Supreme Court case of *New Jersey v. T.L.O.*, 495 U.S. 325 (1985). An advocate argued that a high school assistant vice principal's search of a female student's purse was proper because the Fourth Amendment does not apply to limit searches of students by school officials. The official was searching for cigarettes (a teacher reported that the student had been smoking in the girls room) but found marihuana, customer lists and a quantity of small bills, instead. The Fourth Amendment has been interpreted as prohibiting unreasonable searches by law enforcement authorities, but should not apply, according to the state's advocate, to high school administrators. A justice tested the advocate's rationale for deciding the case by posing the following hypotheticals:

Q: Do you think then that a male teacher could conduct a pat-down search of a young woman student at age sixteen to find the cigarettes?
A: Yes, I believe that there is such a significant difference in the function performed by the school teacher during the school day that the Fourth Amendment shouldn't apply.
Q: And does that mean that their authority then to make searches, if the Fourth Amendment is completely inapplicable, extends to any kind of search, strip search or otherwise?
A: That would violate another Constitutionally guaranteed right, privacy (SUP 1985).

Each example illustrates a kind of purposive interpretation and reinterpretation of cases and dialectical invention of examples that lie at the core of legal reasoning, argumentation and explanation. Each is, I believe, beyond the capacity of current CBR programs and would require CBR researchers to invent new representations of theories, rationales, principles and cases.

Some insights and pieces of the solution exist. We have already seen a program, GREBE, represent and reason with a (simplified) case rationale. CASEY and PROTOS

both employ epistemological networks to represent expressly the connections between theory and low level case facts. Techniques for generating examples to meet dialectical constraints have been implemented in CATO (Ashley & Aleven 1992; Aleven & Ashley 1992) and in earlier work (Rissland & Soloway 1980; Rissland 1981). We have seen a variety of relevance measures, including Dimensional methods, structure-mapping, justified match and heuristic explanatory weights, which could be invoked to explain and test why differences among a case and problem matter. Computationally, Dimensions can represent hypothetical modifications. In the above example, for instance, the justice appears to have made the problem situation more extreme along two applicable Dimensions: along an 'authority' Dimension from a vice-principal down to a teacher and along an 'intrusiveness' Dimension from a search through a pocket book to a pat down search. (For other examples and HYPO's heuristics for posing hypotheticals, see (Ashley 1990, pp. 148–154, 233–237; Rissland & Ashley 1986). Something like CABARET's heuristics could control the interplay between examples and generalizations (see Rissland and Skalak 1991).

Despite the progress, there still are significant open questions. How should one represent case rationales and theories for ill-structured legal problems? Clearly, principles, policies, and purposes play important roles, but so far as I am aware, no AI program for the legal domain represents them in robust way.[10] GREBE, for instance, employs only criterial facts and statutory terms. CABARET does not represent statutory or interpretive principles or legislative purposes, important components of statutory interpretation. Indeed, although CBR programs generally assume enough of some domain theory to support case-based comparisons and inferences, often that theory is not expressly represented or only partially so (as in HYPO). Some progress has been made in generating explanations of events in terms of high level goals and plans and anomalous violations of goal-related expectations (Leake 1991; Kass et al 1986; Owens 1988). AI research, however, has not come up with a robust computational model of the links between high level interpretive concepts such as principles, policies, and purposes, and low level facts.

Missing, too, are programs that can shift among alternative theories and interpret cases from the viewpoint of more than one theory or rationale. Generally, the programs cannot recharacterize cases in terms of different theories. Although PROTOS's use of cases to elaborate classification concepts is a suggestive approach, it does not support multiple case interpretations. One cannot ask PROTOS to attempt to characterize a new case in

[10] For a proposal for research to factor statutory purposes into the interpretation and application of legal precedents, see (Kedar-Cabelli 1984).

More recently, Keith Bellairs has designed a program, BRAMBLE, which treats case relevance as a function of a problem solver's goal structures, problem conceptualization and utility of the precedent in attaining the goal (Bellairs 1989). The program represents concepts as heuristic schemas, not complete definitions. Given a problem, BRAMBLE selects a concept that fits the problem, retrieves precedents that are similar in that the concept fits them, too, selects the precedents that are useful to achieving a problem goal and applies them to solving the problem. In an experiment, the program selected conceptually relevant cases despite the lack of superficial similarity, made different relevance judgments in different conceptual contexts, and drew multiple analogies from a single case. However, the program and experiment dealt with three simple cases, a very small number.

terms of alternate rationales for an exemplar or definitions of a concept. HYPO, CABARET and GREBE can, at least, characterize a case from opposing sides' viewpoints. McCarty's TAXMAN II modeled an interesting Supreme Court argument involving the use of hypotheticals to propose or debunk alternative rationales, but the program did not appear to have a computational model of why the hypothetical modifications strengthened or weakened a conclusion (McCarty & Sridharan 1981, 1982). The SWALE program can generate alternative possible explanations of events such as the death of a famous race horse based on different remembered cases of surprising deaths. It is a good attempt at designing a general representation to support alternate explanations (Kass et al. 1986). The gulf between the general representation of expectation violations and specific facts is enormous, however, and seems to require lots of hand-tailored links, casting doubt on the approach's robustness.[11]

Nor are the programs adept at employing examples to test theories. Ironically, the researchers with the best data on the phenomenon of testing rationales with cases are probably those who build rule-based heuristic expert systems. Developers of these systems employ cases in knowledge acquisition, though typically the user sees only the resulting rules shorn of cases. Knowledge engineers employ cases and hypotheticals to test formulations of rules proferred by experts. Waterman described the process:

> In developing an expert system, researchers (typically computer scientists with some knowledge of the area of expertise) 'pick the brains' of a small number of experts. To do this, they present an expert with a hypothetical case requiring a decision. The expert then indicates what decisions should be made and why. The researchers can systematically explore the decision process by varying the facts presented to the expert and noting how changes in facts change both the decision and the chain of reasoning supporting the decision (Waterman & Peterson 1981, p. 14).

According to the authors, even a computer scientist 'with some knowledge of the area' can employ this methodology to test a rule.

Although cases play an intimate role in suggesting, testing and refining expert system rules, knowledge engineers typically do not incorporate them into the expert system. Protocols of such knowledge acquisition episodes may be worth a second look, not for the rules they engendered, but for the insight they may provide on the process of posing hypotheticals.

Significantly, in a knowledge acquisition context, the hypothetical case-posing methodology can empower even a domain novice to test an expert's advice. Cognitive scientists should strive to model a knowledge representation control methodology that enables even one who is not a domain expert to select and pose meaningful hypotheticals which induce an expert to refine and improve a rule set. That same methodology could enable expert systems users to test the limits of the system's advice and reasoning. At least, an expert system could employ the hypothetical cases which informed the choice of a particular formulation of a heuristic rule as an example to illustrate the rule's origin, intended meaning and limitations of application. PROTOS's techniques for automatically

[11] Students of Roger Schank have developed a number of approaches to indexing cases as stories (Riesbeck & Schank 1989). See, for example, (Leake 1991) (indices derived from explanations of anomalies); (Domeshek 1991) (indices derived from relationships of causal intensionality and themes).

acquiring explanations of cases and invoking cases as examples of concepts may be helpful in this regard.

4.5. INTELLIGENT INFORMATION RETRIEVAL

CBR research directly addresses an important need that rule-based approaches have not: the need to improve access to relevant materials in legal databases. It has been argued that AI's preoccupation with building expert systems is misguided for all but the most limited kinds of problem solving activities, ones where novel problems do not arise and the need for interpretation is minimized (Schank 1984, pp. 32–38).

Instead of building legal expert systems that answer legal problems, it may be better to build systems that assist legal professionals/users to find relevant source materials with which they may answer their own problems, for instance: an improved interface to on-line full text retrieval systems, an 'Intelligent Brief Writer's Assistant' to help in creating legal arguments and finding support among published cases or a law firm's legal memoranda and briefs, or automated drafting assistants with libraries of design materials such as contractual agreements or trust provisions. With the burgeoning amount of on-line legal information, it is an urgent call.

Ideally, legal database systems would make information available tailored to the particular context of the user's problem and would employ the same relevance criteria that human experts do. Clearly, existing full-text retrieval systems accomplish neither goal. Lexis and Westlaw have long depended on a criterion of relevance, the sharing of keywords, which no one would defend as adequate for modelling real legal arguments and which are subject to a well-documented problem of key word searches: very low recall at adequate levels of precision (Blair & Maron, 1985). On the other hand, they seem to achieve some level of practical utility at very little cost (relative to the volume of materials) in terms of case representation and indexing. They store and access vast numbers of cases efficiently, index construction is automatic, and there is no need for a special case representation language or for humans to interpret cases in filling out case representation forms. In an inverted index text retrieval system, typists simply enter cases on line and the program indexes them automatically under all of the big words they contain.

Conversely, CBR programs offer the promise of more meaningful relevance criteria but transforming them into full-scale information retrieval systems presents obstacles: the relevance criteria must be computationally tractible even for large numbers of complex cases, and, probably, some human intervention will be required to represent the cases for indexing and retrieval. Although great strides have been made in automated abstraction of texts (see, for example, (Riloff & Lehnert 1992)), the ability to classify legal texts automatically is still likely to be far off. It depends on advances in natural language understanding which are still elusive.

These challenges have no easy solutions, so I will only venture two recommendations for research in this area.

First, we should strive to develop better cognitive models of relevance. One presumes that expert methods for assessing relevant similarities and differences among cases

involve structured explanations, abstract concepts like rule predicates, rationales, theories, principles and purposes, as well as procedural settings and strategic considerations. As far as I know, however, researchers have not conducted systematic empirical studies of how experts assess relevance. Unfortunately, the observation over twenty years ago of one of AI's foremost researchers, is still true today:

> We know too little about the styles and structures of legal research strategies. Although lawyers do legal research and solve legal problems day in and day out, systematic [empirical] analysis of this process has been rare. As a consequence, our models of the legal research process are incomplete and oversimplified. We have no solid base of data on the legal research process, and of course, have not attempted an exhaustive description of legal thought processes (Buchanan & Headrick 1970, p. 47).

We need to observe and analyze legal problem solvers: '(1) finding conceptual linkages in pursuing goals, (2) recognizing facts, (3) resolving rule conflicts, and (4) finding analogies (Buchanan & Headrick 1970, p. 53)' and, I would add, (5) comparing cases and drawing inferences from those analogies.

Better cognitive models of relevance will not necessarily lead to computationally tractible implementations of those relevance criteria, at least not with the large numbers of cases required in legal information systems.

Computational relevance criteria, however, may not have to be as sophisticated as those human experts employ to present and summarize materials effectively. We may discover certain systematic relationships between simpler, fact-oriented relevance measures and complex conceptual ones. The goal of conceptual analyses, after all, is to predict or rationalize the significance of facts. Connections between lower level facts (e.g., factors as represented by Dimensions in HYPO) and higher level concepts may exhibit enough regularities (statistically or semantically) that the former may be used to filter and rank cases more likely to exhibit the same conceptual issues as a problem.

We may also find that expert relevance criteria are really composites of simpler criteria expertly applied. In this paper, we have seen four basic techniques for reasoning symbolically about relevance: justified matches in CASEY, explanation-related heuristic weights in PROTOS, Dimensional analysis in HYPO and its integration with statutory predicates in CABARET and structure-mapping in GREBE. All of these methods of symbolically reasoning about relevance seem to me to be valid. Intuitively, I am sure that I have encountered each of them in my own legal practice (even, sometimes, more statistically-oriented relevance criteria). Depending upon the domain, the task, and the bibliographic research tools at my disposal, each has led to some legal insight about a particular problem or argument. Of course, although all of the relevance measures are valid and useful, they are not all useful all of the time. Perhaps the most important skill of experienced legal researchers is knowing when to apply a particular relevance measure.

The CBR and AI and Law communities should focus on designing case database retrieval systems that can integrate more than one relevance measure and that are smart enough to adopt the relevance ordering that is most appropriate to the various parts of a user's task. Perhaps a computationally less expensive Dimensional analysis, or even a statistical analysis, is a good way to start a search and focus on a small number of cases. Then the computationally expensive structure mapping could be applied to a smaller

number of cases [Branting, 1991a] to select the one that best supports a proposition. In order to assess the proposition, it may be desirable to switch back to a Dimensional analysis to search for a counterexample, distinguish a close case or pose a hypothetical that exaggerates an important factor.

Controlling the switching among modes will, of course, be complex. Control heuristics like those of CABARET and extending the agenda mechanism to a blackboard architecture would seem a promising start (Rissland & Skalak 1991). It was noted, however, that CABARET's heuristically controlled analysis of a problem lacked an overall control plan (Rissland & Skalak 1991). Perhaps the particular heuristics are to blame. One result of an empirical analysis of legal research methods may be to identify human experts' heuristic control rules for helping to switch relevance measures opportunistically and effectively.

My second recommendation is to experiment with ways that small CBR systems can constrain searches into full text retrieval systems. In practice, legal researchers integrate conceptually indexed retrieval systems with full-text systems. For instance, one may begin researching a problem with a legal treatise or digest. Having selected some conceptually relevant cases, the researcher may then employ a full text retrieval system to determine whether a case is still good law, find other cases that cite it or other cases like it.

Conceivably, knowledge representation and control schemes can be designed to connect smaller-scale CBR systems with full-text retrieval. Small-scale systems could provide the same functionality as written works, provide greater assistance in helping a researcher to analyze a problem and find relevant cases and make it more convenient to pursue follow up queries in the full-text retrieval systems. These descendants of HYPO, CABARET, GREBE, and PROLEXS (and SCALIR, discussed below) will have specialized case bases, support alternative query methods beyond key word search, and graphically display data in a manner that conveys relevance and enables users to explore neighborhoods of alternatives. As with written works, the utility of such systems is not lessened because they do not contain all known cases. Their cases are selected because they are representative, interesting, novel, extreme, contradictory, recent, etc. Selecting, preparing, and characterizing the cases is a kind of work that humans perform now in creating written reference works; they can do something similar in selecting, preparing and characterizing cases in CBR systems.

It remains to be seen to what extent the output of a small-scale CBR program can constrain a search into full-text retrieval systems. In employing a CBR program, can the fact that a legal professional has entered a description of his problem, retrieved certain cases, and indicated that some retrieved cases were relevant be employed to generate automatically a search query into a full-text retrieval system to find other cases that would also be on point? It is trivial for a full-text system to search for other cases citing a retrieved case, an example of one simple kind of constraint. The open research question is whether any other useful constraints can be generated.

Whatever the answer to that question, small-scale case-based legal database systems could enable attorneys to explore visually represented spaces of cases organized systematically according to their relevance to a problem. CBR's focus on computationally

defining relevance criteria invites graphically representing retrieved materials according to relevance. Visual metaphors such as a smaller perceived distance between a retrieved case and a problem can signify greater relevance. Graphic displays could enable users to navigate through potentially relevant materials. Displays can also be reorganized according to alternative relevance definitions enabling a user to obtain alternative views of the relevant materials.

Graphic representations of information in legal databases and their relation to explanation is a fruitful area for further study. In general, the AI and Law community has tended not to emphasize sufficiently the development of graphic representations summarizing program outputs.[12] Graphic representations that apprise the user of where he or she is within a space of relevant alternatives would enable a system designer responsibly to shift some of the burden of explanation and justification to the user. Such an approach might go a long way to make up for the deficiencies in explanation in expert systems. From a practical viewpoint, designing prototype programs that generate graphic representations of legal materials would be a powerful incentive to corporate and governmental funders of AI and Law research.

The goal of enabling a user to navigate through graphically represented neighborhoods of cases relevant to the problem was imperfectly, but partially, realized in HYPO. Claim Lattices graphically represented a neighbourhood of cases that were relevant to a problem situation (Ashley 1990, pp. 128–147; Ashley & Rissland 1988a). Various relationships depicted in the Claim Lattice were interpretable directly in terms of arguments (e.g., relevant case, more on point case, most on point case, best case to cite, trumping counterexample, potential trump). There were a number of directionalities implicit in the neighborhood, visual senses in which one could move 'closer' to or farther 'away' from the problem. The farther one proceeded from the root, the less on point the cases were. In an extended Claim Lattice, one could move to cases that were nearly more on point (Ashley 1990, pp. 56f, 136–139). One could also move a case along an applicable Dimension or through a series of hypothetical variants.

The potential for empowering a user to browse in a space of relevant materials is more dramatically present in SCALIR, a connectionist retrieval system in which legal documents, such as case opinions, are linked to other documents that share common terms or are cited in the opinion (Rose & Belew 1991). The system retrieves relevant documents by spreading activation through the network of documents, orders the retrieved items according to activation levels that reflect weights on the links and presents them graphically to the user. Weights on the common term links are computed automatically and estimate the power of the term to discriminate that document from others in the collection. They are a function of the frequency of the term in the document and, inversely, of the term in other documents. (The weights are not unlike those employed in MBRTALK). By marking selected nodes for feedback, the user can refocus the search, and the graphic display, to emphasize particular aspects of his or her query.

[12] Probably, researchers and graduate students in AI and Law are not especially talented or experienced in graphic interface design and are not sufficiently rewarded academically for devoting efforts to graphics.

4.6. CBR AND DOCUMENT DESIGN

The potential utility of case-based methods is not restricted to databases of legal opinions but extends to databases of legal 'designs'. Suppose a document assembly program represented a design space populated with past legal contracts and provisions. An attorney/user designing a new contract could navigate the space in search of alternative models for language that would address his or her problem. The contract-drafting task is complicated by the fact that, like design tasks in other domains, an attorney who drafts a contract makes many decisions. The decisions are interdependent; they condition and are conditioned by each other.

Researchers in CBR have been experimenting with ways to index and present graphically, in a manner helpful to designers, past designs whose components evidence such interdependencies. One possibility is to create a network based on protocols for designing a kind of agreement, and representing the design decisions, rationales for those decisions and examples drawn from past design cases. By graphically displaying the network, designers seeking models could locate those parts of the network dealing with design decisions they face (Mark & Schlossberg 1990). For instance, a computerized assistant for designing software license and development agreements might display a network branch concerning the 'decision to convey to purchaser full rights to vendor code'. A pointer would lead to actual provisions in prior contracts conveying full rights to market and distribute source code and to make derivative works.

Another tack is to provide a conceptual index to past cases that enables a designer to recognize parts of designs, which, though drawn from different contexts, serve similar purposes in a way that can be transferred to the current problem. See the discussion in (Sycara & Navinchandra 1991). Candidates for such indexing concepts include goals satisfied, or failed to be satisfied, as in the planning-oriented CBR paradigm, the specific purposes for which a document could be used (Mital et al. 1991), as well as concepts invoked by drafters in justifying or explaining why a provision was drafted (Barletta and Mark 1988). In a software license drafting assistant, such an index might include a category of 'Failures in goal to limit rights in software' indexing various provisions from cases where software licenses failed because of overreaching restrictions on the purchaser. For instance, a license to purchase software which purported to restrict the rights of a purchaser of a copy of the software to make any copies, even prohibiting preparation of a necessary archival copy, would violate Copyright Act Section 117.

Even Dimensional comparisons may be appropriate in a design/planning context (Sanders 1991). Past provisions may be partially ordered in various ways. For instance, there are various ways for carving up the rights to software more or less favorably to the purchaser. An intermediate division might reserve a full right of access to vendor source code and to prepare derivative works, but limit the rights to distribute and market source code. Such a provision is less favorable than one that reserves to the purchaser all rights. A partial ordering of such alternatives could be represented in a Dimension.

To the extent that a case-based design assistant for law-related design tasks actually seeks and applies past cases (as opposed to offering ways that the user can browse for

cases), it raises some issues crucial to further advances in planning and design-oriented CBR: (1) how to adapt past cases to current constraints automatically, and (2) how to deal with the control problem of deciding whether to adapt a retreived case or to look for a more similar case.

A robust model of adaptation of past cases has so far eluded CBR. Adapting a retrieved case reintroduces the problem of search; the index cuts short search in leading to a relevant case but the relevant case's solution must still be applied in the context of the problem. There must be some mechanism for searching a space of possible modifications with the goal of solving the problem. Depending on how retrieval of relevant cases is performed, there may also be a more similar case in the database, one that would require less adaptation, resulting in a tradeoff between continuing the index search for the better case and adapting the previously retrieved case. The planning oriented paradigm programs, PERSUADER and CHEF, illustrated some approaches to modification and repair, but they did not really address the tradeoff between retrieval and adaptation costs. In recent work based on MEDIATOR, Hinrichs and Kolodner identify a set of general adaptation heuristics for the JULIA program, a case-based meal planner and provide a nice illustration of the tradeoff problem (Hinrichs & Kolodner 1991).

The adaptation heuristics in JULIA specify various transformations of a plan. Each transformation deals with fixing a particular kind of constraint violation and has expected costs. The program assesses the reason that a retrieved plan fails to satisfy problem constraints and applies the appropriate transformation. The generic transformations include:

Specialize: substitute more specific variant of component
Generalize: substitute more general variant
Substitute-by-function: swap with functionally identical component
Substitute-sibling: swap with taxonomic sibling
Share-function: make one component serve two functions
Split-function: two components are made to serve one function in tandem.

Although such lists of general adaptation techniques are useful, successfully implementing adaptation may require a more domain-specific approach. With the growing expertise in legal document assembly, it would be interesting to learn whether a set of transformations could be specified for some category of legal agreements. In HYPO, I identified a set of heuristics for transforming cases to strengthen or weaken the plaintiff's side (Ashley 1990, pp. 148–154; Rissland & Ashley 1986). Are there Dimensions along which agreements can be modified? Are there other ways of conceptualizing the modification, say of a standard software license agreement, to strengthen or weaken a licensor's protection?

The tradeoff between searching an index for a more relevant case and adapting a case the system has already found was demonstrated dramatically in (Hinrichs & Kolodner 1991, p. 32). In experiments, the program solved certain problems more efficiently when the program's adaptation capability was turned off completely. Apparently, when adaptation was turned on, the program spent a lot of time searching through the space of

modification operators trying to adapt a case it had already found rather than looking for a better case to employ as a starting point.

One recent advance in dealing with the control problem of deciding whether to search for a better case has been reported in connection with the PRODIGY program. PRODIGY implements a model of Jaime Carbonell's theory of Derivational Analogy (Carbonell 1983; Carbonell & Veloso 1988). The goal is to employ cases to speed up a General Problem Solver. A General Problem Solver (GPS) solves problems by 'dividing and conquerring' (Ernst & Newell 1969). At each step, it chooses either: a subgoal from a current set, an operator to achieve a goal already chosen, or a set of variable bindings for the operator. In this fashion it searches through a space of possible solutions (the GPS Search).

PRODIGY seeks to speed up a General Problem Solver by resorting, where possible, to compiled solutions in cases. In PRODIGY, past cases are traces, or complications, of the general problem solver's solution process. For each step, a case records: the optional steps to choose from, whether the step was a success or failure, the justifications for the choice, and the reasons for any failure [Carbonell and Veloso, 1988]. PRODIGY trades GPS Search for the presumably more efficient search through a case index for a similar case (Indexed Search). At any given point in the GPS process, it seeks to find a past case that matches the context of a choice pending in the current problem. The cases are indexed by the detailed context in which the problem-solving choices were made including the alternatives considered and the justifications for making a choice. PRODIGY 'replays' the solution of the retrieved case in so far as it is relevant.

It is a prodigious amount of information to represent and apply, but according to a recent paper, PRODIGY should be able to tell whether to keep looking for a better case or to apply a case already retrieved (Veloso & Carbonell 1991). Assuming that the degree of similarity of retrieved cases increases monotonically with search time, the program should be able to determine when the derivative of the expected savings in GPS Search approaches the incremental cost of continuing the Indexed Search for a more similar case. When that happens, the program can reasonably stop looking for a better case.

4.7. COGNITIVE MODELING AND CBR RESEARCH IN AI AND LAW

4.7.1. *A New 'Casuistry'*

CBR research in law is part of a more general intellectual movement in Cognitive Science. Recently, researchers in different fields have studied how case comparison methods enable reasoners to transform ill-structured problems, not solvable by deductive reasoning, into better-structured problems. Making decisions by comparing cases characterizes a style of reasoning and teaching in such domains as: mathematics (Lakatos 1976), government policy analysis (Neustadt & May 1986), business management (Gansler 1991), medical diagnosis, design (Sinha 1993), scientific theory building, strategic planning, law (Levi 1949), and practical ethical reasoning (Jonsen & Toulmin 1988; Strong 1988).

An ability to reason with cases appears to be at the core of a standard professional methodology for decision making and for bridging the gap between general principles and specific situations in a justified way. Professional education in many of these domains employs 'Case Methods' to teach students alternative ways for dealing with ill-structured problems.

Experts have striven, more or less independently, to develop systematic accounts of case comparison methods in a number of the domains. In addition to the CBR work in law discussed above, researchers in political science and ethics have also developed explicit statements of how experts employ cases.

In their book, *Thinking in Time*, for instance, Neustadt and May describe a methodology for policy decision making that makes intelligent use of historical analogies to justify decisions but also guards against their misuse. They offer three 'minimethods' for critically analyzing the applicability of a proposed analogy including focusing carefully on: (1) what is 'known, unclear and presumed' about the problem, (2) the relevant 'likenesses and differences' between the problem and proffered analogies and (3) the analogies' sometimes misleading sources of appeal (Neustadt & May 1986, pp. 89–90).

4.7.2. *An Example: Practical Ethical Decision-Making*

Ethicists like Jonsen, Toulmin and Strong have turned to case comparison methods as a way of specifying how to deal with competing principles. The interesting parallels between this work in ethics and that in legal reasoning illustrates the intellectual promise of CBR research in law.

Practical ethics involves making decisions about particular problems. One can discuss and argue ethical issues in the abstract, such as whether an expectant mother can refuse a transfusion on religious grounds, a comatose patient has a right to die, a teenager can be required to seek parental consent for an abortion or a soldier can disobey an order to fire. In making practical ethical decisions or creating policies, however, a reasoner must consider the issues in particular contexts and ascertain the consequences of particular features such as (in the first issue) a patient's condition, age, family dependents, family support, medical prognosis and more.

General ethical principles affect, but frequently do not determine, the outcome of the ethical analysis of particular problems. General ethical principles include, for instance, the principle of beneficence, that one ought to prevent harm to others, and the principle of autonomy, that one ought to respect the right of an individual to make his or her own decisions. Somewhat more specific principles (so called middle-level principles) may also apply, including the principle that a health official should avoid abandoning a patient or that a patient should be provided enough information to give an informed consent (Strong 1988, p. 201).

Experts may agree that certain general principles capture important information that should be relevant in making decisions. They may even agree that a principle applies to a particular fact situation and would lead to a course of action. Frequently, however, other principles also apply to the problem; the principles conflict and lead to inconsistent out-

comes. Experts may disagree whether the other principles apply and which principle should prevail. For example, how should one decide the following ethical problem where the principles of beneficence and autonomy conflict? [13]

A seventy-three-year-old man was mortally ill in a hospital and required a mechanical respirator. He had been judged competent, but his request to have the respirator disconnected was refused. He then attempted to disconnect it himself.... The patient contended [in court] that, in the face of his misery, the hospital and his physicians had an obligation to allow him to make his own choices, even though his choice entailed his death (Beauchamp & McCullough 1984, p. 14).

Typically, logical deductive methods are not effective because the antecedents of the general principles are not well defined and the general principles are not consistent. 'No moral philosopher has ever been able to present a system of moral rules free of these kinds of conflicts between principles and exceptions to principles' (Beauchamp & McCullough 1984, p. 16).

Weighting schemes are also ineffective. In practical ethics, if one could assign 'weights' to competing principles, resolving them would simply be a matter of comparing the weights. However, "the metaphor of the 'weight' of a principle of duty has not proven amenable to precise analysis" (Beauchamp & McCullough 1984, p. 16). For one thing, if one represents a weight by some number, it is not clear what the number means or how one computes it. If represented by some hierarchy of principles, the assignment of weights is not sufficiently sensitive to context. Principle A may usually be more important than principle B, but there are always some sets of circumstances in which principle B should win out. The parallel between the problem of assigning weights to principles in ethics and to factors in law is especially intriguing.

In the absence of the possibility of constructing formal logical proofs justifying a decision or of assigning weights to principles, other means of justifying decisions are required. Instead of a proof, a reasoner justifies a decision in terms of an analogy between the problem and past or hypothetical cases. The cases are either paradigmatic examples of the application of a principle whose decision is clear or authoritatively decided cases. Intentionally rehabilitating a term that had fallen into disrepute, Jonsen and Toulmin have urged applying a new casuistry to ethical decision making, defining it as:

the analysis of moral issues, using procedures of reasoning based on paradigms and analogies, leading to the formulation of expert opinions about the existence and stringency of particular moral obligations, framed in terms of rules or maxims that are general but not universal or invariable, since they hold good with certainty only in the typical conditions of the agent and circumstances of action (Jonsen & Toulmin 1988, p. 257).

Carson Strong has proposed a more systematic, five step 'Case Comparison Method' of justification in practical ethics. The last two steps involve comparing the problem to paradigmatic cases evidencing the various principles for deciding the problem in terms of the factors that favor applying or not applying the principle. His method comprises the following steps:

[13] This ethics problem was drawn from a court case in which the judge granted the patient's request. Satz v. Perlmutter, 362 So. 2d 160 (Fla. Dist. Ct. of App., 1978), aff'd 379 So. 2d 359 (Florida 1980).

1. '[I]dentify the middle-level principles and role-specific duties pertinent to the given situation.'
2. '[I]dentify the alternative courses of action that could be taken.'
3. '[I]dentify the morally relevant ways (i.e., factors) in which cases of the type in question can differ from one another....Comparing the case at hand with other cases of the same type also helps one identify these factors.'
4. '[F]or each option [that could be taken]..., identify a case [a paradigm] in which that option would be justifiable.... Paradigms can be actual or hypothetical cases. In addition, one should identify the middle-level principle which would provide that justification.'
5. "[T]he final element is a comparison of the case at hand with the paradigm cases which can be identified. One should try to determine which of the paradigms it is 'closest to' in terms of the presence of the morally relevant factors" (Strong 1988, pp. 202 – 205).

Strong's description of a case comparison method for practical ethics suggests a computational model much like HYPO's computational process for decision-making about trade secret law. Much remains to be specified, of course. How does one define the measure of similarity? How do ethicists draw inferences from the comparison to paradigms? And, are there actual paradigm cases that communities of ethicists refer to in deciding problems? The HYPO model also points out something missing in Strong's statement. Determining the 'closest' paradigm will doubtless involve arguments and counterarguments based on competing analogies. A precedent justifies a decision if and only if: (1) the past case is an authoritative precedent or a paradigmatic example whose decision is clear, (2) the analogy is apt and (3) the analogy is more apt than analogies to competing cases. To the extent these conditions are not fulfilled, the justification is susceptible to counterarguments distinguishing the analogy and analogizing to counterexamples.

AI can make an important contribution to this intellectual movement in practical ethics and other fields. In building computational models, models that will run on computers, AI research forces the new casuists to identify hidden assumptions and explicitly adopt simplifications (Schaffner 1990). The more systematically they can describe the method, the greater the chances of improving its application, by providing professionals with readily available paradigm cases, for instance, and by more effectively teaching the method to novices.

AI and law can contribute because we deal with a domain in which the case comparison method is so highly developed. We have a head start in building computational models of a case comparison method, models which may carry over into other fields. Legal professionals share more or less explicitly expectations about the kinds of case-citing arguments that are appropriate and, to some extent, how to evaluate them. Frequently, the cases are available in bibliographic sources and some authority has assigned an outcome to a case, explained it, or deemed it a success or failure. Experts who draft decisions or fashion contract provisions often record their reasons for each step. The profusion of legal indices means that cases can be generalized to a significant

extent; features that make a case relevant can be readily abstracted. Practitioners know the mechanism by which a reasoner draws inferences from comparing cases, what counts as a relevant similarity or difference and how to assess the comparisons.

4.7.3. *Comparative Studies with Case Comparison Methods*

The multiple uses of case comparison methods in different fields provide a rich vein for comparative Cognitive Science modeling employing AI techniques. AI researchers may be in a position to compare how human experts integrate principles, rules, cases, and paradigms in the various fields and to explain the differences in terms of the models. For instance, both fields of law and practical ethics involve a tension between general principles and rules on the one hand and specific cases on the other. Principles seem to play different roles in ethics and law, however. In ethics, according to Strong, the best-matching paradigm cases are said to enable a reasoner to resolve conflicting principles. In law, some legal philosophers speak as though the general principles enable a reasoner to determine the answer even to hard cases (Dworkin 1978). Also, in law, advocates commonly dispute whether a general principle really applies to a problem.[14]

Another difference which may be explainable in terms of a computational model involves the extent to which experts trust a case comparison method to give a 'just' as well as justified answer. In practical ethics, expectations are high:

> The case comparison method is based on and attempts to take appropriate account of [the actual nature of ethical dilemmas in clinical settings]... [and] yields conclusions in accord with our ordinary moral judgments concerning the resolution of cases (Strong 1988, p. 207).

By contrast, the legal case comparison method, though regarded as indispensible for justifying decisions, is not as highly regarded as insuring a just or correct answer. Engaging in case-citing arguments is regarded as having a game-like, ritualistic quality that presents too large a degree of freedom for rationalizing diametrically opposed decisions. (One thinks of what little it means for a judicial nominee to testify that he or she will 'respect' precedent given the freedom that distinguishing a precedent allows.) Also, there is a concern that case-based analysis in law may not touch the 'real' issues but remain superficial and technical. In short, it suffers form (indeed, caused) many of the complaints about the 'old' casuistry.

This is a matter of special interest to law professors, since legal education can sometimes blur the distinction between the old and the new casuistry. Can the roles that cases, arguments and principles play in training students to perform ethical reasoning be imported to legal reasoning to make it less game-like and to improve the depth of the analogical legal analysis? Perhaps the legal case comparison method serves a different institutional purpose. Although the actual mechanics of a judge's decision making in law are not well understood, a case comparison method's adversarial process of justification and counterargument insures that the relevant factual strengths

[14] Principles also play a role in statutory interpretation (Berman & Hafner 1986), although this role is not well understood.

and weaknesses of the problem have been identified and urged, and thus that the decision maker has, at least, considered them. 'The [law] forum protects the parties and the community by making sure that the competing analogies are before the court' (Levi 1949, p. 5).

4.7.4. *Tutorial Explanations*

In case-base domains like law, practical ethics and the other domains discussed above, one needs to train students in a case comparison method both to show them how justifications and explanations are performed in the domain and to train their professional judgement. Engaging a student in a kind of Socratic discourse in which arguments force the student to defend and justify claims in terms of comparisons to paradigmatic cases shapes a student's ability to think proficiently about problems in such domains (Collins & Stevens, 1982).

In studying how experts reason with cases, researchers should pay special attention to the different 'Case Methods' employed in tutoring novices in the domains. By focusing on case-based teaching methodologies in the various fields, researchers will gain valuable insights to reasoning and explaining in domains with weak models. Significantly, Clancey studied explanation and its relation to the way that an expert system represents knowledge in the context of tutoring.

> The demands of tutoring provide a 'forcing function' for articulating the structure of the rule base and the limitations of the program's explanation behavior. These insights have implications for generating and explaining consultative advice, and modifying the system. For the moment, consider that many, if not most or all, of the issues of tutoring must be addressed in providing explanations to naive users (Clancey 1983, p. 216).

Teachers in different domains employ cases in different ways. Business school professors appear to employ cases singly. They present a complex business case, encourage students to work through the detailed analysis to learn the lesson of the case, and move on to another case. Their focus is not so much on comparing different cases but employing the solved case as a notable sample of a kind of analysis or solution. Law professors and their casebooks focus on small collections of cases which students are invited to compare and contrast in class. They emphasize drawing conclusions about the legal topic from the comparison of the cases. Although this oversimplifies, and there are many variations, these two basic methods correspond to the the two basic inference methods in CBR: constraining search by tracing a solution from a past case or evaluating a case by comparing it to past cases.

With respect to either case-based teaching method, much of the interesting knowledge is only implicit in the final lesson. Simply making such knowledge explicit will go a long way toward a cognitive model of reasoning with cases, such as the:

— teacher's criteria for selecting cases to include in casebooks or course curricula.
— teacher's pedagogical toolkit of basic operations on cases with which to induce students to grapple with concepts. Such operations include drawing a conclusion about a problem by analogy to a case, distinguishing cases, finding counterexamples and

modifying a problem hypothetically to test an assertion, strengthen or weaken an analysis, or pose a quandry.

— underlying links between the cases and general concepts they are meant to exemplify or qualify. Such links include Dimensions for transforming and comparing problems in relevant ways that affect a concept's application.

— Socratic control method for orchestrating a series of such operations to lead students to perceive and absorb the intended lesson.

Significantly, the intended lesson comprises not only the topics and concepts that underlie the selection of cases, but the very process of transforming and comparing cases to elucidate and manipulate the concepts. Miraculously, this process 'scales up' in human students; having learned the methodology by example, they can apply it creatively to different and ever more complex circumstances.

Like Clancey, we are examining these problems of justification and explanation in the context of tutoring, tutoring law students to reason with cases. Our recent work on law school tutoring (see Section 2.5.2), asks Clancey's questions of a case-based legal expert system: Can the knowledge that enabled HYPO to generate arguments citing cases be represented in such a way that a program could justify its advice, explain its criteria, and even tutor a novice about these basic concepts? We believe so. Our work on generating Argument Contexts elucidates the teacher's criteria for selecting case examples to teach a lesson and points towards a cognitive model of how a teacher, student, or a CBR program, can explain an abstract concept by setting up a case comparison to make the point.[15]

5. Conclusions

At the Tenth National Conference on Artificial Intelligence (AAAI-92), held in San Jose, California, Evangelos Simoudis' and my tutorial on Case-Based Reasoning had the highest attendance of any of the tutorials. Kristian Hammond delivered an Invited Talk on Case-Based Reasoning to a packed auditorium. In the exhibition hall, crowds formed at the exhibits of vendors of CBR system-building shells. In short, the topic of Case-Based Reasoning is 'hot'.

I view this development with some satisfaction, but also alarm. If the availability of 'off the shelf' tools to build case-based expert systems raises the expectations of applications-oriented system developers beyond the power of the tools to deliver, CBR may go the way of other oversold AI techniques. I believe CBR deserves better. Drawing conclusions and making decisions by comparing a problem to past cases is a fundamental trait of human reasoning. Building programs that reason with cases has been an arduous task with many false starts and some successes. It does not seem likely to me that, suddenly, the path will be made smooth.

[15] Another area where CBR research may make a contribution to cognitive modeling is in the area of understanding jury decision making in terms of story schemas and scripts. See (Moore 1989) citing (Schank & Abelson 1977).

It would be surprising if shell builders had anticipated all of the ways in which human experts employ cases. It is true that all case-based reasoning involves generalizing cases for purposes of indexing and relevance assessment and evidences two kinds of inference making, constraining search by tracing a solution from a past case or evaluating a case by comparing it to past cases. Across domains and tasks, however, humans reason with cases in very different ways evidencing different mixes of and mechanisms for the three components: generalization, constrained search, and comparative evaluation. It is no accident that I felt it necessary in this article to present five paradigmatic approaches to Case-Based Reasoning, statistically-oriented, model-based, planning/design-oriented, exemplar-based, and adversarial or precedent-based, rather than one unified theory. As illustrated by the representative programs, the paradigms differ widely in the assumptions they make about domain models, the extent to which they support symbolic case comparison, and the kinds of inferences for which they employ cases.

Especially in the field of AI and Law, it would be a shame if so rich and promising a field as CBR succumbs to disappointed expectations driven by hyperbolic claims. Case-based reasoning is important in legal practice of all kinds, not just for adversarial argument in Common Law jurisdictions but in planning, evaluation and explanation in diverse legal tasks and regimes. For any mode of reasoning with cases, in classification, design, diagnosis, evaluation and assessment, planning, teaching, testing and others, one can find a corresponding use of cases in law. There are strong reasons justifying the vigorous pursuit of CBR research and development in the field of AI and Law. Case-based approaches can supplement rule-based expert systems, improving their abilities to reason about statutory predicates, solve problems efficiently, and explain their results. CBR can also contribute to the design of intelligent legal data retrieval systems, improve legal document assembly programs and contribute to Cognitive Science models of the use in various fields of case comparison methods to transform ill-structured problems into better structured ones. All of this will require patient effort, however. Here, as elsewhere in AI, there are few shortcuts.

Acknowledgements

This paper is based, in part, on notes for a tutorial on Case-Based Reasoning which I first presented in June, 1989 at the Second International Conference on Artificial Intelligence and Law (ICAIL) at the University of British Columbia, Vancouver, BC, Canada and again at the Third ICAIL conference in Oxford, U.K. in June, 1991. On three other occasions, I have co-delivered CBR tutorials which were based, in part, on my original tutorial notes: Dr. Katia Sycara of Carnegie Mellon University and I presented two CBR tutorials at AAAI-91 in Anaheim, California and IJCAI-91 in Sydney, Australia, and Dr. Evangelos Simoudis of the Lockheed AI Lab and I presented the CBR tutorial at AAAI-92 in San Jose, California. I have benefitted from Dr. Sycara's and Dr. Simoudis' suggestions, revisions and updates of the tutorial notes. Professor Donald Berman's thoughtful editorial comments and suggestions have led me to rethink various assertions and helped me to improve the final draft of this article. I particularly want to thank

Vincent Aleven for his careful reading of the final draft and painstaking editorial comments. Bruce McClaren read an intermediate draft and also provided very useful feedback.

References

Aleven, V. & Ashley, K. D. 1992. Automated Generation of Examples for a Tutorial in Case-Based Argumentation. In *Proceedings of the Second International Conference on Intelligent Tutoring Systems*. Montreal.

Allen, L. E. & Engholm, C. R. 1978. Normalized Legal Drafting and the Query Method. *Journal of Legal Education* 29:380–412.

Allen, L. E. & Saxon, C. S. 1987. Some Problems in Designing Expert Systems to Aid Legal Reasoning. In *First International Conference on Artificial Intelligence and Law*, 94–103, Northeastern University, Boston.

Ashley, K. D. & Aleven, V. 1991. Toward an Intelligent Tutoring System for Teaching Law Students to Argue with Cases. In *Proceedings of the Third International Conference on Articial Intelligence and Law*, 42–52, Oxford, England.

Ashley, K. D. & Aleven, V. 1992. Generating Dialectical Examples Automatically. In *Proceedings AAAI-92*. American Association for Artificial Intelligence, San Jose, CA.

Ashley, K. D. & Rissland, E. L. 1987. Compare and Contrast, A Test of Expertise. In *Proceedings AAAI-87*. American Association for Artificial Intelligence, Seattle.

Ashley, K. D. & Rissland, E. L. 1988. A Case-Based Approach to Modeling Legal Expertise. *IEEE Expert*. Vol. 3, Nr. 3.

Ashley, K. D. & Rissland, E. L. 1988. Waiting on Weighting: A Symbolic Least Commitment Approach. In *Proceedings AAAI-88*. American Association for Artificial Intelligence, St. Paul.

Ashley, K. D. 1989. Defining Salience in Case-Based Arguments. In *Proceedings IJCAI-89*. International Joint Conferences on Artificial Intelligence, Detroit.

Ashley, K. D. 1990. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. MIT Press, Cambridge. Based on Ashley's 1987 PhD. Dissertation, University of Massachusetts, COINS Technical Report No. 88–01.

Ashley, K. D. 1991. Reasoning with Cases and Hypotheticals in HYPO. *International Journal of Man-Machine Studies* 34 (6):753–796.

Bareiss, E. R., Porter, B. W. & Wier, C. C. 1988. Protos: An Exemplar-Based Learning Apprentice. *International Journal of Man-Machine Studies* 29:549–561.

Bareiss, E. R., Porter, B. W. & Murray, K. S. 1989. Supporting Start-to-Finish Development of Knowledge Bases. *Machine Learning* 4:259–283.

Bareiss, E. R. 1988. *Exemplar-Based Knowledge Acquisition – A Unified Approach to Concept Representation, Classification, and Learning*. Academic Press, San Diego, CA. Based on PhD thesis, University of Texas.

Barletta, R. & Mark, W. 1988. Explanation-Based Indexing of Cases. In *Proceedings of the Case-Based Reasoning Workshop*, 50–60, Clearwater Beach, FL. Defense Advanced Research Projects Agency – Information Science and Technology Office.

Beauchamp, T. & McCullough, B. 1984. *Medical Ethics: The Moral Responsibilities of Physicians*. Prentice-Hall, Englewood Cliffs, NJ.

Bellairs, K. 1989. *Contextual Relevance in Analogical Reasoning: A Model of Legal Argument*. PhD thesis, University of Minnesota.

Berman, D. H. & Hafner, C. 1986. Obstacles to the Development of Logic-Based Models of Legal Reasoning. In Charles Walter (ed.), *Computer Power and Legal Language*, 185–214. New York: Quorum Books.

Berman, D. H. & Hafner, C. D. 1991. Incorporating Procedural Context into a Model of Case-Based Legal Reasoning. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, 12–20, Oxford, England.

Blair, D. C. & Maron, M. E. 1985. An Evaluation of Retrieval Effectiveness for a Full-Text Document-Retrieval System. *Communications of the ACM* 28(3): 289–299.

Branting, L. K. & Porter, B. W. 1991. Rules and Precedents as Complementary Warrants. In *Proceedings AAAI-91*, 3–9. American Association for Artificial Intelligence, Anaheim.

Branting, L. K. 1991. Building Explanations from Rules and Structured Cases. *International Journal of Man-Machine Studies* 34(6): 797–837.

Branting, L. K. 1991. *Integrating Rules and Precedents for Classification and Explanation: Automating Legal Analysis*. PhD thesis, The University of Texas at Austin. Artificial Intelligence Laboratory, Technical Report No. AI90–146.

Buchanan, B. G. & Headrick, T. E. 1970. Some Speculation about Artificial Intelligence and Legal Reasoning. *Stanford Law Review* 23: 40–62.

Burton, S. J. 1985. *An Introduction to Law and Legal Reasoning*. Little, Brown, Boston.

Capper, P. N. & Susskind, R. E. 1988. *Latent Damage Law – The Expert System*. London: Butterworths.

Carbonell, J. & Veloso, M. 1988. Integrating Derivational Analogy into a General Problem Solving Architecture. In *Proceedings of the Case-Based Reasoning Workshop*, Clearwater Beach, FL. DARPA/ISTO.

Carbonell, J. G. 1983. Derivational Analogy and Its Role in Problem Solving. In *Proceedings AAAI-83*, Washington, DC. American Association for Artificial Intelligence.

Clancey, W. J. 1983. The epistemology of a rule-based expert system: A framework for explanation. *Artificial Intelligence* 20: 215–251. Reprinted in: Buchanan B. G. & Shortliffe E. H. (eds.), *Rule-Based Expert Systems; The MYCIN Experiments of the Stanford Heuristic Programming Project*.

Collins, A. & Stevens, A. L. 1982. Goals and Strategies of Inquiry Teachers. In Robert Glaser, editor, *Advances in Instructional Psychology*, Volume 2. Hillsdale, NJ: Lawrence Erlbaum Associates.

Domeshek, E. A. 1991. Indexing Stories as Social Advice. In *Proceedings AAAI-91*, 16–21. Anaheim: American Association for Artificial Intelligence.

Dworkin, R. 1978. No Right Answer? *New York University Law Review* 53:1–32.

Eisenberg, T. & Henderson, Jr., J. A. 1992. Inside the Quiet Revolution in Products Liability. *UCLA Law Review* 39: 731.

Ernst, G. W. & Newell, A. 1969. *GPS: A Case Study in Generality and Problem Solving*. New York: Academic Press, ACM Monograph Series.

Gansler, J. S. 1991. Acquisition of High Technology Equipment: An Application for Case-Based Reasoning. Address presented at DARPA Workshop on Case-Based Reasoning. Washington, D.C.

Gardner, A. vdL. 1987. *An Artificial Intelligence Approach to Legal Reasoning*. Cambridge: MIT Press.

Golding, A. R. & Rosenbloom, P. S., 1991. Improving Rule-Based Systems through Case-Based Reasoning. In *Proceedings AAAI-91*, 22–27. Anaheim: American Association for Artificial Intelligence.

Greenleaf, G., Mowbray, A. & Tyree, A. 1991. The DataLex Legal Workstation – Integrating Tools for Lawyers. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, 215–224, Oxford, England.

Hammond, K. J. 1986. CHEF: A Model of Case-based Planning. In *Proceedings AAAI-86*, 267–271. Philadelphia: American Association for Artificial Intelligence.

Hammond, K. J. 1986. Learning to Anticipate and Avoid Planning Problems through the Explanation of Failures. In *Proceedings AAAI-86*, 556–560. Philadelphia: American Association for Artificial Intelligence.

Hammond, K. J. 1987. Explaining and Repairing Plans That Fail. In *Proceedings IJCAI-87*. Milan: International Joint Conferences on Artificial Intelligence.

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

Henderson, J. A. Jr. & Eisenberg, T. 1990. The Quiet Revolution in Products Liability: An Empirical Study of Legal Change. *UCLA Law Review* 37:479.

Henrion, M., Breese, J. S. & Horvitz, E. J. 1991. Decision Analysis and Expert Systems. *AI Magazine* 12(4): 64–91.

Hinrichs, T. R. & Kolodner, J. L. 1991. The Roles of Adaptation in Case-Based Design. In *Proceedings AAAI-91*, 28–33. Anaheim: American Association for Artificial Intelligence.

Jonsen, A. R. & Toulmin, S. 1988. *The Abuse of Casuistry A History of Moral Reasoning*. Berkeley: University of California Press.

Kass, A. M. Leake, D. & Owens, C. C. 1986. Swale: A Program that Explains. In Roger C. Schanck, editor, *Explanation Patterns: Understanding Mechanically and Creatively*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kedar-Cabelli, S. 1984. Analogy with Purpose in Legal Reasoning from Precedents: A Dissertation Proposal. Technical Report LRP-TR-17, Laboratory for Computer Science Research, Rutgers University.

Kolodner, J. L. Simpson, R. L. & Sycara-Cyranski, K. 1985. A Process Model of Case-Based Reasoning in Problem Solving. In *Proceedings IJCAI-85*, 284–290, Los Angeles: International Joint Conferences on Artificial Intelligence.

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

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

Koton, P. 1988. Reasoning about Evidence in Causal Explanations. In *Proceedings of the Case-Based Reasoning Workshop*, 260–270, Clearwater Beach, FL: DARPA/ISTO.

Koton, P. 1988. Reasoning about Evidence in Causal Explanations. In *Proceedings AAAI-88*, 256–261. St. Paul, MN: American Association for Artificial Intelligence.

Koton, P. 1988. *Using Experience in Learning and Problem Solving*. PhD thesis, MIT.

Kuhn, T.S. 1980. *The Structure of Scientific Revolutions*. University of Chicago Press.

Lakatos, I. 1976. *Proofs and Refutations*. London: Cambridge University Press.

Leake, D. B. 1991. An Indexing Vocabulary for Case-Based Explanation. In *Proceedings AAAI-91*, 10–15. Anaheim: American Association for Artificial Intelligence.

Lesgold, A. M., Bonar, J. G., Ivill, J. M. & Bowen, A. 1987. An intelligent tutoring system for electronics troubleshooting: DC-circuit understanding. Technical report, Learning Research and Development Center, University of Pittsburgh.

Levi, E. L. 1949. *An Introduction to Legal Reasoning*. University of Chicago Press.

Llewellyn, K. N. 1930. *The Bramble Bush: On Our Law and Its Study*. Dobbs Ferry, NY: Oceana Publications, 1960 edition.

Llewellyn, K. N. 1938. On Our Case Law of Contract: Offer and Acceptance. *Yale Law Journal* 48: 1–36, 779–818.

Llewellyn, K. N. 1989. *The Case Law System in America*. Chicago: University of Chicago Press. Recent translation of *Praejudizienrecht und Rechtssprechung in Amerika*, 1933.

MacGregor, R. M. 1988. A Deductive Pattern Matcher. In *Proceedings AAAI-88*, 403–408, Saint Paul, MN: American Association for Artificial Intelligence.

Mark, W. & Schlossberg, J. 1990. A Design Memory Without Cases. In *Proceedings of the Spring Symposium Series*, 51–55. Palo Alto: American Association for Artificial Intelligence.

Mark, W. S. 1989. Case-Based Reasoning for Autoclave Management. In *Proceedings of the Case-Based Reasoning Workshop*, 176–180, Pensacola Beach, FL: Defense Advanced Research Projects Agency – Information Science and Technology Office.

McCarty, L. T. & Sridharan, N. S. 1981. The Representation of an Evolving System of Legal Concepts: II. Prototypes and Deformations. In *Proceedings IJCAI-81*, Vancouver, BC: International Joint Conferences on Artificial Intelligence.

McCarty, L. T. & Sridharan, N. S. 1982. A Computational Theory of Legal Argument. Technical Report LRP-TR-13, Laboratory for Computer Science Research, Rutgers University.

McCarty, L. T. 1980. A Computational Theory of Eisner v. Macomber. Technical Report LRP-TR-9, Laboratory for Computer Science Research, Rutgers University.

McKeown, K. R., Wish, M. & Matthews, K. 1985. Tailoring Explanations for the User. In *Proceedings IJCAI-85*, Los Angeles: International Joint Conferences on Artificial Intelligence.

Miller, R. A., Schaffner K. F. & Meisel, A. 1985. Ethical and Legal Issues Related to the Use of Computer Programs in Clinical Medicine. *Annals of Internal Medicine* 102(4): 529–536.

Mital, V., Stylianou, A. & Johnson, L. 1991. Conceptual Information Retrieval in Litigation Support Systems. In *Proceedings of the Third International Conference on Artificial Intelligence and Law*, 235–243, Oxford, England.

Moore, J. D. & Swartout, W. R. 1989. A Reactive Approach to Explanation. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, MI. Reprinted as Tech. Rep. ISI/RS-89-239, Information Sciences Institute, University of Southern California, Marina del Rey, CA.

Moore, J. D. & Swartout, W. R. 1990. A Reactive Approach to Explanation: Taking the User's Feedback into Account. In Paris, C., Swartout, W. & Mann, W. (eds), *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, 1–48. Dordrecht: Kluwer Academic Publishers.

Moore, J. D. & Swartout, W. R. 1990. Pointing: A Way Toward Explanation Dialogue. In *Proceedings AAAI-90*, 457–464. Boston: American Association for Artificial Intelligence.

Moore, A. 1989. Trial by Schema: Cognitive Filters in the Courtroom. *UCLA Law Review* 37:273.

Neustadt, R. E. & May, E. R. 1986. *Thinking in Time*. New York: Free Press.

Nimmer, R. & Krauthaus, P. A. 1986. Computer Error and User Liability Risk. *Jurimetrics* 26(2): 121–137.

Owens, C. 1988. Domain-Independent Prototype Cases for Planning. In *Proceedings of the Case-Based Reasoning Workshop*, 302–311, Clearwater Beach, FL: DARPA/ISTO.

Radin, M. 1933. Case Law and Stare Decisis. *Columbia Law Review* 33:199.

Riesbeck, C. K. & Schank, R. C. 1989. *Inside Case-Based Reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Riloff, E. & Lehnert, W. 1992. Classifying Texts Using Relevancy Signatures. In *Proceedings AAAI-92*. San Jose, CA: American Association for Artificial Intelligence.

Rissland, E. L. & Ashley, K. D. 1986. Hypotheticals as Heuristic Device. In *Proceedings AAAI-86*. Philadelphia: American Association for Artificial Intelligence.

Rissland, E. L. & Skalak, D. B. 1989. Combining Case-Based and Rule-Based Reasoning: A Heuristic Approach. In *Proceedings IJCAI-89*. Detroit: International Joint Conferences on Artificial Intelligence.

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

Rissland, E. L. & Soloway, E. M. 1980. Overview of an Example Generation System. In *Proceedings AAAI-80*, Stanford, CA: American Association for Artificial Intelligence.

Rissland, E. L. 1981. Constrained Example Generation. Technical Report 81–24, Computer and Information Science Department, University of Massachusetts, Amherst, MA.

Rissland, E. L. 1990. Artificial Intelligence and Law: Stepping Stones to a Model of Legal Reasoning. *Yale Law Journal* 99(8):1957–1981.

Rose, D. E. & Belew, R. K. 1991. A Connectionist and Symbolic Hybrid for Improving Legal Research. *International Journal of Man-Machine Studies* 35(1):1–33.

Sanders, K. E. 1991. Within the Letter of the Law: Reasoning Among Multiple Cases. In *Proceedings of the Case-Based Reasoning Workshop*, 317–326, Washington, DC: Defense Advanced Research Projects Agency – Information Science and Technology Office.

Schaffner, K. F. 1990. Case-Based Reasoning in Law and Ethics. Presentation at the 'Foundations of Bioethics' Conference. Hastings Center.

Schank, R. C. & Abelson, R. 1977. *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. 1984. *The Cognitive Computer: On Language, Learning and Artificial Intelligence*. Reading, MA: Addison-Wesley.

Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P. & Cory, H. T. 1986. The British Nationality Act as a Logic Program. *Communications of the ACM* 29(5):370–386.

Shortliffe, E. H. 1984. Details of the Consultation System. In Buchanan, Bruce G. and Shortliffe, Edward H. (eds.), *Rule-Based Expert Systems*, 78–132. Reading, MA: Addison-Wesley.

Simoudis, E. & Miller, J. S. 1991. The Application of CBR to Help Desk Applications. In *Proceedings of the Case-Based Reasoning Workshop*, 25–36. Washington, DC: Defense Advanced Research Projects Agency – Information Science and Technology Office.

Simpson, R. L. 1985. *A Computer Model of Case-Based Reasoning in Problem Solving*. PhD thesis, Georgia Institute of Technology. GIT-ICS-85/18.

Sinha, A. P. 1993. *IDEA: A Case-Based Model of Design Assistance*. PhD thesis, University of Pittsburgh, Katz Graduate School of Business. Pittsburgh.

Smith, J. C. & Deedman, C. 1987. The Application of Expert Systems Technology to Case-Based Law. In *First International Conference on Artificial Intelligence and Law*. Boston: Northeastern University.

Stanfill, C. & Waltz, D. 1986. Toward Memory-Based Reasoning. *Communications of the ACM* 29(12): 1213–1228.

Stanfill, C. W. 1987. Memory-Based Reasoning Applied to English Pronunciation. In *Proceedings AAAI-87*, 577–581. Seattle: American Association for Artificial Intelligence.

Strong, C. 1988. Justification in Ethics. In Baruch A. Brody, editor, *Moral Theory and Moral Judgments in Medical Ethics*, 193–211. Dordrecht: Kluwer Academic Publishers.

SUP. 1985. The Complete Oral Arguments of the Supreme Court of the United States. Frederick, MD: University Publications of America.

Susskind, R. E. 1987. *Expert Systems in Law: a Jurisprudential Inquiry*. Oxford: Oxford University Press.

Swartout, W. R. 1983. XPLAIN: a System for Creating and Explaining Expert Consulting Programs. *Artificial Intelligence* 21(3): 285–325.

Sycara, K. & Navinchandra, D. 1991. Influences: A Thematic Abstraction for Creative Use of Multiple Cases. In *Proceedings of the Case-Based Reasoning Workshop*, 133–144. Washington, DC: Defense Advanced Research Projects Agency–Information Science and Technology Office.

Sycara, K. 1987. *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods.* PhD thesis, Georgia Institute of Technology. School of Information and Computer Science, Technical Report No. 87–26. To be published under the title *Case-Based Reasoning: A Study in Conflict Resolution*, Lawrence Erlbaum Associates, Hillsdale NJ.

Sycara, K. 1989. Argumentation: Planning Other Agents' Plans. In *Proceedings IJCAI-89*, 517–523. Detroit: International Joint Conferences on Artificial Intelligence.

Sycara, K. 1989. Resolving Goal Conflicts via Negotiation. In *Proceedings AAAI-88*, 245–250. St. Paul: American Association for Artificial Intelligence.

Veloso, M. M. & Carbonell, J. G. 1991. Variable-Precision Case Retrieval in Analogical Problem Solving. In *Proceedings of the Case-Based Reasoning Workshop*, 93–106, Washington, DC: Defense Advanced Research Projects Agency – Information Science and Technology Office.

Vossos, G., Zeleznikow, J., Dillon, T. & Vossos, V. 1991. An Example of Integrating Legal Case Based Reasoning with Object-Oriented Rule-Based Systems: IKBALS II. In *Proceedings of the Third International Conference on Articial Intelligence and Law*, 31–41, Oxford, England.

Walker, R. F., Oskamp, A., Schrickx, J. A., van Opdorp, G. J. & van den Berg, P. H. 1991. Prolexs: Creating Law and Order in a Heterogeneous Domain. *International Journal of Man-Machine Studies* 35(1):35–68.

Waterman, D. A. & Peterson, M. 1980. Rule-Based Models of Legal Expertise. In *Proceedings AAAI-80*, Stanford, CA: American Association for Artificial Intelligence.

Waterman, D. A. & Peterson, M. 1981. Models of Legal Decisionmaking. Technical Report R-2717-1CJ, Santa Monica, CA: Rand Corporation.