Toward an Intelligent Tutoring System for Teaching Law Students to Argue with Cases

Kevin D. Ashley and Vincent Aleven
University of Pittsburgh
School of Law,
Intelligent Systems Program and
Learning Research and Development Center
Pittsburgh, Pennsylvania 15260

Abstract

This paper describes a research project to devise and test an intelligent, case-based tutorial program for teaching law students to argue with cases. In order to present pedagogically interesting lessons and develop a Student Model, we have designed memory structures such as Argument Contexts and a hierarchy of Issues in Case-Based Legal Reasoning. Using logical expressions in the knowledge representation language Loom, we also explicitly represent case-based argument concepts such as a case's being on point to a problem, more on point than another case, most on point of all the cases, a best case to cite, and a counterexample to another case. The program will be able to reason with the explicit concepts in selecting cases from a Case Library, assembling lessons and examples, analyzing student inputs, and in generating explanations and feedback. We hope to demonstrate empirically that, by providing law students a conceptual model of the criteria for selecting and describing precedents that would be useful in an argument, the tutorial program will help them to learn to select and apply cases more efficiently and to make more effective arguments.

1 Introduction to Arguing with Cases

This paper describes a research project to design and build a tutorial program to teach students to make arguments with cases. Under the program's guidance, students will argue with the program; the program will argue back, and comment on the student's arguments.

An important lawyering skill is the ability to compare and contrast cases critically for the purpose of using them in arguments by analogy. The law is rife with examples (e.g., If a defective coffee urn was held "inherently dangerous" and the manufacturer liable, then surely the manufacturer of a car with a defective wheel should be liable. [Levi, 1949]). Legal reasoning is not unique in this regard. The ability to compare and contrast problems and past cases is a mark of expert arguing in a variety of domain contexts including mathematics [Lakatos, 1976], scientific research, policy analysis [Neustadt and May, 1986], philosophy and ethics.

In the American legal system, and that of other common law jurisdictions (as distinguished from Continental or Civil Code jurisdictions), a standard warrant justifying a legal conclusion is to draw an analogy to past cases or precedents. A good precedent involves similar facts whose decision of an issue favored the adversary's side. In law as elsewhere, exactly what "similar" means is an important but debatable question with possibly many useful answers. An appropriate way to challenge the warrant is to show that there is a distinction, that is, a difference that justifies different treatment, or to find other analogous cases to cite as counterexamples.

Even when an attorney relies on the authority of a legal rule, whether drawn from a constitution, statute, regulation or court statement, he or she must know how to find favorable precedents, ones which applied the legal rule in similar circumstances, and to argue that they justify the same outcome in the current case. This is so because it is rare that attorneys can construct a logical proof that a legal term applies. Legal rules are not ordinary rules. They contain open textured legal predicates that are not adequately defined in terms of other rules. See [Gardner, 1987]. Legal concepts are also dynamic; their meanings change as they are applied from case to case [McCarty and Sridharan, 1981; McCarty and Sridharan, 1982]. Even logical connectors used in legal rules can be ambiguous, because their scopes may not be clearly specified [Allen and Saxon, 1987].

In order to bridge the gap between general legal rules and specific facts, attorneys must resort to comparing the problem to past cases. Recent work in AI and Law has adopted a number of approaches to modeling this process: In past work we have employed Dimensions and cases to link facts to conclusions about claims [Ashley, 1991a; Ashley, 1991b]. Rissland and Skalak have extended the approach with rules to link facts to conclusions about statutory rules and predicates [Rissland and Skalak, 1991], and Branting has employed matching facts to precedents' explanations to link the facts to conclusions about claims and statutory rules [Branting, 1991].

There are other ways to bridge the gap, for example, by arguing from the structure of a code of laws, the purposes or policies of the code, the intent of the framers of a rule,

This work is supported by a National Science Foundation Presidential Young Investigator Award and a grant from the National Center for Automated Information Retrieval.

or linguistic analysis. It would appear that in Civil jurisdictions, attorneys routinely employ these other methods in formal arguments. Nevertheless, one would expect that in informal arguments, expert attorneys (Civilians and common law attorneys alike) are equally adept at comparing and contrasting cases, recognizing that one situation presents a stronger or weaker position than another, distinguishing situations, and citing real or posing hypothetical counterexamples to test conclusions. Such cognitive abilities are universally valuable in formulating, assessing and explaining legal conclusions.

We believe that attorneys employ fundamental argument-making skills that involve the cognitive ability to compare problems and cases. We shall refer to these collectively as case-based argument-making abilities. They include:

a. Retrieving similar cases. Lawyers need to recognize the kinds of facts that strengthened or weakened a legal conclusion in a past case to assess whether the case would be a good case to cite as a justification for a legal conclusion in a current problem. Fundamentally, such factual strengths and weaknesses are the relevant similarities among the cases; the cases to look for in performing a search of legal databases and libraries are cases that share such strengths and weaknesses with the problem.

b. Analogizing problems to cases. In citing a case in support of a conclusion, lawyers need to be able to draw an analogy between the problem and the case that justifies treating the problem and the case in the same way. Reciting the shared strengths and weaknesses is a basic way of analogizing.

c. Distinguishing cases. Lawyers need to be able to respond to arguments citing cases. One way is by distinguishing the case from the problem. It involves identifying relevant differences between the problem and case, such as a strength favoring the winner in the case that is missing from the problem. Distinguishing also involves explaining why the differences are reasons for not treating the case and problem in the same way.

d. Identifying counterexamples. Another way of responding is by citing a counterexample. This means finding other cases that are just as anologous as or more analogous to the problem than a given case (i.e., share the same strengths and weaknesses and maybe some additional ones, too) but which had a different outcome.

e. Selecting best cases. Given a problem and a side in an argument to represent (i.e., plaintiff or defendant), lawyers need to be able to select among the relevant cases those that best support the side given the factual strengths and weaknesses of its position. Best case selection requires taking into account the analogies, distinctions and counterexamples.

f. Assessing argument strengths. The lawyer needs to assess how strong an argument his or her side has in light of the cases and counterexamples that can be cited for and against the proposition.

g. Generalizing a rule of classification. The classification rule to be generalized from the best cases attempts to differentiate between positive and negative examples and classifies the problem along with one or the other.

h. Testing rules with counterexamples (real or hypothetical).Testing that rule involves attempting to find counterexamples in the libraries of cases or to generate hypothetical cases which would demonstrate the rule's limitations, and suggesting appropriate reformulations of the rule.

2 Why Learning to Argue with Cases is Difficult

Currently, law school students learn the argumentation skills primarily by participating in classroom discussions in courses where teachers employ a Socratic method, of which making arguments by comparing and contrasting cases is an integral element. The practiced Socratic professor stage manages an argument to convey a lesson, perhaps to teach students about the meaning of a general legal rule or concept or the significance of a principle or policy. Significantly, the Socratic professor teaches students about abstract legal concepts within the context of the specific facts of a carefully crafted problem situation and a handful of cases from the readings. The professor lays bare an ambiguity in the rule or a conflict among principles or policies by inducing students to argue for or against a position in the problem situation.

Like Socrates' ancient listeners, law students are expected to participate in the argument, but, unlike the ancients, law students are expected to do more than say "yes" at the master's prodding. Law students respond by performing the basic case argument tasks listed above: selecting a case to cite, drawing an analogy in terms of some proposed classification rule, responding by distinguishing or citing a counterexample or offering a different rule. In order to steer the argument to serve pedagogical goals, the professor may hypothetically modify the problem's facts to test the sensitivity of the students' arguments to changes in various conditions.

A fascinating aspect of the law school Socratic method is that a small number of formal case-argument skills are applicable across a wide array of law school subjects. *If* a student has learned to participate in this adversarial dialogue, he or she is prepared to learn all manner of substantive legal areas. The same methodology basically applies across the curriculum (although, admittedly the "fit" is better with some law school subjects than others.) These legal argument skills "scale up" in another interesting sense. Although Socratic lessons may involve only a small number of cases, legal rules and alternative viewpoints, the argumentation skills they teach apply even in the most complex legal arguments. They also "scale down". One can restrict lawyers and law students to the task of making the best arguments possible given an artificially restricted collection of precedents, and they will make as good arguments as the precedents allow. The skills are still evident even though the problems are simplified.

Although Socratic lessons teach fundamental argument skills, law schools could do a better job of teaching law students to argue with cases [Paul, 1988; Jaff, 1986]. Only a few students get to engage in an argument during class, the law school classroom can be a stressful environment in which to learn the skills, and not all law school teachers are proficient at this method of teaching. Additionally, these important lawyering skills are not directly assessed in law schools. Typically, law school examinations do not directly measure argument-making skills.

Beyond lack of practice, law schools also do not provide students with an adequate conceptual model of arguing with cases. Students appear to lack a basic conceptual structure for making or responding to an argument citing a case. Lacking that, they are unable to translate how a case would be used in an argument into criteria for finding and evaluating relevant cases. Abstract descriptions of how to perform the skills, though helpful, are not enough; students need practice making and responding to arguments characterized in terms of a conceptual model.

3 An Intelligent Case-Based Tutor

We believe that an intelligent tutoring system could improve students' case argument-making ability by providing both a useful conceptual model and practice. (That computer-assisted instruction can be effective in the legal domain, at least more effective than group instruction, has been documented by [Teich, 1986].) We are building a tutorial program to teach students to make arguments that compare and contrast cases. More specifically, the system will teach a student (1) basic argument moves that apply cases in analyzing a problem, and (2) basic argument criteria that students can use to find the best cases to employ in those moves. The argument moves involve citing cases, distinguishing them, citing counterexamples, real or hypothetical, and asserting rule-like justifications for the significance of similarities and differences.

The system will (1) explain the argument moves and criteria for evaluating arguments, (2) provide the student with opportunities to practice his/her skills, and (3) provide the student with feedback on his/her problem solving activities. It will place the student in a game-like, adversarial context where students make and respond to arguments, and prompt the student to explore alternative interpretations of a problem's facts under competing rules or interpretations of rules. In responding to students' moves, the system will be able dynamically to determine the cases that are most relevant for effective moves and countermoves. The system will be able to select examples that serve pedagogical goals such as illustrating an explanation, or bring up issues that the student ought to know about.

The program will administer lessons testing and improving a law student's case-based argument-making ability. The lessons will comprise sets of materials, argument assignments and problem situations drawn from different substantive legal areas. The materials will provide a general introduction to the legal area and a selection of legal authorities including a set of legal cases involving a small set of issues. Each problem situation will describe a dispute between a plaintiff and defendant. The assignments will direct students to make arguments citing cases for or against a particular side in the problem or to defend against such arguments.

Examples of assignments include questions such as:

> Which cases can you cite in support of the plaintiff's claim?
>
> Which cases could the defendant cite?
>
> Which is the best case to cite for plaintiff?
>
> In citing the best case, which similarities between the case and the problem should you emphasize?
>
> How would you respond to that case on behalf of the defendant? Are there any relevant differences between the problem and the case?
>
> Can you find any counterexamples to the case?
>
> If you could make up a counterexample to cite on behalf of the defendant what facts would it include?

The system will engage the student in a competition, in effect, a mini argument. The tutor poses a problem, asks the student to make a point, response or rebuttal and offers a variety of possible moves. Some moves will be better than others in that there will be good responses to the less optimal moves. If the student chooses a less than optimal move, the system will make a strong response, illustrate the better moves by running the student through the argument and explain why the move is better by comparing the responses. The tutor will analyze the student's inputs to determine if the student understands the kind of case that would satisfy the argument move's constraints. The system will both explain the more effective query and provide an example of the best case or move.

In a first version of the system, the lessons will be based on the model of arguing with cases implemented in the HYPO program [Ashley, 1991a; Ashley, 1991b; Ashley, 1989a; Ashley and Rissland, 1988; Ashley and Rissland, 1987]. In that model, relevant similarities and differences among cases are represented by factors. Factors are collections of facts that make cases stronger or weaker for plaintiff. Each legal domain, like trade secrets misappropriation, has a set of factors that have been recognized by courts as relevant to a decision of such claims, and any problem can be expected to present a collection of factors some of which favor the plaintiff and others of which do not. HYPO provides a general mechanism for representing factors called Dimensions. HYPO makes and responds to arguments from competing viewpoints about who should win the dispute and poses hypotheticals to strengthen or weaken a side's position. On this model of legal argument, an arguer justifies that a plaintiff should win by citing precedents that involve the same collections of competing factors where the plaintiff also won. An opponent responds to such arguments by distinguishing the case or citing counterexamples. Distinguishing involves pointing out factors that the case and problem do not share which justify not treating the case and problem alike. Counterexamples include cases that involve the same or more inclusive sets of factors shared with the problem that have the opposite outcome. HYPO's model of arguing with cases and factors provides working definitions for such argument concepts as a case's being on point to a problem, more on point than another case, most on point of all the cases, a best case to cite, and a counterexample to another case. These argument concepts capture a realistic aspect of legal argument. HYPO's argument concepts make up the curriculum of the first version of our tutoring system.

The lessons will be administered by the tutor running on a personal computer or workstation (Our development work has been performed on a Microexplorer, a kind of Lisp machine consisting of a Macintosh II with a Texas

Instruments Explorer board.) Students will interact directly with a user interface that presents the problems and situations, enables students to browse through the materials, and solicits the students' responses. Students will not be able to write responses to the lesson assignments in natural language text. Instead, the students will construct their answers using tools provided by the interface. Menus of options will be used to constrain the range of student responses. Since students and the system will argue primarily by comparing case facts, the input / output characteristics of the program can be fairly constrained and yet still be interesting.

For each lesson, the cases and materials will reside in a Case Library. The cases will be based on real legal cases, but the opinions will not be reproduced. Instead, the cases' facts and the ultimate outcome will be described in a simplified manner (based on HYPO's Factual Predicates).

The interface will provide the student with a set of tools for retrieving cases from the library, selecting cases to cite, and emphasizing selected features of a case for the purpose of drawing analogies or distinctions between a cited case and a problem. Students will use a mouse to select cases from a list, to select elements of a case's description to emphasize as a relevant similarity or difference, or to select among various browsing and case retrieval tools (e.g., to show all the cases that were won by a plaintiff or that involved a particular factor.) The interface will provide a help facility, including examples, to assist students to learn to manipulate the tools.

4 Tutorial Program Architecture

The architecture of the system we are designing, shown in Figure 1, comprises four modules: the Pedagogical Module, Domain Expert, Student Model and Agenda Manager. (Of the standard tutoring system modules described in [Wenger, 1987], the User Interface is missing from our architecture. We regard its functions as part of the Pedagogical Module and Domain Expert.) In this section, we describe the architecture in overview and illustrate its intended performance with an example.

4.1 Architectural Overview

In the basic control loop of the tutor:

1. The Agenda Manager assigns a task to the Dialogue Manager. A typical task involves administering a lesson that advances the student through the curriculum.
2. The Dialogue Manager selects one or more Issues from the Student Model, selects a lesson that addresses the Issue(s), and poses a problem to the student.
3. With the aid of the Tool Kit, the student responds to the problem.
4. The Student Interpreter analyzes the response, updates the Student Model accordingly, and posts new task(s) to the Agenda Manager. The Agenda Manager may inspect the Student Model to propose new tasks. The cycle repeats with a new task and an up-to-date Student Model.

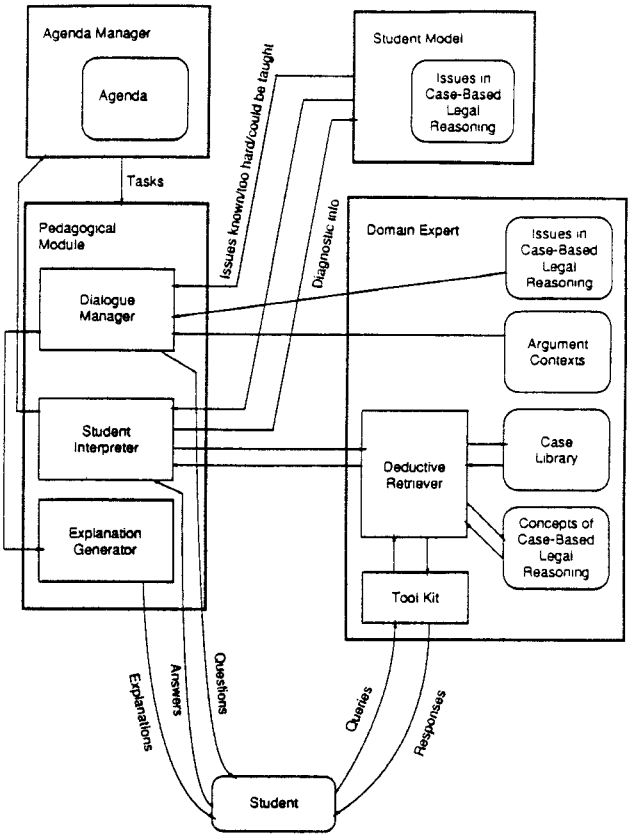

Figure 1: The architecture of the first phase of the tutorial system comprises four modules: the Student Model, Domain Expert, Pedagogical Module and Agenda Manager.

The curriculum is represented by a data structure called Issues in Case-Based Legal Reasoning (part of the Domain Expert's knowledge). The overall goal of the system is to make sure that the student knows all of the Issues. The idea of organizing the domain knowledge as a set of issues was taken from WEST [Burton and Brown, 1982]. The Issues are organized roughly in terms of conceptual complexity; the basic order of presentation proceeds from simple to complex Issues, but, subject to that general constraint, the order is determined dynamically. Some of the Issues are:

1. *Minimum criteria for citing a precedent* –A precedent c is *citable* for side s (plaintiff or defendant), if it was won by s, and shares at least one factor that favors s with the current fact situation.
2. *Prefer the more-on-point precedent* –If precedent c_1 and c_2 are both citable for the same side, and if c_1 is more on point than c_2, c_1 is better.
3. *Prefer the undistinguishable precedent* –If precedent c_1 and c_2 are both citable for the same side, and equally on point, and if c_1 is undistinguishable while c_2 is not, then c_1 is better.
4. *Prefer the untrumped point* –A point to which there is no response citing a more on point counterexample is better than a point to which there is.

Other Issues pertain to teaching the factors that influence the outcome of a trade secrets misappropriation case, concepts such as a case's being more on point than another, the possible ways of responding to a precedent-citing argument (by distinguishing the cited case or citing a counterexample), and evaluating overall argument strength. These Issues are drawn from HYPO's model of legal argument and its Argument Evaluation Criteria, the criteria HYPO implicitly implemented for selecting which points and responses to make and in evaluating competing arguments [Ashley, 1991a, Appendix G].

In each cycle, the Agenda Manager takes the task with the highest priority on the Agenda and hands it to the Dialogue Manager, which is part of the Pedagogical Module. The Pedagogical Module's task is to question the student, interpret the student's answers and fashion explanations. Let us suppose that the task is to advance the student through the curriculum (i.e. to bring up new Issues). In deciding which pedagogical Issues to pursue with the student, the Dialogue Manager consults the Student Model.

The Student Model keeps track of the parts of the curriculum that the student appears to have mastered. It is an "overlay" on the set of Issues. Roughly speaking, the Student Model records for each Issue whether the student knows it or not This means that at any time the student's knowledge is considered to be a subset of the Domain Expert's knowledge; the student's missing conceptions can be modelled, but not his/her misconceptions. The overlay model was developed in SCHOLAR [Carbonell, 1970] and WUSOR [Carr and Goldstein, 1977].

Having selected an Issue or Issues, the Dialogue Manager has to decide how to conduct a lesson that brings up the Issues. It selects an Assignment Type (see Section 3 for a listing of the different kinds of assignments) and an Argument Context. Argument Contexts consist of an ensemble of cases, factors and a problem situation (the current fact situation or "cfs"). They are assembled manually by the program designer from among cases in the program's Case Library. The various ensembles of cfs and cases raise pedagogically interesting Issues for the student who must choose among the cases in constructing arguments about the cfs. The Argument Contexts are indexed in a Context Library by the Issues raised.

The tutor presents the cases in the Argument Context to the student, and poses a question. The student performs his answers by thinking, reading the cases with the aid of browsing tools, and by employing retrieval tools in the Tool Kit to search for cases to use in responding to the tutor's questions.

In analyzing a response, the Student Interpreter draws on the Student Model and on the Domain Expert. Basically, the Student Interpreter compares the student's answers to those of the Domain Expert in light of the Issues that the student is presumed to know and not to know. Assessing whether the student's answer evidences mastery of an Issue presents a problem of credit assignment. Our tentative approach is illustrated in the example below. The Student Interpreter employs the Domain Expert's Deductive Retriever (described below) directly to determine the "correct" or best answer to the question that was posed to the student.

The Domain Expert provides optimal answers to the Dialogue Manager's questions, against which the student's answers are compared. In the first version of the system, the Domain Expert implements HYPO's model of case-based argumentation (see Section 5). Beside the Issues and the Argument Contexts, the Domain Expert comprises:

1. A Library of Cases; following HYPO, for each case the outcome and the factors that apply are represented;
2. A set of Concepts of Case-Based Legal Reasoning. Each concept is a relation, defined over cases or factors in terms of a logical expression, that plays a role in legal argument. They include relevantly similar, relevant difference, more on point, most on point, and trumping counterexample.
3. A Deductive Retriever that accepts the logical definition of a concept (or some composition of such definitions) and returns all of the items, cases or factors (i.e., Dimensions), in the Case Library that satisfy the specifications.
4. A Tool Kit of tools associated with the various concepts by which the student can analyze the Argument Context he is working in.

If the student's answers evidence a mastery of the Issue, the Student Interpreter informs the Student Model accordingly. If not, the Interpreter generates additional tasks to provide feedback and explanations. The updated Student Model will lead the Dialogue Manager to revert to more basic Issues, reinforce a lesson with additional exercises, or proceed to more advanced Issues.

4.2 Example of Target Input/Output Behavior

Here is an example of the kind of input/output behavior we intend the tutor to support. The example is schematic to focus on the kind or reasoning we would like the tutor to perform; currently, the system does not generate this I/O.

Let us assume that the Agenda Manager has passed on to the Dialogue Manager the task of advancing to new Issues. The Dialogue Manager has to choose a lesson that brings up Issues that are beyond, but not too far beyond, the current scope of the student's knowledge. It consults the Student Model to find out which Issues are already "known" by the student, which are currently "too hard", and which Issues "could be taught". Assume that the Student Model informs the Dialogue Manager that the Issue *Minimum criteria for citing a precedent"* (Issue 1) is "known", that the Issue *Prefer the undistinguishable precedent* (Issue 3) is "too hard", and that the Issues *Prefer the more on point precedent* (Issue 2) and *Prefer the untrumped point* (Issue 4) "could be taught". The Dialogue Manager selects a lesson that is appropriate to bring up the selected "could be taught" Issues. It decides to assign the student the task of selecting from among a number of precedents the one that is best to cite, and retrieves an Argument Context.

The Argument Context contains four factors, three of which favor plaintiff (namely, f1, f2, and f3), and one of which favors defendant (f4). Furthermore, the Argument

Context contains a current fact situation, cfs, and four precedent cases (c1, c2, c3 and c4). The cases, their outcome, and the factors that apply in each are:

Case	Factors	Outcome
cfs	f1, f2, f3, f4	n/a
c1	f1	plaintiff
c2	f2, f3, f4	plaintiff
c3	f2, f4	plaintiff
c4	f1, f4	defendant

For clarity, we present the outputs schematically in terms of variables standing for the cfs, cases, and factors. In the actual outputs, the system would refer to cases by names, to factors by descriptive phrases, and would provide students with tools for reading complete descriptions of cases and factors.)[1]

The *Claim Lattice* of this Argument Context is shown in Figure 2. A Claim Lattice is a graph that represents the ordering, in terms of on-pointness, of a given set of cases. (A case x is *more on point* than a case y, with regard to a current fact situation *cfs*, if the *relevant similarities* of y –i.e., the factors that y shares with *cfs*– form a proper subset of the relevant similarities of x.) Each node of the Claim Lattice has a factor list and a case list. Each node represents a group of cases that are all equally on point, meaning that they all have the same relevant similarities. The relevant similarities of a case are contained in the factor list of the node in which the case resides. The root node's case list contains the cfs (and perhaps other cases), its factor list contains the factors that apply in the cfs. A link between two nodes indicates that the factor list at the end of the link is a subset of the factor list at the origin of the link. Therefore, the links of the graph represent the more on point relations that exist among the cases; case x is more on point than case y if and only if there is a path from x to y. One can infer from the Claim Lattice of Figure 2, for instance, that the relevant similarities of c3 are f2 and f4, that c2 is more on point than c3 (notice that c3's relevant similarities form indeed a proper subset of c2's relevant similarities, as is required by the definition of more on point), and that c2 is not more on point than c4 or c1.

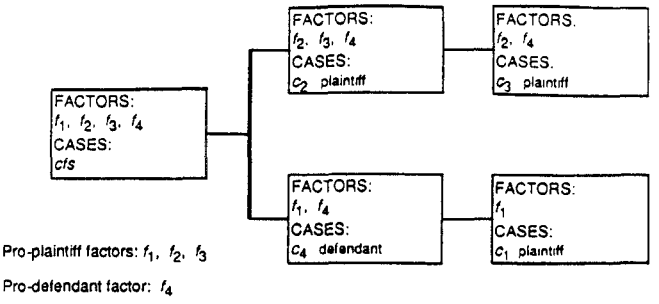

Figure 2: The Claim Lattice of the Argument Context

The lesson that the tutor selected is pedagogically interesting, because the task of selecting the best case to cite within the selected Argument Context involves the Issues that the tutor decided to focus on (Issues 2 and 4), but does not bring up the Issue that the tutor wants to avoid (Issue 3). This is a result of the particular relations that exist among the cases in the Argument Context.

The precedents c1, c2 and c3 are all *citable* for plaintiff, meaning that they satisfy the *Minimum criteria for citing a precedent* (Issue 1), described in the previous section. They are citable because they were won by the plaintiff, and share at least one pro-plaintiff factor with the cfs. Plaintiff can make a reasonable argument citing any of these cases. However, precedent c2 is plaintiff's best precedent. Let us see why. First, c2 is better than c4. Since c4 was won by the defendant, it is not citable for the plaintiff. Since c2 is, it is better than c4.

Second, c2 is better than c1. Notice that c4 is a *trumping counterexample* to c1. This is so because it (c4) was won by the opposing side (c1 was won by plaintiff, c4 was won by defendant) and, as is clear from the Claim Lattice, it is more on point than c1. It follows that defendant can trump a point citing c1 by citing c4 as a counterexample. A point citing c2, however, cannot not be trumped, since the Argument Context does not contain a trumping counterexample to c2. According to Issue 4 (*Prefer the untrumped point*), then, c2 is a better precedent than c1.

Finally, c2 is better than c3. As is evident from the Claim Lattice, c2 is more on point than c3. According to Issue 2 (*Prefer the more on point precedent*), c2 is better.

To summarize, in order to find plaintiff's best case in this Argument Context, one has to (1) recognize that c1, c2, and c3 but not c4 are citable for the plaintiff (Issue 1), (2) recognize that c4 is a trumping counterexample for c1, and apply Issue 4 to conclude that c2 is a better precedent for plaintiff than is c1, and (3) recognize that c2 is more on point than c3 and apply Issue 2 to conclude that c2 is better than c3[2]. Also, although the discussion didn't focus on this, an important aspect of almost any argumentation exercise (and certainly the exercise of finding plainitff's best case to cite in the current Argu-

[1]The text of the cfs is: *America's Best Computers Corp. vs. Digi General, Inc.* ABC Corp. manufactured the Super A 1200 minicomputer. Digi General ordered a Super A 1200 from a third party supplier. From the supplier it also obtained a maintenance manual with design drawings, which it copied and returned. Even though the drawings bore a mark prohibiting copying, and the sales contract prohibited use of the drawings for manufacturing, Digi General used the drawings to produce its D-116 minicomputer. By using the drawings, it was able to save 50% development time, compared to the time it took ABC to develop the Super A 1200. ABC distributed the same drawings, bearing the same restrictive legend,to 600 customers, users, vendors and trainees. ABC's sales contracts prohibited the use of the drawings for manufacturing. ABC took additional security measures, including plant security.

The factors that apply, and their corresponding names used in the example, are:

f1: competitive-advantage-gained
f2: disclosures-subject-to-restriction
f3: security-measures-adopted
f4: secrets-disclosed-outsiders

[2]We are interested in exploring the possibility of the system's automatically assembling Argument Contexts from cases in the Case Library. It could follow a generate-and-test approach, generating and testing collections of cases until one was found that allowed the "could be taught" Issues, and disallowed the "too hard" Issues. Care should be taken to adequately constrain the generator.

ment Context) is recognizing which factors apply in the cfs and the precedent cases. This information will not be given to the student; the student has to infer it from a textual description of the cases. Moreover, the Claim Lattice will not be presented to the student.

Notice that the Argument Context does not raise the forbidden Issue 3 *Prefer the undistinguishable precedent* (Issue 3), because there are no equally on point precedents in the Argument Context. This is no coincidence. The Argument Context was selected (among other reasons) because it avoids this Issue.

The Dialogue Manager then presents the Argument Context to the student, assigns him as attorney to one of the sides, and asks him to select a case that makes a good point. The student will be able to use the Tool Kit to help him analyze a problem. Selecting a browsing tool "TEXT", the student will be able to read cases in a text window. A tool called "PRO-PLAINTIFF-CASES" returns the cases that favor the plaintiff on a claim. The student can also use tools to verify his conclusions about the cases in the Argument Context: By selecting "FACTORS c f_1 ... f_n", the student can verify that the listed factors f_1 through f_n apply in case c. The tutor will simply respond with "good" or "no". Using the tool "RELEVANT SIMILARITIES c_1 c_2 f_1 ... f_n", the student can verify that the listed factors form the relevant similarities between case c_1 and c_2. In the example, the following dialogue might take place:

> Tutor: The current fact situation is: cfs. Suppose you are plaintiff's attorney. Which of the following cases would you cite: c1, c2, c3 or c4?
> Student: TEXT cfs
> Tutor: [Presents a textual description of cfs]
> Student: FACTORS cfs f1 f2
> Tutor: Good
> Student: FACTORS cfs f3 f4
> Tutor: Good
> Student: PRO-PLAINTIFF-CASES
> Tutor: c1 c2 c3
> Student: TEXT c3
> Tutor: [Presents a textual description of c3]
> Student: FACTORS c3 f2 f4
> Tutor: Good
> Student: RELEVANT SIMILARITIES c3 cfs f2 f4
> Tutor: Good
> Student: CITE c3
> Tutor: Yes, citing c3 would make a good argument, since your opponent cannot respond by citing a more-on-point counterexample. There is, however, an even better precedent that you can cite. Which precedent is that?

The Student Interpreter receives the student's answer (in this case, "CITE c3"). Its task is to analyze the student's answer, update the Student Model accordingly, and decide what feedback and perhaps follow-up questions are appropriate. It will propose new tasks to the Agenda Manager that achieve the feedback and follow-up.

Since the Student Model is an overlay on the set of Issues, the analysis of the student's answer is stated in terms of Issues. In other words, the Student Interpreter looks for evidence that the student knows, or doesn't know, particular Issues. In general, the analysis assumes that the student knows the Issues involved in rejecting moves that are inferior to the move that he chose, but doesn't know the Issues that are involved in selecting superior moves. This approach was taken in WUSOR [Goldstein, 1982]. The Domain Expert is enlisted to determine which of the alternative moves are better and which ones are worse, and to analyze which Issues are involved in recognizing the worse moves as being worse, and the better moves as being better.

Let us see how the Student Interpreter, in cooperation with the Domain Expert, could carry out such an analysis. Of the four cases in the Argument Context, c4 doesn't satisfy the minimum criteria for citing a precedent on behalf of the plaintiff because it was won by the defendant (Issue 1). The fact that the student didn't pick this case can be taken as evidence that he knows Issue 1. (One cannot be absolutely sure, however, that the student knows the Issue. It could be that he simply looked at case c3 first, found that this case was acceptable for him, and didn't come to look at case c4 at all.) The fact that the student selected case c3 instead of the more-on-point case c2 is evidence that he doesn't know Issue 2, because this Issue tells him to prefer the more-on-point precedent. The fact that the student selected case c3, and not case c1, for which there is a trumping counterexample, is evidence that he knows Issue 4 (*Prefer the untrumped point*).

After analyzing the student's answer, the Student Interpreter informs the Student Model of its findings. This may cause the labels ("known", "could be taught", or "too hard") of some of the Issues to change. The feedback that is appropriate depends on the diagnosis, and also on the Student Model. The first conclusion was that the student applied Issue 1. Since Issue 1 was one of the "known" Issues, according to the Student Model, the Student Interpreter decided that no feedback related to this conclusion was necessary.

The second conclusion was that the student applied Issue 4. Since this Issue was not "known", some positive feedback is necessary. The feedback that is appropriate is: "Citing c3 would make a good argument, since your opponent cannot respond by citing a more on point counterexample." The Student Interpreter therefore posts a task to the Agenda Manager to generate this feedback.

The third conclusion was that the student doesn't know Issue 2. It seems reasonable that the system's response should focus on this lack of knowledge. The Student Interpreter therefore proposes a task to the Agenda Manager to discuss that c2 is a better precedent than c3. The tutor starts executing this task by saying: "There is, however, an even better precedent that you can cite. Which precedent is that?"

In explaining its advice, the tutorial will provide a combination of conceptual explanations, applied definitions and, where possible, an example of how an opponent would respond to the student. Thus, in explaining why c2 is better than c3, the system (the Explanation Generator) would offer a conceptual explanation and an applied definition:

> Tutor: (definition as applied:) c2 is a better precedent to cite for the plaintiff than c3 because c2 is *more on point* to the cfs.
>
> (conceptual explanation:) In general, if a precedent is more on point than another precedent, the more-on-point precedent is better.
>
> (definition as applied:) c2 is *more on point* than c3 relative to the cfs because c2 shares every factor with the cfs that c3 does (namely f2 and f4) but also a factor that c3 does not share with the cfs: f3.

In some instances, the tutor will be able to provide another kind of feedback explanation; it will argue back. For instance, if the student had chosen to cite c1, the tutor would also be able to argue back at the student:

> If you cite c1 for the plaintiff, defendant will trump your point by citing c4 as a more on point or *trumping counterexample.*

5 Explicitly Representing Case Argument Concepts

A central element of the design of the tutorial program involves making explicit the representations of the core argument concepts and relations. These argument concepts are closely related to the Issues that make up the program's curriculum, and serve as a basis of the system's interpretation of student responses, and subsequent generation of feedback and explanations. We have represented these argument concepts and relations with logical expressions in the knowledge representation language Loom. Defining the concepts expressly should facilitate the tutor's ability to explain and apply the concepts. As is illustrated below, the explicit definitions of the argument concepts can be used to perform many interesting inferences.

Loom is a structured inheritance system, or KL-ONE-style system [MacGregor, 1988; Woods and Schmolze, 1990]. One can view Loom as a deductive retriever that manages a database of propositions, and is able to perform deductive queries on this database. Before propositions can be asserted into the database, a vocabulary needs to be defined. The vocabulary consists of *concepts* and *relations*; roughly speaking, concepts correspond to the unary predicates of first-order logic, and relations to n-ary predicates[3]. Loom has a vast number of definition constructs, including the facility to state definitions in a language that is close to first-order logic. A definition usually states the necessary and sufficient conditions for the given concept or relation.

Using Loom, we expressed definitions for the basic concepts of the HYPO model, namely **Case**, **Factor**, and **Side**, and also for the basic relations: a case's being won by a certain side (**outcome**), a factor's favoring a particular side (**favors**), and a factor's applying to a particular case (**applicable-factor**). These concepts and relations are the "primitives" of our knowledge base, meaning that it cannot be inferred from other facts in the database whether they apply in a particular situation or not. Primitive concepts and relations apply only when this has been explicitly asserted.

Using these primitive concepts and relations as building blocks, we expressed definitions for many argument concepts in Loom. For instance, here, using Loom, are the definitions for **shared-factor** and **more-on-point**:

```
(defrelation shared-factor
  :domains (Case Case)
  :range Factor
  :is (:satisfies (?c1 ?c2 ?f)
        (:and (Case ?c1)
              (Case ?c2)
              (Factor ?f)
              (applicable-factor ?c1 ?f)
              (applicable-factor ?c2 ?f)))
  :attributes :multiple-valued)

(defrelation more-on-point
  :domains (Case Case)
  :range Case
  :is (:satisfies (?c1 ?c2 ?cfs)
        (:and (Case ?c1)
              (Case ?c2)
              (Case ?cfs)
              (:for-all ?f
                 (:implies
                    (:and (Factor ?f)
                          (shared-factor ?c2 ?cfs ?f))
                    (applicable-factor ?c1 ?f)))
              (:for-some ?f
                 (:and (Factor ?f)
                       (shared-factor ?c1 ?cfs ?f)
                       (:not (applicable-factor ?c2 ?f))))
  :attributes :multiple-valued)
```

These definitions state the sufficient conditions for the relations **shared-factor** and **more-on-point**. In English the first definition says: "a Factor f is a shared factor of Cases c1 and c2 if it applies in both c1 and c2". The second definition says, in effect: "a Case c1 is more on point than a Case c2 with respect to a Case cfs (typically the Case that represents the current fact situation), if the factors that are shared between c2 and cfs form a proper subset of the Factors shared between c1 and cfs)".

As the example illustrates, more complex definitions are composed of the simpler ones. The definition of **shared-factor** is built out of the primitive concepts **Case** and **Factor**, and the primitive relation **applicable-factor**. The relation **more-on-point**, in turn, employs the relation **shared-factor**, as well as the primitive relation **applicable-factor**. The relation **most-on-point**, whose definition is is shown in the appendix, refers to **more-on-point**. The definition of this relation can be stated in English as: "A Case c is most on point, with respect to the current fact situation cfs, if no Case is more on point."

The appendix shows the Loom representation for other relations including relevantly-similar, relevant-difference, citable, best-case-to-cite, trumping-cex (trumping or more on point counterexample) and untrumped-best-case. These relations are the same ones discussed in [Ashley, 1991a; Ashley, 1991b; Ashley, 1989b]. Formerly, however, these relations were represented only procedurally in HYPO, not in a form in which a program could manipulate or explain them.

[3] Alternatively, and more true to Loom's heritage, one could liken Loom to a semantic network system, and think of the concepts as nodes, and of relations as links between nodes.

After the concepts and relations of an application area have been defined, facts (propositions) can be asserted using the Loom command **tellm**. A fact either expresses that a certain individual is an instance of a certain concept, or that a certain relation holds among two or more individuals. For instance, to represent the factor *Competitive-Advantage-Gained*, which favors the plaintiff, we created an individual named **f8** by the following command:

```
(tellm (:about f8
    Factor
    (factor-name "COMPETITIVE-ADVANTAGE-GAINED")
    (favors plaintiff)
  ))
```

As a result of this command, Loom stores the new Factor, **f8**, and the associated information in its database. All Dimensions (factors) related to trade secrets misappropriation, HYPO's original domain, were represented in this manner. We also created individuals to represent the cases in HYPO's Case Knowledge Base, one individual for each case. For example, the *Analogic* case, which was won by plaintiff, and in which *Competitive-Advantage-Gained* and two other factors apply, was represented as follows:

```
(tellm (:about case7
    Case
    (case-name
       "Analogic Corp. v. Data Translation, Inc.")
    (outcome PLAINTIFF)
    (:filled-by applicable-factor f4 f7 f8)
  ))
```

Loom's query language allows one to retrieve any asserted fact, and any logical consequence of the asserted facts and the definitions[4]. For example, if the database contains the facts that **cfs** is a Case in which the Factors **f2**, **f4** and **f7** apply, then the query

```
(retrieve ?f (shared-factor case7 cfs ?f))
```

returns **f4** and **f7**, as one would expect. If it has been asserted that **case23** is a Case in which only Factor **f4** applies, then the query

```
(ask (more-on-point case7 case23 cfs))
```

returns **T**, since **case7** is more on point than **case23** with respect to current fact situation **cfs**; the factors that **case23** shares with **cfs** (namely, **f4**) form a proper subset of the factors that **case7** shares with **cfs** (namely, **f4** and **f7**). To retrieve plaintiff's best cases to cite (finding the best cases to cite for each side in the dispute is one of HYPO's key operations), one simply poses the query:

```
(retrieve ?c (best-case-to-cite ?c cfs plaintiff))
```

Explicitly representing the argument concepts and relations in Loom will be useful in (1) designing retrieval tools, (2) constructing Argument Contexts, and (3) explaining concepts. The concept definitions can be conceptualized as filters which may be applied to the Case Library to retrieve only those cases that pass through the filter (i.e., satisfy the tool's conceptual definition.)

Using the concepts, one can implement retrieval tools that students can use to verify their conclusions as to how the concepts apply to the current Argument Context. Using the tools, the student can ask for a "yes/no answer" as to whether a particular argument concept applies to a particular set of cases and/or factors. As the example of Section 4.2 illustrated, students will have access to a Tool Kit including retrieval tools like PRO-PLAINTIFF-CASES, PRO-DEFENDANT-CASES, FACTORS, PRO-PLAINTIFF-FACTORS, AND PRO-DEFENDANT-FACTORS. As they gain proficiency with the more advanced concepts, students will be able to use tools like CITABLE, RELEVANTLY-SIMILAR, RELEVANT-DIFFERENCE, SHARED-PRO-WINNER-FACTOR, SHARED-PRO-LOSER-FACTOR, MORE-ON-POINT, MOST-ON-POINT-FOR-SIDE, BEST-CASE-TO-CITE, TRUMPING-CEX, UNTRUMPED-BEST-CASE. These tools are readily implemented by means of simple expressions in Loom's query language. We anticipate that these tools will be represented graphically by icons (as in an Apple Macintosh or Windows 3 style user interface) and that students will be able to select tools by mouse to search the Case Library.

Retrieval tools like these, implemented with argument concepts and relations represented in Loom, are also useful for the system designers because it allows us to construct Argument Contexts easily and to perform a large variety of inferences relating to these Argument Contexts. For example, in order to generate an Argument Context containing real cases from the program's Case Library with which one could teach the sample lesson of Section 4.2, we constructed a query like the following (The actual query is written in Loom and expressly refers to a number of the concepts defined above and in the Appendix such as **more-on-point** and **citable**):

Retrieve all cases (cfs c1 c2 c3 c4) such that:

- c1, c2 & c4 are pro-plaintiff, c3 is pro-defendant
- c1, c2 & c4 are citable for plaintiff
- c1 is more on point than c2 relative to cfs
- c3 is more on point than c4 relative to cfs
- neither c1 nor c3 has all the factors that apply to cfs
- neither c1 and c4 are as, less, or more on point than each other
- neither c2 and c3 are as, less, or more on point than each other

A set of five cases satisfying these constraints would raise all of the pedagogically interesting issues of the Argument Context in Section 4.2. With twenty cases in the Case Library, the Deductive Retriever generated eight Argument Contexts in 3 minutes and 10 seconds. Four of the Argument Contexts involved a real case called *Structural Dynamics* as cfs; four involved the *Telex v. IBM* case as cfs. Each Argument Context had four other real cases that satisfied the various constraints described above. We plan to use these Argument Contexts as the basis of some tutorial sessions in which we "manually" teach law students to argue with cases. It should be noted that it is a difficult and time consuming mental exercise for humans to come up with such pedagogically interesting Argument Contexts from a collection of twenty cases. In addition, this kind of inference, where the cfs as well as the cases are not specified beforehand, would be impossible for HYPO to perform.

We anticipate that the explicit representation of the concepts and relations involved in case-based argument also will be useful for the purpose of generating explanations. For example, from the Loom definition of **more-on-point**, generating the applied definition in the sample output explaining why c2 is more on point than

[4]This is an oversimplification. Loom does not deduce all logical consequences.

c3 is fairly straight forward. We hope, too, that Loom will help in devising conceptual explanations such as the one above.

6 Planned Evaluation and Experiments

We believe that the tutorial program will be an effective teaching tool because it incorporates HYPO's conceptual model of case-based argument in which the relevance of cases is closely linked to their potential uses in an argument. Using the tutorial program, students can explicitly relate their needs in making an argument to queries for relevant cases. This is more difficult when students use, for example, full text retrieval services like Lexis and Westlaw where the relationship of the key word queries to a student's legal argument needs is much more obtuse.

We intend to evaluate the developing system by running experiments on a variety of problems with actual law school students. (See [Littman and Soloway, 1988] for a discussion of evaluating tutoring systems.) We believe that law students' abilities to making case-based arguments will be measurable through their performance with the case argument tutor. Using Student Models, we will be able to measure how far students get in mastering the curriculum, that is, how far they get in satisfying the assignments of graduated difficulty. It will also be possible to measure the time that it takes students to master particular argument skills or the skills associated with the most complex argument concepts and the directness of the paths they take through the curriculum until they can demonstrate proficiency. Using the tutorial program, it will be possible to compare experts' and novices' abilities to make arguments effectively and efficiently. We hope to demonstrate that practice with the tutorial program is an effective way of improving novice law students' abilities to argue with cases.

7 Conclusion

This paper describes a research project to devise and test an intelligent, case-based tutorial program for teaching law students to argue with cases. In order to represent pedagogically interesting lessons and develop a Student Model, we have designed memory structures such as Argument Contexts and a hierarchy of Issues in Case-Based Legal Reasoning. Using logical expressions in the knowledge representation language Loom, we also explicitly represent case-based argument concepts such as a case's being on point to a problem, more on point than another case, most on point of all the cases, a best case to cite, and a counterexample to another case. The program will be able to reason with the explicit concepts in selecting cases from a Case Library, assembling Argument Contexts and examples, analysing student inputs, and in generating explanations and feedback. We hope to demonstrate empirically that, by providing law students a conceptual model of the criteria for selecting and describing precedents that would be useful in an argument, the tutorial program will help them to learn to select and apply cases more efficiently and to make more effective arguments.

References

[Allen and Saxon, 1987] Layman E. Allen and Charles S. Saxon. Some Problems in Designing Expert Systems to Aid Legal Reasoning. In *First International Conference on Artificial Intelligence and Law*, Northeastern University, Boston, 1987.

[Ashley and Rissland, 1987] Kevin D. Ashley and Edwina L. Rissland. Compare and Contrast, A Test of Expertise. In *Proceedings AAAI-87*. Seattle, WA, August 1987.

[Ashley and Rissland, 1988] Kevin D. Ashley and Edwina L. Rissland. Waiting on Weighting: A Symbolic Least Commitment Approach. In *Proceedings AAAI-88*. St. Paul, MN, August 1988.

[Ashley, 1989a] Kevin D. Ashley. Defining Salience in Case-Based Arguments. In *Proceedings IJCAI-89*. Detroit, MI, August 1989.

[Ashley, 1989b] Kevin D. Ashley. Toward a Computational Theory of Arguing with Precedents: Accommodating Multiple Interpretations of Cases. In *Second International Conference on Artificial Intelligence and Law*, University of British Columbia, Vancouver, BC, 1989.

[Ashley, 1991a] Kevin D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. MIT Press, Cambridge, 1991. Based on Ashley's 1987 PhD. Dissertation, University of Massachusetts, COINS Technical Report No. 88-01.

[Ashley, 1991b] Kevin D. Ashley. Reasoning with Cases and Hypotheticals in HYPO. *International Journal of Man-Machine Studies*, 1991.

[Branting, 1991] L. Karl Branting. Building Explanations from Rules and Structured Cases. *International Journal of Man-Machine Studies*, 1991.

[Burton and Brown, 1982] R.R. Burton and J.S. Brown. An Investigation of Computer Coaching for Informal Learning Activities. In D. Sleeman and J.S. Brown, editors, *Intelligent Tutoring Systems*, pages 79–98. Academic Press, London, 1982.

[Carbonell, 1970] Jaime R. Carbonell. AI in CAI: an Artificial Intelligence approach to Computer-Assisted Instruction. *IEEE Transactions on Man-Machine Systems*, 11(4):190–202, 1970.

[Carr and Goldstein, 1977] B. Carr and I.P. Goldstein. Overlays: a Theory of Modeling for Computer-Aided Instruction. Technical Report AI Lab Memo 406, Massachusetts Institute of Technology, Cambridge, MA, 1977.

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

[Goldstein, 1982] I.P. Goldstein. The Genetic Graph: a Representation for the Evolution of Procedural Knowledge. In D. Sleeman and J.S. Brown, editors, *Intelligent Tutoring Systems*, pages 51–78. Academic Press, London, 1982.

[Jaff, 1986] Jennifer Jaff. Frame-Shifting: An Empowering Methodology for Teaching and Learning Legal Reasoning. *Journal of Legal Education*, 35:249–267, 1986.

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

[Levi, 1949] Edward H. Levi. *An Introduction to Legal Reasoning*. University of Chicago Press, 1949.

[Littman and Soloway, 1988] David Littman and Elliot Soloway. Evaluating ITSs: The Cognitive Science Perspective. In Martha C. Polson and J. Jeffrey Richardson, editors, *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[MacGregor, 1988] Robert M. MacGregor. A Deductive Pattern Matcher. In *Proceedings AAAI-88*, pages 403–408, Saint Paul, MN, August 1988.

[McCarty and Sridharan, 1981] L. Thorne McCarty and N. S. Sridharan. The Representation of an Evolving System of Legal Concepts: II. Prototypes and Deformations. In *Proceedings IJCAI-81*, Vancouver, BC, August 1981.

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

[Neustadt and May, 1986] R. E. Neustadt and E. R. May. *Thinking in Time*. Free Press, New York, 1986.

[Paul, 1988] Jeremy Paul. A Bedtime Story. *Virginia Law Review*, 74:915–934, 1988.

[Rissland and Skalak, 1991] Edwina L. Rissland and David B. Skalak. CABARET: Statutory Interpretation in a Hybrid Architecture. *International Journal of Man-Machine Studies*, 1991.

[Teich, 1986] Paul F. Teich. Research on American Law Teaching: Is there a Case against the Case System? *Journal of Legal Education*, 35:167–188, 1986.

[Wenger, 1987] Etienne H. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufmann Publishers, San Mateo, CA, 1987.

[Woods and Schmolze, 1990] William A. Woods and James G. Schmolze. The KL-ONE Family. Technical Report TR-20-90, Center for Research in Computing Technology, Harvard University, Cambridge, MA, August 1990.

Appendix – Definitions of Concepts of Case-Based Legal Reasoning

```
(defrelation relevantly-similar
  :domain Case
  :range Case
  :is (:satisfies (?c1 ?c2)
       (:and (Case ?c1)
             (Case ?c2)
             (:for-some ?f
               (:and (Factor ?f)
                     (shared-factor ?c1 ?c2 ?f)))))
  :attributes :multiple-valued)

(defrelation citable
  :domains (Case Case)
  :range Side
  :is (:satisfies (?c ?cfs ?s)
       (:and (Case ?c)
             (Case ?cfs)
             (Side ?s)
             (outcome ?c ?s)
             (:for-some ?f
               (:and (Factor ?f)
                     (favors ?f ?s)
                     (shared-factor ?c ?cfs ?f)))))
  :attributes :multiple-valued)


(defrelation relevant-difference-1
     ; pro-loser factor that applies
     ; only to the current fact situation
  :domains (Case Case)
  :range Factor
  :is (:satisfies (?c ?cfs ?f)
       (:and (Case ?c)
             (Case ?cfs)
             (Factor ?f)
             (opposite (favors ?f)
             (outcome ?c))
             (applicable-factor ?cfs ?f)
             (:not (applicable-factor ?c ?f))))
  :attributes :multiple-valued)
```

```
(defrelation as-on-point
  :domains (Case Case)
  :range Case
  :is (:satisfies (?c1 ?c2 ?cfs)
       (:and (Case ?c1)
             (Case ?c2)
             (Case ?cfs)
             (same-as (shared-factor ?c1 ?cfs)
                      (shared-factor ?c2 ?cfs))))
  :attributes :multiple-valued)


(defrelation most-on-point
  :domain Case
  :range Case
  :is (:satisfies (?c ?cfs)
       (:and (Case ?c)
             (Case ?cfs)
             (:for-all ?c1
               (:implies
                 (:and (Case ?c1)
                       (nequal ?c1 ?cfs))
                 (:not (more-on-point ?c1 ?c ?cfs))))))
  :attributes multiple-valued)


(defrelation most-on-point-for-side
  :domains (Precedent Case)
  :range Side
  :is (:satisfies (?c ?cfs ?side)
       (:and (Precedent ?c)
             (Case ?cfs)
             (Side ?side)
             (outcome ?c ?side)
             (neq ?c ?cfs)
             (:for-all ?c1
               (:implies
                 (:and (Case ?c1)
                       (outcome ?c1 ?side)
                       (neq ?c1 ?cfs))
                 (:not (more-on-point ?c1 ?c ?cfs))))))
  :attributes :multiple-valued)


(defrelation best-case-to-cite
  :domains (Case Case)
  :range Side
  :is (:satisfies (?c ?cfs ?side)
       (:and (Case ?c)
             (Case ?cfs)
             (Side ?side)
             (:for-some ?f
               (:and (Factor ?f)
                     (favors ?f ?side)
                     (shared-factor ?c ?cfs ?f)))
             (most-on-point-for-side ?c ?cfs ?side)))
  :attributes multiple-valued)


(defrelation untrumped-best-case
  :domains (Case Case)
  :range Side
  :is (:satisfies (?c ?cfs ?side)
       (:and (Case ?c)
             (Case ?cfs)
             (Side ?side)
             (best-case-to-cite ?c ?cfs ?side)
             (:for-all ?cex
               (:implies (Case ?cex)
                         (:not (trumping-cex ?cex ?c ?cfs))))))
  :attributes :multiple-valued)


(defrelation trumping-cex
  :domains (Case Case)
  :range Case
  :is (:satisfies (?cex ?c ?cfs)
       (:and (Case ?cex)
             (Case ?c)
             (Case ?cfs)
             (opposite (outcome ?cex)
                       (outcome ?c))
             (more-on-point ?cex ?c ?cfs)))
  :attributes :multiple-valued)
```