Issue Spotting in a System for Searching Interpretation Spaces

Thomas F. Gordon*
German National Research Center for Computer Science (GMD)
Institute for Applied Information Technology
Expert Systems Research Group
Sankt Augustin, Federal Republic of Germany
thomas@gmdzi.uucp

Abstract

A method for spotting issues is described which uses a system we are developing for searching interpretations spaces and constructing legal arguments. The system is compatible with the legal philosophy known as legal positivism, but does not depend on its notion of clear cases. AI methods applied in the system include an ATMS reason maintenance system, Poole's framework for default reasoning, and an interactive natural deduction theorem prover with a programmable control component for including domain-dependent heuristic knowledge. Our issue spotting method is compared with Gardner's program for identifying the hard and easy issues raised by offer and acceptance law school examination questions.

1 Introduction

Anne Gardner's program for spotting the issues raised by offer and acceptance examination questions [Gardner 87] was innovative in that it was one of the first legal reasoning programs to be based on legal postivism, rather than the discredited theory of mechanical jurisprudence. The state of the art in AI has progressed considerably since 1984, when Gardner completed her dissertation, and there now are theoretically satisfying solutions to some of the problems handled in an ad doc manner in her program, particularly in the areas of nonmonotonic reasoning and multiple context reasoning. The work described here presents an alternative method for spotting issues applying some of these recent developments.

Our issue spotting method uses a system we are developing for constructing arguments and searching interpretation spaces. The system uses Poole's approach to nonmonotonic reasoning [Poole 88], de Kleer's ATMS approach to managing dependencies between contexts [de Kleer 86a] and a programmable natural deduction theorem prover of our own design. Unfortunately, due to space limitations, we must assume familiarity with Gardner's program, Poole's framework, and de Kleer's ATMS. Some highlights of these systems of particular relevance to our method for spotting issues will be described here, but our treatment will be much too succinct for those not already familiar with them. This paper is a condensed version of a much more detailed GMD research paper [Gordon 89], which should be available by the time this appears.

2 Jurisprudential Background

Most previous work on computational legal reasoning, e.g. [McCarty 77,Sergot 86a,Gordon 87], has been based on the discredited theory of legal reasoning known as *conceptualism* or *mechanical jurisprudence*. To be fair, developers have usually been aware of this limitation of their systems; they expect their systems to be used for testing interpretations of the law, not for mechanically deciding cases.

The theory of legal reasoning most widely believed today, at least by practicing lawyers in common law jurisdictions, is *legal positivism* or *analytical jurisprudence*. Its foremost proponent is H.L.A. Hart [Hart 61]. Legal positivism claims that there

*The research reported here was conducted as part of the Assisting Computer project (AC) of the GMD's Institute for Applied Information Systems. I would like to thank my collegues Gerd Brewka, Ulrich Junker and, especially, for his last minute editing, Joachim Hertzberg, for their generous assistance with the ideas in this paper.

are *clear cases*, decidable by mechanically applying rules of law to the facts of the case, and *hard cases* which can only be decided by the application of judicial discretion. Notice that there is a trace of mechanical jurisprudence left when cases are considered clear. Legal positivism was the basis for the approaches to computational legal reasoning described in [Gardner 87,Susskind 87].

Legal positivism has its critics, of course, most notably Dworkin [Dworkin 77]. Of particular relevance here is Leith's argument that there can be no clear cases because, as legal positivism admits, there can be no method for identifying valid legal rules or the "core of certainty" of legal rules [Leith 85].

Whatever the merits of Leith's argument, our system for supporting legal reasoning does not depend on the distinction between clear and hard cases made in legal positivism. The legal reasoning tasks performed by the system include:

1. Identify legal and factual issues.
2. Construct arguments resolving issues.
3. Manage dependencies between versions of facts, alternative interpretations of legal texts, arguments and inferences.

It must be pointed out that the performance of the system depends critically on the competence of the lawyers using it, who remain completely responsible for all other legal tasks, such as identifying and interpreting relevant legal texts.

Our system was originally motivated by Fiedler's vision of a CAD (Computer-Aided Design) system for drafting legal arguments [Fiedler 85]. In that paper, Fiedler sketched out an alternative to the usual deductive view of legal reasoning, which one may want to call a *constructive* theory of legal reasoning. The role of deduction in Fiedler's conception of legal reasoning is principally one of *justifying* decisions, after they have been made, at least tentatively, rather than to provide a method for reaching or discovering decisions.

In AI terms, legal reasoning according to Fiedler's constructive theory can be viewed as search through the space of interpretations of legal texts and facts of the case.[1]

A philosopher whose work has been unduly neglected by the AI and Law community is Stephen Toulmin. We in the field often argue that jurisprudence is a good source of ideas and methods for AI in general. Toulmin made a stronger claim in 1958; he argued that those interested in the nature of common sense reasoning should take legal reasoning as their model [Toulmin 58, pp. 7-8]. Toulmin is of interest to us here not just because of his positive appraisal of jurisprudence and legal reasoning for "logic". His work on the structure of arguments includes discussions of many of the themes which are of current interest in the AI, including nonmonotonic (defeasible) reasoning, probabilistic inference, and the role of justification. Although his discussion of these matters is much too abstract to serve as the basis for an AI program, his philosophy was another important source of guidance during the design of our system.

3 Preliminaries

Our system is an incremental implementation of Poole's approach to default reasoning using a Natural Deduction theorem prover and an ATMS. That is, it will be one of the first AI sytems to support default reasoning, multiple context reasoning, and reason maintenance, with each component having a solid theoretical foundation. It is not possible here to describe the implementation of our Natural Deduction theorem prover, which makes effective use of the ATMS, or our implementation of an incremental version of Poole's THEORIST language, which is realized as a compiler into, primarily, statements of the control language of our prover. See [Gordon 89] for details. We have chosen to focus here on the use of the system for issue spotting, which does not require knowledge about the implementation of our prover. However, it *is* necessary to know something about Poole's framework for default reasoning and de Kleer's ATMS. Certain particularly relevant aspects of these systems are discussed briefly next.

3.1 Poole's Framework for Default Reasoning

Poole argues in his recent AI Journal article [Poole 88] that there is no need to define a new logic to support common sense or defeasible reasoning. Poole views common sense reasoning as theory formation, where deduction plays an important but subordinate role. Happily, this view of common sense reasoning complements our constructive view of legal reasoning; both view reasoning as theory construction.

In his article, Poole presents his theory in stages,

[1] Rissland described the *task* performed by Gardner's program in the same terms [Rissland 88]. However, Gardner does not describe her program in this way and, indeed, unlike the system presented here, her program does not explicitly represent and search in the space of interpretations.

starting with a basic system and then extending it with a concept of constraints to resolve certain problems with the initial proposal. However, even the extended system is quite simple, so here I will just briefly present the complete system, and refer you to his article for a more completely motivated treatment of the rationale for the various constructs of the approach. The user provides three sets of first-order formulae:

- a set F of wffs, called the "Facts". Notice, that in contrast to the notion of facts used in Prolog, these facts can be arbitrary wffs; they are not restricted to atomic formulae. These wffs represent propositions about which we are confident and do not wish to call into question.
- a set Δ of *possible hypotheses*. Any ground instance of a formula in Δ can be used a hypothesis. In other words, although these formulae are syntactically indistinguishable from ordinary first-order formulae, they are used as formula schemata, with free variables in the formulae serving as schemata parameters. Nonetheless, Δ does denote a set of ordinary first-order formulae, the set obtained by expanding each of the schemata with all possible substitutions out of the set of available ground terms. It is not required that Δ be finite, or that all of its members be identified before reasoning begins.
- a set C of wffs, called the *constraints*.

An important property of Poole's approach is that Δ need not be consistent. These formulas are not a theory or representation of some domain. Rather, they are just the raw material out of which we are at the moment prepared to construct explanatory theories of the domain of interest. As is well known, anything can be derived from an inconsistent set of formulas in standard predicate calculus. This has been a problem for large "knowledge bases", as it difficult, if not impossible, to consistently "represent" a realistic common sense domain using predicate logic. Poole's approach only partially avoids this problem, as the facts and constraints together must still be consistent.

Now for a few definitions:

- A *scenario* of F, C, Δ is a set $D \cup F$ where D is a set of ground instances of Δ such that $D \cup F \cup C$ is *consistent*.
- An *explanation* of a closed formula g is a scenario from F, C, Δ which implies g. That is, a set of ground instances D of the schemata in the set of hypothesis Δ provides an explanation for g iff $D \cup F \models g$ and $D \cup F \cup C \not\models \bot$.
- An *extension*of F, C, Δ is the set of logical consequences of a maximal scenario. A scenario is maximal if no other instance of a schema from Δ can be consistently added to it. Poole shows that a formula is explainable if and only if it is in some extension. Multiple extensions are possible. Poole notes that there is always at least one extension if the facts and constraints are consistent.

That's it! These definitions provide a complete semantics for a form of defeasible reasoning building on the standard model-theoretic semantics of first-order logic.

In addition to his theoretical framework for default reasoning, Poole has designed and implemented a "programming language", called THEORIST, applying the framework. Essentially the THEORIST language merely consists of a concrete syntax for formulas and three key words for categorizing formulas as either defaults, facts, or constraints. Let us now use this language (with a slightly different syntax to avoid typographical problems; predicates, e.g., may include spaces) to show how the approach can be applied to one of the standard examples of legal defeasible reasoning, Hart's park vehicles example:

```
default d1(X) : not allowed in park(X)
 <- vehicle(X).
constraint allowed in park(X) -> not d1(X).
fact bicycle(X) -> vehicle(X).
fact bicycle(a).
```

The purpose of the constraint in the example is to prevent the contrapositive form of the default from being used to derive that something is not a vehicle if it is allowed in the park. This is a useful feature for legal reasoning. Legal rules have directionality.

With this "program", `not allowed in park(a)` is *explained* by the following theory:

```
d1(a)
d1(X) -> vehicle(X) ->
  not allowed in park(X)
bicycle(X) -> vehicle(X)
bicycle(a)
```

That is, bicycles are not allowed in the park in this theory. We could add a default rule stating that bicycles are permitted:

```
default d2(X) : allowed in park(X)
   <- bicycle(X).
```

```
constraint not allowed in park(X)
   -> not d2(X).
```

Now theories can be constructed explaining *both* `allowed in park(a)` and `not allowed in park(a)`.

That is, plausible arguments can be made for both sides of the case. In the nonmonotonic reasoning literature this is known as the "multiple extensions problem". For legal reasoning, this is not a problem but a feature. A constraint could be added to the program cancelling the general default for vehicles in the case of bicycles, if we want to eliminate the first extension.

It is tempting to suggest that Poole's notion of explainability coincides with Llewellyn's theory of "technically defensible arguments". (See [Gardner 87, pp. 9-10] for a discussion of Llewellyn's theory.) However, explainability is both too restrictive and too permissive. It is too restrictive because, I submit, arguments should not have to be proven consistent to be technically correct. It is too permissive because Poole does not restrict the set of allowed hypotheses; one is free to put forward arguments having, in Toulmin's sense, no authoritative *backing*, for example.

3.2 De Kleer's ATMS

The principal job of a reason maintenance system is to cache inferences which have been made. At the cost of memory, (time) efficiency is increased. De Kleer's description of his ATMS reason maintenance system gives the impression that it is only capable of managing dependencies between atomic sentences in a propositional logic [de Kleer 86a]. As the following rational reconstruction of the ATMS makes clear however, dependencies between arbitrary first-order wffs can be managed:

- Inferences are recorded by asserting *sequents* or, if you prefer, *inference rules*, of the form

  $$\Gamma \vdash_{atms} \phi,$$

  where Γ is a set of arbitrary wffs in a first-order language and ϕ is a wff.

- A set of such sequents Δ define a custom *inference relation*, also denoted by $\vdash_{atms}$.

- $\bot$ is a constant atomic formula denoting *false*.

- $\vdash_{atms}$ is *correct* only if it is a subset of the usual first-order predicate calculus derivability relation, $\vdash$. Correctness of a $\vdash_{atms}$ is the responsibility of the user.

De Kleer's major contribution is an efficient implementation of $\vdash_{atms}$ which allows it to be *incrementally* defined. The ATMS increases efficiency as $\vdash_{atms}$, unlike ordinary first-order derivability, is decidable.

To understand our concept of an issue, defined later, an implementation detail of the ATMS must be mentioned. Each wff stored in the ATMS has a *label*, which is a set of environments or contexts (i.e. a set of a set of wffs) from each of which the wff has been derived. These labels are guaranteed by the ATMS to satisfy four properties: correctness, consistency, completeness and minimality. Because of the monotonicity of the ATMS, to determine whether $\Delta \vdash_{atms} \phi$ holds, it is only necessary to check whether $\Delta \supseteq \Gamma$ for some Γ in the label of ϕ.

Although the ATMS is monotonic, it can be used to support nonmonotonic reasoning using Poole's framework. Let F be a set of fact wffs, C be a set of constraint wffs and Δ be a set of defaults (wff schemata). Recall that, in Poole's theory, a *scenario* is a set of wffs $D \cup F$, where D is a set of instances from Δ, and that such a scenario is an *explanation* of a closed formula ϕ if $D \cup F \models \phi$ and $D \cup F \cup C \not\models \bot$. As it turns out, if $D \cup F \vdash_{atms} \phi$ then $D \cup F$ is an explanation of ϕ, *assuming that* $D \cup F \cup C \not\models \bot$. This assumption is justifiable if $D \cup F \cup C$ is not *known* to be inconsistent; that is, if $D \cup F \cup C \not\vdash_{atms} \bot$. This relationship allows the ATMS to be used to implement an incremental version of Poole's THEORIST programming language.

4 Issue Spotting

In her excellent book on Artificial Intelligence and legal reasoning, Anne Gardner describes an AI system capable of spotting issues in offer and acceptance law school examination questions [Gardner 87]. Although it is difficult to do Gardner's complex program justice in this short space, at the risk of oversimplifying I would like to briefly summarize its architecture, as I understand it. Then I will present and compare our approach to issue spotting.

4.1 Gardner's Approach

Gardner represented the law of offer and acceptance, as well as relevant common sense knowledge, in three ways:

1. An *Augmented Transition Network* is used to represent the relationship between the various central legal events of offer and acceptance law, such as offer, counteroffer, acceptance, and so on.

As the network is traversed, the current node is occupied by an MRS [Genesereth 83] representation of the facts of some event in the examination question. The arcs represent alternative legal interpretations of such facts.

2. A predicate is associated with each arc of the transition network. An arc can be tranversed only if the predicate is satisfied by the facts of the current event. These predicates are defined with MRS "rules", as are other legal and common sense concepts. Rules may be organized into rule sets and marked as complimentary or competing. The rules in a competing rule set represent alternative interpretations of some legal rule.

3. The system also includes a limited facility for representing and reasoning with *cases*. The stored cases are all considered to be *clear cases*. These are used "when the rules run out" to check whether some question is easy or hard. If the relevant facts of the problem match a stored case, the question is considered easy. Cases are also used to realize a kind of defeasible reasoning. MRS rules are used as defaults. If an applicable MRS rule conflicts with a stored case matching the facts of the current problem, the issue is considered hard and the alternative solutions are both recorded in the program's output.

It is not necessary to try to explain here just how Gardner's program works. However, its output is a two-tiered decision tree. The top-level tree is a tree of contexts, where a context is a set of MRS statements or wffs. Each node in the top-level tree includes its own lower-level decision tree for determining which branch in the top-level tree to follow. Each branching point in these trees represents a legal or factual *issue*, or *hard question*. The easy questions have all been decided during the process of generating these trees; they are not reflected in the output.

Alternative theories of the case can be recovered from the top-level tree. The context of each node in the tree represents one interpretation of the case up until the event to be interpreted at that point in the tree. The context associated with a node, it seems, inherits and extends the context of the node's parent. Thus, the contexts of the leaf nodes of the top-level tree contain complete alternative theories of the case.

Thus, the output of Gardner's program contains not only the issues spotted in the problem (represented by the choice points in the decision trees) but also alternative arguable resolutions of these issues (represented by the branches of these trees).

4.2 Our Approach

Now I would like to discuss how an ATMS-based system for theory formation tasks, such as ours, can be used to spot issues. The resulting system will be only roughly comparable to Gardner's. We do not attempt, e.g., to use case-based reasoning to resolve hard questions. Nonetheless, as we will see, the two systems have a great deal of overlapping functionality.

We start by supposing that the relationship between the relevant legal events has been represented in some first-order theory, rather than in an ATN. There are a variety of ways to accomplish this. One could, for example, directly translate the ATN into a set of predicate logic sentences. Nodes could be represented by terms; arcs by a binary predicate. Another possibililty would be to use some variety of "situation calculus". (See [Genesereth 87, pp. 263-283] for an introduction to representing states, events and actions in predicate logic.) This is not a trivial problem, of course. One must be especially careful about the *frame problem*. One approach we intend to investigate is applying Poole's framework for default reasoning. Goodwin and Goebel suggest that Poole's framework can be used to solve a variety of the temporal reasoning problems in the literature, such as the Yale Shooting Problem [Goodwin 89].

In our system, "common sense" knowledge and alternative interpretations of legal rules are to be represented using Poole's framework for default reasoning. The constructs of Poole's THEORIST language are translated into first-order wffs and into control rules of our Natural Deduction prover. Unlike in Gardner's system, there is no need to manually designate rules as being competing or compatible.

Although we do not use case-based reasoning, we arguably can achieve functionality comparable to Gardner's use of cases with Poole's framework for default reasoning. Gardner primarily uses cases to achieve a kind of default reasoning; they can call the applicability of general rules into question. Cases are not used, for example, to support analogical reasoning.

Intuitively, an issue is a proposition which is *known* to play a role in an argument proving some other proposition of interest. Other propositions are irrelevant, or simply "non-issues". Recall that the label of an ATMS wff is a set of environments, where each environment is itself a set of wffs. Each environment is a minimal, consistent set of wffs from which the wff is ATMS derivable. Using these labels, we can capture the intuitive idea of an issue with the following definition:

Definition 1 (Issue) *A wff ψ is an* issue *iff*

1. $\exists\ \Gamma\ .\ \Gamma \in l\phi$, *where* $l\phi$ *is the* label *of the goal wff,* ϕ, *and*

2. $\psi \in \Gamma - \Theta$, *where* Θ *is the current context.*

That is, in English, a formula is an issue (or, if you prefer, denotes an issue) if and only if there is an environment in the label of the goal such that the formula is a member of the difference between this environment and the context of interest. Thus, "issue" is a relative concept; whether or not a wff is an issue depends on the wff to be proved and some context. The context is the set of wffs we are currently assuming *for the sake of argument*. The undisputed facts of the case can be one such context.

The first condition of the definition requires that an argument has already been found which would prove the wff of interest if the wffs in $\Gamma - \Theta$ are true, as well as the wffs we are assuming, Θ. Thus, issues are raised only after the arguments have been discovered. This seems intuitively correct. A proposition becomes an issue only after its potential significance is called to our attention.

The second condition removes the wffs in the current context Θ from consideration. A wff in Θ cannot be an issue, as the whole point of having a current context is to be able to stipulate that certain things are true, at least for the sake of argument. A wff in Θ can, of course, become an issue by deleting it from Θ, thus switching contexts.

An important property of this definition is that the minimality of labels, guaranteed by the ATMS, ensures that no irrelevant wffs are considered to be issues.

Supposing now that the law of offer and acceptance has been represented as a THEORIST program, how could our system be used to identify the issues raised by an examination question, using this definition? Essentially, the following procedure is proposed:

1. represent the facts presented in the exam question with THEORIST facts (i.e. ATMS premises);

2. let $\phi = \exists\ x\ .\ \text{contract}(x)$

3. while there is time and theories can be found:

   (a) try to find a Γ such that $\Gamma \vdash \phi$;

   (b) if such a Γ has been found, try to find a Δ such that $\Delta \cup \Gamma \vdash_{atms} \bot$ or, more strongly, $\Delta \vdash_{atms} \neg\phi$;

4. inspect the labels of ϕ, $\neg\phi$ and $\bot$ in the ATMS to retrieve the issues.

Here ϕ is the main issue of the case. It is assumed that the main issue is known in advance, or is apparent. (This is an assumption made in Gardner's system as well.) It is also assumed that the facts of the case, as they are represented in the examination question, are consistent. If this cannot be assumed, then they should be represented as THEORIST defaults, rather than THEORIST facts. The main loop of the program consists of alternating attempts to construct a theory showing that a contract exists and then formulating a rebuttal or counterargument to that theory. Although I have not shown this here, the heuristics encoded into the control rules of our system are used to find easy arguments first. This main loop can be left at any time. Gardner's program apparently runs until all hard questions have been discovered. This can be achieved here by continuing until no other theories for or against the main issue can be generated from the available hypotheses.

At any time, a list of the issues currently identified can be retrieved. Again, whether or not some wff is an issue depends on the goal and some context of interest. Here the goal is the top-level goal of the exam question, whether or not there is a contract. If the facts of the question have been represented as THEORIST facts, as suggested above, then this context is empty, as THEORIST facts are in turn represented as ATMS premises, which are universally true.

Essentially, when answering examination questions, the law student plays the roles of the attorneys for both the plaintiff and defendent. I have suggested that the student alternate between these roles, but there may be other strategies, such as first trying to find a number of alternative arguments for the plaintiff before switching perspective to represent the defendant.

The definition of issue is general enough to handle rebuttals and counterarguments. A rebuttal can be defined as follows:

Definition 2 (Rebuttal) *An environment* Δ *is a* rebuttal *to* $\Gamma \vdash_{atms} \phi$ *iff* $\Delta \cup \Gamma \cup \Theta \vdash_{atms} \bot$, *where* ϕ *is the conclusion to be rebutted,* Γ *is theory constructed by the proponent of* ϕ *and* Θ *are the stipulated facts in the current context.*

If Δ is empty, then the theory constructed by the proponent of ϕ was already known to be inconsistent with the stipulated facts in Θ. It would have been possible to define an issue in such a way as to require Γ to be ATMS consistent with ϕ, but that would have meant that we could not use the definition to delineate the issues raised by a rebuttal, precisely because in a rebuttal we want Γ to be ATMS inconsistent with the stipulated facts. That is, using the definition as

it is, the issues raised by a rebuttal are the following: Let $\Theta' = \Theta \cup \Gamma$ and $\Gamma' = \Delta \cup \Theta'$. Then ψ is an issue iff Γ' is in the label of $\bot$ and $\psi \in \Gamma' - \Theta'$. Thus, rather than refusing to consider a wff to be an issue when its environment is known to be inconsistent with the stipulated facts, we simply allow for an easy rebuttal.

A counterargument, on the other hand, can be defined as:

Definition 3 (Counterargument) *An environment* Δ *is a* counterargument *to* $\Gamma \cup \Theta \vdash_{atms} \phi$ *iff* $\Delta \cup \Theta \vdash_{atms} \neg\phi$.

That is, in a counterargument the opponent of ϕ tries to construct another theory in which $\neg\phi$ is derivable. Extralogical arguments should also be made, showing that this theory is to be preferred. A wff ψ is an issue raised by a counterargument iff $\exists\, \Gamma\, .\, \Gamma \in l\neg\phi$ and $\psi \in \Gamma - \Theta$. Again, this is just an application of our issue definition.

5 Conclusion

There are a few more points to be made in comparing our approach to issue spotting with Gardner's. Firstly, the output of her program is a decision tree for deciding the case, where the nodes of the tree represent the remaining hard questions, or issues. In contrast, our approach does not generate a procedure for deciding the case. It should not be difficult, however, to design such a procedure using our issue concept. For example, one could select an issue (e.g. ϕ or $\neg\phi$?) decide the issue, add the choice made to Θ and then recompute the issues. This procedure could then be repeated until no issues remain. An interesting area for further research would be to develop a strategy for picking the issue to consider next.

Secondly, because our approach does not distinguish between hard and easy questions, *all* questions known to make a difference are issues. However, we can remove all issues deemed to be easy by just extending Θ, the stipulated facts.

Thirdly, the alternative theories of the case are represented in the leaf nodes of the decision tree in Gardner's system. In our approach, these theories are represented in the label for the wff representing the main question of the case. An important point is that the ATMS quarantees that the environments in these labels are minimal. That is, no wffs are in these environments which are unimportant for deciding the case. It is unclear whether this claim can be made for Gardner's system.

Finally, a related point is that Gardner's system may identify too many issues. Although all of the issues identified may represent hard questions, it is unclear whether each of the hard questions in some branch of the decision tree must really be addressed to decide the case. Again, the ATMS guarantees that this problem cannot arise.

In the introduction of this paper we stated that we would present a new *method* for spotting issues. Although we have done this, we have also done something else of perhaps greater significance: we have presented a formal theory of issue spotting. Our definition of issue depends only on the formal semantics of predicate logic and de Kleer's ATMS. (Interestingly, it does not depend directly on Poole's theory; this is because of the way Poole builds his approach to default reasoning on standard predicate logic.) Our issue concept does not depend on any particular program or method for implementing a theorem prover or ATMS.

The prototype of the system discussed here is being implemented in Standard ML, a typed functional programming language [Harper 88]. Although the implementation is progressing well, there is still much to be done. Our group is also, at the same time, developing a legal application using the system; it will be a system for "assembling" legal documents in the field of German divorce law.

References

[de Kleer 86a] Johan de Kleer; *An Assumption-Based TMS*; Artificial Intelligence; Volume 28; 1986.

[Dworkin 77] Ronald Dworkin; *Taking Rights Seriously*; Harvard University Press; 1977.

[Fiedler 85] Herbert Fiedler; *Expert Systems as a Tool for Drafting Legal Decisions*; II Convegno Internazionale Logica, Informatica, Diritto; Florence; 1985.

[Gardner 87] Anne von der Lieth Gardner; *An Artificial Intelligence Approach to Legal Reasoning*; MIT Press; 1987

[Genesereth 83] Michael R. Genesereth; *An Overview of Metalevel Architecture*; AAAI-83; Washington D.C.; Morgan Kaufmann; 1983.

[Genesereth 87] Michael R. Genesereth and Nils J. Nilsson; *Logical Foundations of Artificial Intelligence*; Morgan Kaufmann; 1987.

[Goodwin 89] S.D. Goodwin, R.G. Goebel; *Nonmontonic Reasoning in Temporal Domains: The*

Knowledge Independence Problem; Second International Workshop on Non-Monotonic Reasoning; Grassau; Springer-Verlag; 1989.

[Gordon 87] Thomas F. Gordon; *OBLOG-2: A Hybrid Knowledge Representation System for Defeasible Reasoning*; First International Conference on Artificial Intelligence and Law; Boston; 1987.

[Gordon 89] Thomas F. Gordon; *A System for Planning Arguments and Seraching Interpretation Spaces*; GMD Working Paper; to appear 1989.

[Harper 88] Robert Harper, Robin Milner and Mads Tofte; *The Definition of Standard ML, Version 2*; Laboratory for Foundations of Computer Science, University of Edinburgh; Report ECS-LRCS-88-62; 1988.

[Hart 61] H.L.A. Hart; *The Concept of Law*; Clarendon Press; 1961.

[Leith 85] Philip Leith; *Clear Rules and Legal Expert Systems*; II Convegno Internazionale Logica, Informatica, Diritto; Florence; 1985.

[McCarty 77] L. Thorne McCarty; *Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning*; Harvard Law Review; Volume 90; 1977.

[Poole 88] David Poole; *A Logical Framework for Default Reasoning*; Artificial Intelligence; Volume 36; Number 1; 1988.

[Rissland 88] Edwina L. Rissland; *Artificial Intelligence and Legal Reasoning, A Discussion of the Field and Gardner's Book*; AI Magazine, Fall 1988.

[Sergot 86a] M. J. Sergot, F. Sadi, R. Kowalski, R. A. Kriwaczek, P. Hammond and H. T. Cory; *The British Nationality Act as a Logic Program*; Communications of the ACM; Volume 29; 1986.

[Susskind 87] Richard E. Susskind; *Expert Systems in Law*; Oxford University Press; 1987.

[Toulmin 58] Stephen Toulmin; *The Uses of Argument*; Cambridge University Press; 1958.