法とコンピュータ

No.5  July 1987

《特集 コンピュータ・ネットワーク社会のフロンティア —法と技術》

主要目次

圏頭言
〈法とコンピュータ学会国際シンポジウム〉
名誉会長挨拶
理事長挨拶
基調講演〈コンピュータ・ネットワーク社会のフロンティア〉
セッション1；コンピュータ・ネットワーク取引の法的問題
セッション2；ソフトウェアの保護
セッション3；個人データ保護
セッション4；半導体チップ保護
セッション5；法律エキスパートシステム
〈外国判例紹介〉
〈文献目録〉

法とコンピュータ学会

The Law and Computers Association of Japan
The Journal of the Law and Computers Association of Japan

LAW AND COMPUTERS

No.5  July 1987

(Special Issue: Frontiers of the Computer Network Society
— Legal & Technological Problems)

CONTENTS

Foreword

〈Reports on the Law and Computers Association International Symposium〉

Address of Honorary President
Address of Chairman
Keynote Lectures; Frontiers of the Computer Network Society
Session 1; Legal Problems of Computer Network Transactions
Session 2; Software Protection
Session 3; Protection of Personal Data
Session 4; Protection of Semi-Conductor Chips
Session 5; Legal Expert Systems

〈Foreign Judicial Decisions〉

〈Bibliographies〉
Intelligent Legal Information Systems: 
an update

L. Thorne McCarty*

In September, 1982, at a colloquium in Leicester, England, I presented a paper entitled "Intelligent Legal Information Systems: Problems and Prospects," which was subsequently published as [16] and [18]. Reviewing this paper four years later, it is clear that the research activity in this field has increased dramatically, but the basic analysis remains the same. In the present paper, I will update my earlier discussion through October, 1986, and highlight the most important outstanding problems, as I see them.

As in the earlier paper, it is useful to divide intelligent legal information systems into two categories. In Section 1, I will discuss "Legal Analysis and Planning Systems," and in Section 2, I will discuss "Conceptual Legal Retrieval Systems." The possibility of constructing hybrid systems will also be mentioned. Finally, in Section 3, I will outline the features of a "Language for Legal Discourse" which is intended to serve as the foundation for the legal information systems of the future.

1. Legal Analysis and Planning Systems

Although it is common in the literature to refer broadly to legal expert systems, I prefer to distinguish two varieties of expert systems in the legal field. First, imagine a system which accepts as input the factual description of a case, and provides as output a classification of the case in terms of various legal categories, e.g., the system report that the client owes a capital gains tax. We would call this a legal analysis system. It locates the relevant legal rules, and it provides a suggested analysis of a fixed set of facts. More interesting, though, is a legal planning system. Here the set of facts would not be fixed, but variable. The lawyer would describe an initial situation, e.g., an existing corporate structure, and a desired end result, e.g., an acquisition of certain assets, and the system would search the space of possible transaction patterns to suggest a course of action which satisfied additional constraints, e.g., minimal tax consequences. Although the generic planning problem has been studied extensively by artificial intelligence researchers, no one has yet built a true legal planning system. All the examples of legal expert systems to date have been legal

* Professor of Computer Science and Law, Rutgers University, New Brunswick, New Jersey, U.S.A.
analysis systems.

In comparing these various legal analysis systems, it is helpful to categorize them in terms of the computational technology they use. The simplest systems are based on conventional programming techniques. The author works out the complete decision tree for a particular legal problem, and then encodes the possible paths through this tree in a procedural programming language, such as BASIC or PASCAL. The best published examples of this approach can be found in the work of Robert Hellawell from the Columbia Law School [9] [10] [11], and William Boyd from the University of Arizona Law School [3]. The work of Larry Farmer and his colleagues at Brigham Young University [7] also fits within this framework. Although it is relatively simple to write programs in this style, once the decision tree has been specified, there are several problems with this approach. The most significant problems arise from the fact that conventional programs contain no explicit representation of the legal rules, but only represent legal rules implicitly by a pattern of conditional branch statements. This means that it is difficult to examine the program and verify that the rules are correctly stated, and it is difficult to modify the program if the rules change.

For these reasons, most researchers have adopted the technology of rule-based expert systems, in which the legal rules are given an explicit, declarative, modular representation. It is helpful to classify these rule-based systems further by considering the particular rule representation chosen, where the simplest choice involves the use of only propositional rules. The representation of legal rules in propositional logic has a long history, going back to Layman Allen's work in the 1950's [1], and including the JUDITH system developed by Walter Popp and Bernhard Schlink in the early 1970's [27]. More recently, James Sprowl's ABF system [31] is essentially a backward-chaining interpreter for a subset of propositional logic, influenced strongly by the work of Layman Allen. The various legal systems developed within the MYCIN paradigm, such as Robert Michaelson's tax planning system written in EMYCIN [25], and the product liability systems written by Don Waterman and Mark Peterson in ROSIE [32] [26], are also closely related. Although the MYCIN rules are composed of object/attribute/value triples, which have some of the characteristics of predicate logic in principle, they are invariably reduced to a propositional system in practice.

More general, and more powerful, is a representation of legal rules in first order logic. The earliest implementation of my own TAXMAN system, completed in 1972-73, fits within this category [15]. The original TAXMAN I system was written in micro-PLANNER, which was a forerunner of PROLOG, a contemporary language based on the Horn clause subset of first order logic [12]. Today, there is an enormous amount of research activity devoted to PROLOG-based legal expert systems. The most extensive project, and the best known, is the work of Robert Kowalski, Marek Sergot, and their colleagues at Imperial College, London [29]. Other work in the published literature includes the systems developed by Dean Schlobohm [28], Duncan MacRae [14], and David Sherman [30], and new projects are appearing almost weekly. My own judgment, currently, is that PROLOG is indeed the language of choice for the construction of a simple
backward-chaining, rule-based legal expert system.

However, there are serious deficiencies in a straightforward encoding of legal rules into PROLOG. Although Horn clause logic provides the modularity and flexibility that is lacking in conventional programming languages, thus simplifying the task of constructing and maintaining a large body of legal knowledge, the behavior of a PROLOG-based legal expert system is virtually the same as the behavior of Robert Hellawell's systems written in BASIC [9] [10] [11]. Both systems proceed backwards from a top-level goal, and they both pose questions to the user to elicit the facts of a case; the user is not permitted to volunteer facts which he knows are relevant, and he is not permitted to volunteer a partial legal analysis. Furthermore, a true planning system, which searches a space of transactions to satisfy a given goal, cannot be constructed in this way, since the rules of the analysis system are not, by themselves, sufficient to generate transaction patterns. I argued in my 1982 paper that the solution to these problems lies in the construction of a deep conceptual model of the legal domain. I will return to this point in Section 3 of the present paper.

2. Conceptual Legal Retrieval Systems

The construction of a deep conceptual model of a particular legal domain is also necessary for the second component of an intelligent legal information system: the conceptual legal retrieval system. The basic idea here is to use the conceptual model of a legal domain to build up a database containing the essential information about a set of cases: the facts, the applicable rules, the alternative analyses, etc. For example, if the domain were corporate tax law, we could use the conceptual model of the TAXMAN system [15] [21] [22] to represent (1) the transaction patterns of particular cases, (2) the relevant rules and concepts from the Internal Revenue Code, and (3) an analysis of how these rules were applied, or not applied, in each particular case. The search procedures for this database would then include a set of pattern-matching operations, at various levels of abstraction. We could search for factual patterns, conceptual patterns, analysis patterns, etc. The advantages of this system should be readily apparent. By comparison to the full-text key-word searches used in LEXIS, WESTLAW, and similar systems, the pattern-matching searches would be much more closely attuned to the way a lawyer naturally thinks about a case, and the conceptual retrieval system would therefore provide a much more precise and flexible access to the data.

Surprisingly, there has been very little research so far on this proposal for a conceptual legal retrieval system. The early dissertation work of Jeffrey Meldman [24] was partially motivated by these concerns, but Meldman never implemented his system, and he abandoned the project after completing his thesis. The only other significant work on this problem so far is the dissertation of Carole Hafner, completed at the University of Michigan in 1978, and subsequently published as a monograph by the University of Michigan Press [8]. Hafner selected as her problem domain Articles 3 and 4 of the Uniform Commercial Code, which are the provisions governing negotiable instruments such as checks and notes. Working with a conceptual model of the situations that typically occur in negotiable instruments law, she constructed a data base consisting of approxi-
mately 200 cases and 200 subsections of the Code, plus a query language which permitted her to search for certain patterns in the data. Since I reviewed Hafner's work in detail in my 1982 paper, I will not discuss it further here. Another project along these lines, not yet completed, is the work of Cary deBessonet at the Louisiana Law Institute in Baton Rouge, Louisiana [5]. DeBessonet is building a conceptual representation of several sections of the Louisiana Civil Code, using artificial intelligence techniques. Although the main purpose of his project is to clarify the structure of the Civil Code itself, an additional purpose, once the conceptual model has been formulated, is to experiment with alternative designs for a conceptual legal retrieval system.

There is one serious obstacle to the construction of a conceptual legal retrieval system, however. How can we build up a realistic legal data base? Hafner was able to code her collection of 200 cases and 200 statutory provisions by hand, but an automated knowledge acquisition system would clearly be needed to extend this data base into the thousands, or, realistically, the tens of thousands of cases. There are two approaches which seem to me worthy of further investigation. The first approach involves the use of a human abstractor, engaged in an interactive dialogue with an evolving legal data base, and constrained at all times by a conceptual model of the legal domain. Since the abstractor would be forced to describe each case, in English, in such a way that the system could "understand" it, the resulting abstracts should be much more coherent and consistent than the unconstrained headnotes written today. A second approach, and a much more speculative one, involves the use of a sketchy parser to analyze directly the text of a case. An example of this approach is the FRUMP system, developed at Yale [4] [6], which can "read" the Associated Press wire and extract from it sketchy information about earthquake stories, assassination stories, etc., in several discrete news categories. Transferred to a legal domain, this information would never be reliable or complete, but used in conjunction with the first approach, this second approach might turn out to be quite useful.

Although I have discussed the analysis/planning systems and the conceptual retrieval systems separately in this paper, there is no reason to keep these two systems separate. A hybrid system should be our ultimate goal. The analysis and planning system would be more useful if it could provide direct access to the case materials which justified its conclusions, and this would be possible if the system were linked to a conceptual legal retrieval system. The retrieval system would be more powerful if it could follow the patterns of inference suggested in the cases, and this would be possible if the system had access to the rules of the legal analysis system. The key to all of this, as I pointed out in my 1982 paper, is to write both systems in the same representation language, a language which reflects the underlying conceptual structure of the legal domain. I will call this representation language: A Language for Legal Discourse.

3. A Language for Legal Discourse

In my 1982 paper, I outlined an ambitious research programme. To fully exploit the benefits of advanced computer science techniques in legal information systems, I wrote, it is necessary to build a deep conceptual model of the relevant legal domain [16] [18]. This point has often been
misunderstood. It is a simple idea, however, even if the research programme itself is rather difficult.

There are many common sense categories underlying the representation of a legal problem domain: space, time, mass, action, permission, obligation, causation, purpose, intention, knowledge, belief, and so on. The basic idea is to select a small set of these common sense categories, the ones that are most appropriate for a particular legal application, and then develop a knowledge representation language which faithfully mirrors the structure of this set. The language should be formal: It should have a recursive syntax, a precise semantics, and a well-defined inference mechanism. The semantic interpretation of the common sense categories in the language should be intuitively correct, that is, it should generate exactly those entailments that ordinary people (and ordinary lawyers!) generate in similar situations. The inference mechanisms for the language should be complete and sound, in principle, but, in practice, completeness and soundness would often be sacrificed for computational tractability, just as they are in ordinary (and ordinary legal!) reasoning. My claim is: Once a language of this sort has been developed, it will provide a unified framework for the construction of a variety of legal analysis/planning/retrieval systems.

For my initial work on a Language for Legal Discourse, or LLD, I have focused on several features which turned out to be important in the corporate tax domain studied within the TAXMAN project. The language has a full system of sorts and subsorts (e.g., an ACTOR can be a PERSON or a CORPORATION), and it includes both count terms and mass terms (e.g., PERSON is a count term, and STOCK is a mass term). Several common sense categories are represented: states, events, actions, permissions and obligations. The semantic interpretation of these categories is based on my work in [17] and [19]. An important feature of the language is that all expressions are evaluated in partial models, as discussed in [20], and this provides a formal mechanism for reasoning with prototypes. Although LDD was initially designed with the corporate tax domain in mind, the basic features of the language have also been useful in a pilot project on the representation of a portion of the Uniform Commercial Code [2] [23]. It is interesting to note that several other researchers, motivated also by the problems of representing legal concepts, have developed languages with similar features. For example, the Event Calculus developed by Kowalski and Sergot [13] resembles in some ways the state/event sub-language of LLD.

The Language for Legal Discourse is being implemented in Common Lisp on a SUN/3 Workstation. I will describe the language in greater detail in my future publications.

4. Conclusion

In this short paper, I have reviewed the recent work on "Intelligent Legal Information Systems," updating my 1982 paper on the same subject [16] [18]. I have shown that the research in this field so far has been badly skewed. Almost all of the recent work has been devoted to legal analysis systems, with only a small effort devoted to conceptual legal retrieval systems, and no effort at all devoted to true legal planning systems. I have argued that a hybrid analysis/planning/retrieval system should be our ultimate goal, based on a knowledge representation language which
reflects the underlying conceptual structure of the legal domain. As an example, I have outlined my own work on a *Language for Legal Discourse*, which is intended to serve as the foundation for the intelligent legal information systems of the future.

References


