INTELLIGENT LEGAL INFORMATION SYSTEMS: PROBLEMS AND PROSPECTS

VOLUME 9
NUMBER 2
1983

L. Thorne McCarty

Reprinted from
RUTGERS COMPUTER AND TECHNOLOGY LAW JOURNAL
Volume 9, No. 2
Copyright © Rutgers Computer and Technology Law Journal, 1983
INTELLIGENT LEGAL INFORMATION SYSTEMS: PROBLEMS AND PROSPECTS

by L. Thorne McCarty*

TABLE OF CONTENTS

Deep Conceptual Models: An Example .................................. 267
Conceptual Legal Retrieval Systems ................................. 276
Legal Analysis and Planning Systems ............................. 281
Conclusion ................................................................. 287

LIST OF FIGURES

Figure 1: The Organization of Description Spaces ............ 271

The possibility of building an "intelligent" legal information system, an information system which in some sense "understands" the "concepts" of a particular area of law, has attracted much attention in recent years. In the past three years alone there have been several international conferences devoted to this topic: the Conference on Computer Science and Law, held in Swansea in September, 1979 [Niblett 80]; the Conference on Logica, Informatica, Diritto, held in Florence in April, 1981 [Ciampi 82] [Martino 82]; the Symposium on Legal Data Processing in Europe, held in Thessaloniki in July, 1981 [Council of Europe 81]; and the Workshop on Formal Methods in Law, held in Bonn in March, 1982 [Forthcoming]. In addition, as indicated in the initial paper by Stephen Skelly [Skelly 83], this is one of the principal topics of the Colloquium this week in Leicester.

Part of this interest in intelligent systems arises from a desire to augment the current techniques for legal document retrieval, which

*Copyright 1982 by L. Thorne McCarty, Distinguished Visiting Professor of Law and Computer Science at Rutgers University; Professor of Law at SUNY-Buffalo; B.A. Yale (Mathematics and Philosophy, 1966); J.D. Harvard (1969).

This article is reprinted by permission of the author and Sweet and Maxwell, Ltd. who will shortly publish the article in a book entitled, Data Processing and the Law (ed. Colin Campbell).

This article was presented at a Colloquium on Data Processing and the Law, held in Leicester, England, September 15-16, 1982. Sponsored by the UK National Committee of Comparative Law. Publication forthcoming. All rights reserved.

This material is based upon work supported by the National Science Foundation under Grant No. MCS-79-21471 and Grant No. MCS-82-03591. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation.

265
still rely exclusively on full-text key-word search. This goal has often been stated, see, e.g., [Allen 83], and it has stimulated a great deal of research on the morphological analysis of legal language, the construction of legal thesauri, and similar approaches, see, e.g., [Bauer-Bernet 82] [Knapp 82]. There is a sense today, however, that the techniques developed in the fields of artificial intelligence and computational linguistics might provide solutions to these very difficult problems. A second reason for the interest in intelligent legal information systems today has to do with the success of “expert systems” in several other professional disciplines, most notably in medicine [Shortliffe 76] and geology [Duda, et al. 79]. The physician who seeks advice from the MYCIN system on the diagnosis and treatment of a bacterial infection seems roughly analogous to the lawyer who seeks advice on the diagnosis and treatment of a client’s tax problems, but a system capable of providing this advice would have to go far beyond document retrieval to a sophisticated form of information retrieval, as Bryan Niblett has observed [Niblett 81]. It would be an “intelligent” legal information system, in a rather strong sense.

In this paper I will argue that the most critical task in the development of an intelligent legal information system, either for document retrieval or for expert advice, is the construction of a conceptual model of the relevant legal domain. This is a point which appears frequently in the discussions of expert systems in other areas. In the medical field, for example, MYCIN is often described as a shallow rule-based system, because it contains hundreds of judgmental rules relating observed symptoms to plausible bacterial infections, but it possesses no internal representation of the disease mechanisms which account for these symptoms. By contrast, the CASNET/Glaucoma system [Weiss, et al. 78] is generally characterized as a deep system, or a model-based system, because its diagnoses are derived from a representation of the various hierarchical, causal, and temporal relationships which exist among the physiological states associated with glaucoma. Increasingly, the leading researchers in the expert systems field are stressing the importance of these deep conceptual models for the next generation of expert systems, see, e.g., [Hart 82], and the argument seems to me to be particularly persuasive for legal systems. This is not to say that it will be impossible to build an expert legal system out of MYCIN-like rules—in fact, I will discuss examples of such systems later in this paper—but only that the approach has serious limitations. There is a second reason, too, for building an expert legal system on top of a deep conceptual model of the legal domain. As I will attempt to demonstrate, the same conceptual
model which provides a foundation for the consultation system could also provide a foundation for a sophisticated document retrieval system, and this suggests a way to build a complex legal information system within a single unified framework.

Now it may not be apparent to the readers of this paper what I mean by a “deep conceptual model of a legal domain.” In the following section, as an example, I will describe a model of the corporate tax domain which was developed as part of the TAXMAN project by myself and my colleague, Professor N.S. Sridharan of Rutgers. We have described this model in detail elsewhere [McCarty 77] [McCarty and Sridharan 80] [McCarty and Sridharan 81] [Sridharan 81] [McCarty 82], but with an emphasis there on various theoretical issues in either law or computer science. Here I will present a very simple exposition of the model, with an emphasis on its usefulness for practical applications. Following this, I will discuss the two main categories of applications, as I see them: (1) conceptual legal retrieval systems, and (2) legal analysis and planning systems. In each category I will discuss examples of related work on the applications of computers to law, work which comes close to, but does not yet attain, the depth of conceptual modeling which I am advocating. But here I should strike a cautionary note. Although I will argue that the legal information systems of the future should be based on conceptual models similar to those developed in the TAXMAN project, these models will not be easy to formulate, and the corresponding information systems will not be easy to construct. For the near term, then, the critical problem will be to select an appropriate level of conceptual detail, and an appropriate level of system complexity, for each contemplated application. I will return to this point at the end of the paper.

Deep Conceptual Models: An Example

What is our purpose in building a conceptual model of a legal domain? Basically, we are looking for a language in which we can describe legal cases. At a minimum, we want to be able to describe the “facts” of a case at a comfortable level of abstraction, and we want to be able to describe the “law” in our chosen area of application, where the “law” consists of a system of “rules” and “concepts” which specify the rights and the obligations of the various parties. What language should we use? It is well known that different languages incorporate different sets of assumptions about the structure of the world. We are looking here for a language which is rich enough to express the important facts about a particular legal world,
and yet abstract enough to suppress the irrelevant detail. The purpose of our conceptual model, then, is to specify exactly which of these details should be expressed, and which should be suppressed, and how.

To build a model along these lines, we first examine a legal domain, such as corporate tax law, and we identify the conceptual objects and the conceptual relationships which occur in that domain. What entities do we want to talk about, we ask, and what do we want to say about them? For corporate tax law, as a first approximation, we will be talking about ACTORS, who can be either CORPORATIONS or natural PERSONs; we will be talking about intangible PROPERTY, which can be either CASH, or a DEBT, or a SECURITY, the latter being either a STOCK or a BOND; and we will be talking about the OWNership relation between an ACTOR and a PROPERTY. Using these simple concepts, we then construct descriptions of the following sort:

\[(OWN 01
\langle owner (PERSON A1) >
\langle owned (STOCK S1)
\langle issuer (CORPORATION C1) >
\langle quantity (NUMBER N$1)$ >
\langle common YES) >
\langle quantity (NUMBER N$1)$ )\]

which we might paraphrase in English as “the OWNership by a PERSON A1 of N$1$ shares of COMMON STOCK issued by a CORPORATION C1, for which there are N$1$ shares outstanding.” These descriptions are then used in two ways. First, we can instantiate a description, replacing its free variables with the names of concrete individuals, in order to create a network of particular facts about a given case. Second, given a fully instantiated case, we can match a description to the facts to determine whether it is satisfied, and, if so, to determine the bindings of its free variables.

For example, consider the following interaction with the TAXMAN system, in which the program prompts with the name of a context, CASE, and the name of a state, TIME-1, and the user then enters the information shown in lower case:

CASE:TIME-1 (make (corporation dupont))

DUPONT

CASE:TIME-1 (make (stock

  (issuer dupont)
  (quantity 50000.0)
  (common yes)))
STOCK-1

This trace illustrates our procedures for the instantiation of a description. The first command from the user creates an instance of a CORPORATION, named DUPONT. The second command creates an instance of a COMMON STOCK, named STOCK-1, which is issued by DUPONT, and which has 50,000 shares outstanding. At this point, if we were to display the network of facts about the newly created stock, we would see the following:

```
STOCK
   STOCK-1
      QUANTITY
          50000.0
      COMMON
          YES
SECURITY
   STOCK-1
      ISSUER
          CORPORATION
          DUPONT
PROPERTY
   STOCK-1
```

In other words: "There exists an object named STOCK-1, which is a common STOCK of quantity 50,000; STOCK-1 is also a SECURITY, whose issuer is a CORPORATION named DUPONT; STOCK-1 is also a PROPERTY." Notice that the system has here added the information that every STOCK is a SECURITY, and that every SECURITY is a PROPERTY, which was contained in our initial model of the domain.

Now consider the converse operation. Suppose we have just added to our network, in the state TIME-1, the additional fact that "John D. Rockefeller owns 1000 shares of the common stock issued by duPont," or, as the system would display it:

```
OWN OWN-1
   OWNER
       PERSON
          ROCKEFELLER
   OWNED
      STOCK
          STOCK-1
      QUANTITY
          1000.0
```
It should be apparent that we can now match our original OWNership description to this factual network, and that the match would succeed, at least in the state TIME-1. The match would be successful, of course, because the variable A1 could be bound to a PERSON named ROCKEFELLER, the variable S1 could be bound to a STOCK named STOCK-1, the variable C1 could be bound to a CORPORATION named DUPONT, and so on. More generally, the description matching procedures of the TAXMAN system have the following characteristics: (1) If there are free variables in the description, the result of the match is a list of all the bindings of those variables for which the description is true. However, (2) since we can also specify an initial bindings list as an input to the matching procedure, we can force any of these free variables to be bound initially to particular individuals. For example, we could force the variable A1 to be bound to a PERSON named CARNEGIE, in which case the previous match would fail. Taken together, these features permit us to construct a great variety of queries: “Who owns stock in the duPont corporation?” “How much common stock does Rockefeller own?” “How much stock has the duPont corporation issued?” (3) Finally, if the description cannot be fully satisfied, the matching procedure computes the “best partial match,” where the notion of a “best” match is defined implicitly by the structure of our conceptual model. Thus if we were trying to establish in our previous factual network that “Rockefeller owned a BOND issued by the duPont corporation,” the system would be able to determine that Rockefeller did indeed own a SECURITY issued by duPont, but that it was a STOCK rather than a BOND. This partial matching capability is especially important, we believe, in a legal context.

These procedures for instantiating and matching descriptions play a central role in the TAXMAN system, but the examples I have used so far to illustrate them obviously constitute a rather impoverished model of the corporate tax domain. It is not enough to describe an OWNership relation in the state TIME-1, since the important facts of a corporate tax case will depend on how these OWNership relations have changed over time, and how the actions of the various parties have effectuated these changes. Our complete model of the corporate tax domain thus makes use of the full set of description spaces illustrated in Figure 1. First, as indicated at the bottom of Figure 1, our network of facts is divided into a binary branching tree of states. Since certain facts may be true in TIME-3, but not in TIME-4, and vice versa, the system permits us to store multiple versions of these facts, as if we had taken a series of “snapshots” of the world at different times. The branching tree structure of this
PERMISSION/OBLIGATION

Situation:  <description>

Perm/Oblig: ACTION

Agent:  ACTOR

Statechange:

State1:  <description>

State2:  <description>

TIME-1 --- TIME-2 --- TIME-3 --- TIME-4 --- TIME-5 --- TIME-6 --- TIME-7 --- TIME-8

Figure 1: The Organization of Description Spaces

representation then permits us to engage in a simple form of hypothetical reasoning. We can designate some linear sequence of states—say, TIME-1, TIME-2, TIME-3, TIME-4, TIME-7—as the actual course of events, and designate each alternative branch as a course of events which might have occurred, but did not. To manipulate these hypothetical states, we need the notion of a STATECHANGE, which is also illustrated in Figure 1. A STATECHANGE is represented by a pair of descriptions, one which matches the factual network before the change, and one which matches the factual network after the change. Like the OWNership description discussed earlier, these STATECHANGE descriptions can be used in two ways: first, they can be instantiated, to generate a new state as a successor to an old state; second, given a fully instantiated tree of factual networks, they can be matched to these networks to recognize the existence of a difference between two states. In addition, the STATECHANGE descriptions can themselves be incorporated into higher level structures, as indicated in Figure 1. In this way the representation gradually acquires its complexity and power. A STATECHANGE coupled with an ACTOR gives us an ACTION. ACTIONS, in turn, are embedded in a system of PERMISSIONs and OBLIGATIONs, i.e., in a system of rules which state that if a given situation description is satisfied then a certain action is permitted, or obligatory, as the case may be. But since these rules of PERMISSION and OBLIGATION are in general dependent on time, they may themselves become part of a state description, and subject to the
actions of the STATECHANGEs, in a fully recursive manner. This permits us to represent legal powers, legal immunities, and other complex legal concepts, as suggested in the early work of Hohfeld [Hohfeld 13] [Hohfeld 17].

The system of description spaces outlined in Figure 1 is an example of what we call an “abstraction/expansion” hierarchy. At one level of the hierarchy we write out an abstraction, i.e., a summary description of a concept, and at a lower level of the hierarchy we write out the corresponding expansion, i.e., a somewhat more detailed description of the same concept. Our procedures for instantiating (or matching) descriptions are then extended to instantiate (or match) the expansions whenever we instantiate (or match) the associated abstractions. For example, suppose we wanted to represent a transfer of property from one party to another. At the ACTION level, we would write the following description:

\[
\text{(TRANS TR1)} \\
\text{<agent (ACTOR A1)>} \\
\text{<object (PROPERTY P1)>} \\
\text{<oldowner (ACTOR A1)>} \\
\text{<newowner (ACTOR A2)>} \\
\text{<time1 (STATE T1)>} \\
\text{<time2 (STATE T2)>} 
\]

which represents “the TRANSfer by an ACTOR A1 of a PROPERTY P1 to an ACTOR A2, occurring between a STATE T1 and a STATE T2.” Is this an adequate representation? If we wanted to treat the transfer of property as a primitive concept, and to talk about the participating ACTORS, and the time of occurrence, but nothing more, then the representation of a transfer as a TRANS might be perfectly acceptable. Frequently, though, we will want to describe the transfer itself as a change in OWNership, and this leads us to construct a STATECHANGE description:

In State T1: \[
\text{(OWN 01)} \\
\text{<owner (ACTOR A1)>} \\
\text{<owned (PROPERTY P1)>} 
\]

In State T2: \[
\text{(OWN 01)} \\
\text{<owner (ACTOR A2)>} \\
\text{<owned (PROPERTY P1)>} 
\]

which can be attached to the TRANS description as part of its definition. We thus have a simple example of an “abstraction/expansion” pair: the TRANS description is the abstraction; the STATECHANGE description is the expansion. If there exists a TRANSfer of property, we are saying, then there exists a change in OWNership.
In this case, as we can readily see, and similarly in the more complex cases illustrated in Figure 1, the expansion tells us part of the meaning of the associated abstraction.

Consider now how the instantiation procedures would work with these abstraction/expansion pairs. Suppose we were in the state TIME-1, above, in which John D. Rockefeller owns 1000 shares of the common stock issued by duPont, and suppose we wanted to say that "John D. Rockefeller transferred the duPont common stock to Andrew Carnegie." We would type:

\[
\text{CASE:TIME-1} \quad \text{(make}
\text{(trans}
\text{(agent rockefeller) \hspace{1cm} (make)}
\text{(find}
\text{(the stock (issuer dupont) \hspace{1cm} (find)}
\text{(common yes)))
\text{(oldowner rockefeller) \hspace{1cm} (find)}
\text{(newowner carnegie) \hspace{1cm} (find)}
\text{(time1 time-1) \hspace{1cm} (find)}
\text{(time2 time-2))}
\]

and the system would create an instance of a TRANS, in exactly the same way that it previously created an instance of a CORPORATION, a STOCK, and an OWN. However, since the TRANS is also an ACTION, and since it has a STATECHANGE description, i.e., a change of OWNership, as its definitional expansion, this STATECHANGE description would be instantiated, too. This means that the system would first match the old OWNership description in TIME-1, to verify that the STATECHANGE was possible, and then, assuming that the match was successful, it would generate the new OWNership description in TIME-2. If we subsequently examined the network of facts in TIME-2, we would see that Andrew Carnegie, rather than John D. Rockefeller, was now the owner of 1000 shares of duPont common stock. The TAXMAN system would have simulated the action by updating the factual network in the appropriate way.

The converse operation, the matching of a description to a factual network, likewise follows the logic of the abstraction/expansion hierarchy, and this enables us to carry out a limited form of legal "analysis." Suppose we have already instantiated, by the use of the preceding techniques, the facts of a corporate transaction: the issuance of stock, the transfers of property, the distributions to stockholders, etc. We now construct an abstract concept which has legal consequences, for example, the concept of a tax-free "Type B" Reorganization:
The "Type B" Reorganization is defined in the United States Internal Revenue Code, Section 368(a)(1)(B), as:

The acquisition by one corporation, in exchange solely for all or a part of its voting stock . . ., of stock of another corporation, if, immediately after the acquisition, the acquiring corporation has control of such other corporation. . . .

and so, in the TAXMAN system, we would write the definitional expansion of a BREORGANIZATION as follows:

(ACQUISITION ACQ1
  <acquirer (CORPORATION C1)>
  <acquiredprop
    <STOCK S1
      <issuer (CORPORATION C2)>>
    <transferredprop
      <STOCK S2
        <issuer (CORPORATION C1)>
        <voting YES>>>
    <time1 (STATE T1)>
    <time2 (STATE T2)>)

(CONTROLL CON1
  <controller (CORPORATION C1)>
  <controlled (CORPORATION C2)>
  <time (STATE T2)>)

Note that this representation is somewhat similar to the English version: if there exists an ACQUISITION of STOCK in exchange for VOTING STOCK, and if there exists a CONTROL relationship following the acquisition, then there exists a BREORGANIZATION. The abstraction/expansion hierarchy cannot stop here, though, since ACQUISITION and CONTROL are not themselves primitive concepts. The ACQUISITION description is thus given a further definitional expansion in terms of a sequence of EXCHANGES, and the EXCHANGE description is given an expansion in terms of a pair of reciprocal TRANSfers. The CONTROL relationship is likewise given a definitional expansion, following Section 368(c) of the Internal Revenue Code, as the OWNership of 80% of the STOCK of the controlled CORPORATION. Given this complete abstraction/expansion hierarchy, then, what happens when we attempt to match the upper level BREORGANIZATION pattern to the facts of a
particular corporate transaction? First, the system checks to see if there is an instance of a BREORGANIZATION stored explicitly in the factual network, but, if this search fails, the system then attempts to find an instance of an ACQUISITION, and an instance of a CONTROL, and so on, down the hierarchy. Eventually, the system either succeeds, and returns an analysis of the transaction as a BREORGANIZATION, or it fails, and returns the reason for its failure in the form of a partial match result. For a more detailed discussion of these procedures, see [McCarty 77] and [McCarty and Sridharan 80].

Readers familiar with our recent papers on the TAXMAN project will recognize the BREORGANIZATION definition as an example of what we call the logical template representation of a legal concept: logical, because the expansion is a logical expression giving necessary and sufficient conditions for the existence of the abstraction; template, to suggest the way in which this logical expression is matched to the lower level factual network during the analysis of a case. We are not claiming, though, that this representation provides us with an adequate theory of legal reasoning. Quite the contrary. In fact, we have developed in our recent papers an alternative representation, which we call the prototype-plus-deformation representation, to account for the “open-texture” of most legal concepts, and to account for the incremental “construction” and “modification” of legal concepts which occurs during the analysis of a difficult case. See [McCarty and Sridharan 81] [McCarty 82]. But the prototype-plus-deformation representation is still primarily an aspect of our theoretical research. It is not yet ready for practical applications. By contrast, we believe that the logical template representation, though inadequate as a matter of legal theory, is adequate enough for the practical task of building intelligent legal information systems. There are many important legal domains, mostly in the corporate and commercial fields, in which the “facts” of a case can be expressed quite adequately within a representation similar to the TAXMAN representation, and in which the legal “rules” and “concepts” can be expressed quite adequately in a form similar to that of the BREORGANIZATION definition. For an earlier discussion of this point, see [McCarty 80]. There are the areas of the law which are now ready for the practical applications of artificial intelligence techniques.

Before turning to a discussion of these applications, it may be helpful to comment briefly on the prospects for natural language communication with a system such as TAXMAN. Although we are not currently building a natural language “front end” to our system,
our conceptual representations have been designed in such a way as to make this task as manageable as possible. To see this, consider the ACTION description used in our earlier example: "John D. Rockefeller transferred the duPont common stock to Andrew Carnegie." Suppose we wanted to convey this information to the system in English, instead of in the TAXMAN formalism. The techniques for writing a grammar, and a parser, to analyze this sentence syntactically are now well known: see, e.g., [Marcus 80] [Winograd 83]. The main verb is "transfer," which corresponds to a TRANS in our domain, and the syntactic subject is "John D. Rockefeller," which corresponds to the agent of the TRANS. The remainder of the verb phrase consists of a noun phrase ("the duPont common stock") which corresponds to the object of the TRANS, and a prepositional phrase ("to Andrew Carnegie") which corresponds to the newowner of the TRANS. In other words, as one can see from an inspection of our earlier example, it is possible, indeed, relatively easy, to write a program which translates from a syntactic analysis of the English sentence directly into the input language of the TAXMAN system.

This is a trivial example, of course. What are the prospects for handling more complex examples? As one might expect, no complete grammar of English has ever been written, but large subsets of English grammar have been written, large enough for substantial practical applications. Experience has shown, in fact, that the most intractable problems are semantic, rather than syntactic. In other words, if the conceptual model of the domain of application has been adequately specified, as it has been in the TAXMAN system, then it is possible to construct a very acceptable natural language interface, with only a minimal simplification of English syntax. This is no longer a laboratory curiosity, but a commercial reality: see, e.g., [Harris 79]. For practical purposes, then, there are no insuperable obstacles to the development of a legal information system, based on the TAXMAN representation, which could communicate with a lawyer in English.

Conceptual Legal Retrieval Systems

Our first proposed application is the development of a conceptual legal retrieval system, as distinguished from the full-text key-word systems now on the market. The basic idea here is to use the conceptual model of a particular legal domain to build up a data base 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 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 and concepts were applied, or not applied, in each particular case, in the same way that the BREORGANIZATION definition was applied in our previous example. 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, and analysis patterns, etc. The advantages of this system should be readily apparent. By comparison to the existing key-word searches, 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. The existing key-word systems would not be replaced by the conceptual systems, of course, but only augmented by them. We would still preserve our full-text database, for browsing, if nothing else, and we would still preserve the key-word search as an alternative access technique. But the conceptual search would add a new dimension to legal retrieval.

The only extended example, so far, of a conceptual legal retrieval system appears in the dissertation of Carole Hafner, completed at the University of Michigan in 1978, and published recently as a monograph by the University of Michigan Press [Hafner 81]. 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, Hafner constructed a database consisting of approximately 200 cases and 200 subsections of the Code, plus a query language which permitted her to search for certain conceptual patterns in the data. For example, for the case of Jackson v. First National Bank of Memphis, 403 S.W.2d 109 (1966), Hafner stored the facts that the

Holds that there was a CHECK-IMPROPERLY-DEDUCTED,

- because of the CHECK that had not been HANDLED-WITH-REASONABLE-CARE
• and because of the DRAWERS-OR-MAKERS-SIGNATURE that was FORGED,
• and despite the RETURNED-TO-CUSTOMER CHECK that had not been EXAMINED-PROMPTLY;
• citing as a basis UCC3-406 and UCC4-406;

and this information was stored in Hafner's data base using a formal language which was fully equivalent, semantically, to the simplified English shown here. Several kinds of queries were then possible. For example:

FIND all cases
• about a FORGED INDORSEMENT,
  • in which the PLAINTIFF was not a HOLDER-IN-DUE-COURSE,
  • and such that, if the case decided that a SIGNATURE was FORGED, then the DEFENDANT was not a BANK.

a query which fails to match the case of Jackson v. First National Bank, in two respects. It is important to note here that Hafner's system, as implemented, did not accept queries in English, but it would be perfectly feasible to add an English language "front end" to the query interpreter, using the techniques outlined in the previous section of this paper. Again, the critical problem is the development of an adequate conceptual model of the legal domain.

How complete is Hafner's conceptual model? Obviously there are some differences between the representation used in the negotiable instruments domain and the representation used in the TAXMAN domain. One difference, apparent in these examples, is that Hafner chose to express most complex concepts, such as a "drawer's or maker's signature," as single atomic terms, whereas the TAXMAN representation would express these complex concepts as structured descriptions. Another difference can be seen in the treatment of events. Hafner's system contains no representation of time, and no representation of a change of state, but instead represents an event as an atomic attribute attached to the appropriate object. A CHECK thus has the possible attributes RETURNED-TO-CUSTOMER and EXAMINED-PROMPTLY, but there is no way to describe the action of "returning" or "examining" the check itself.

To facilitate comparison between these two systems, let us translate a small portion of Hafner's negotiable instruments model into the TAXMAN notation. As before, we will create a class of ACTORS, who can be either PERSONs or CORPORATIONs, but we will also create a special subclass of CORPORATIONs known as
BANKS. In place of the hierarchy of property rights, we will create a class of INSTRUMENTs, which can be "negotiable" or not, and which, if negotiable, will have a "payee," one or more "indorsers," and a "holder," all of whom must be ACTORS. The negotiable INSTRUMENTs will have two special subclasses: a NOTE, which has a "maker," who must be an ACTOR; and a CHECK, which has both a "drawer," an ACTOR, and a "drawee," a BANK. Finally, we will create a special object called a SIGNATURE, which can be "forged" or not, which will always be a "constituent of" an INSTRUMENT, and which will always have a "signer" who is an ACTOR. In these terms, the checks at issue in Jackson v. First National Bank of Memphis would be represented by instantiating the following description:

(CHECK CH1
   <payee (ACTOR A1)>
   <drawer (ACTOR A2)>
   <drawee (BANK A3)>)

in which the payee A1 would be one Cleve Jordan, the drawer A2 would be the St. Matthew's Church of Memphis, and the drawee A3 would be the First National Bank. Significantly, there would be two authorized signatures on this check, one of them forged:

(SIGNATURE S1
   <forged YES>
   <constituentof (CHECK CH1)>
   <signer (ACTOR A4)>)

but ostensibly signed by Milton Jackson, the Trustee of the Church, and the other one valid, but signed by Cleve Jordan, the same person who is also the payee! Note, however, that this important structural information about the case was completely obscured by the original representation of a FORGED DRAWERS-OR-MAKERS-SIGNATURE. As these examples indicate, then, Hafner's conceptual model is neither as finely detailed nor as fully decomposed as the conceptual model of the TAXMAN system. But this is not in itself a disadvantage. Indeed, Hafner argues persuasively that it is essential in a conceptual legal retrieval system to sacrifice some depth in the representation to achieve breadth in the coverage of the law, and I consider it a very important research issue to determine exactly how to strike this compromise, as a practical matter, in various legal domains.

The most critical research issue, however, concerns another problem facing the designer of a conceptual legal retrieval system. 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. Also, since the system would have access to all the previously abstracted cases, it should be able to "prompt" the abstractor for any additional facts that might be relevant. Balanced against these advantages of consistency and completeness, though, are the problems which would arise whenever a new case fails to fit within the existing conceptual model. Clearly an abstractor must have the option in this situation to "escape" to a human supervisor, someone who can revise and augment the conceptual model as necessary, but the precise mechanisms for accomplishing these revisions, and at the same time maintaining the consistency of the data base, would require a great deal of additional research. 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 [Cullingford 78] [DeJong 79], which can "read" the Associated Press wire and extract from it sketchy information about earthquake stories, assassination stories, etc., in several discrete news categories. Perhaps a similar system could "read" the statement of facts in a judicial opinion and extract from it some sketchy information about the distribution of stock, the forging of a signature, etc., in several discrete legal categories. 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.

It should be clear from this discussion that the construction of a conceptual legal retrieval system will not be a trivial task, and that it will be helpful, over the next few years, to initiate a number of research projects in this field, as a source of ideas and techniques. Two additional projects should be mentioned here, neither of which is yet complete. One is the POLYTEXT/ARBIT system developed jointly by SRI International, in California, and the Kval Institute for Information Science, in Sweden [Loef 80]. The POLYTEXT/ARBIT system responds to requests, in English, to locate passages in a legal
document, specifically, a translation of the Rules of the Arbitration Institute of the Stockholm Chamber of Commerce. Some of these requests trigger searches which are fairly common in retrieval systems: a search for index terms, for example, and a search through the hierarchical structure of the text. But the system can also search through a "propositional model" of the arbitration rules, and this is similar to a search through the conceptual model of a legal domain, in the TAXMAN terminology. A second project, only recently initiated, is the endeavor of Cary deBessonet at the Louisiana Law Institute, in Baton Rouge, to build a conceptual representation of several sections of the Louisiana Civil Code, using artificial intelligence techniques. One of the main purposes of deBessonet's project is to clarify the structure of the Civil Code itself, but an additional purpose, once the conceptual model has been formulated, is to experiment with alternative designs for a conceptual legal retrieval system. Hopefully, we will see many more projects of this sort in the near future.

**Legal Analysis and Planning Systems**

Our second proposed application is the development of a legal analysis and planning system. The main ideas here can also be explained by reference to the TAXMAN examples. First, imagine a system which accepts as input the description of a corporate transaction, and provides as output a classification of the transaction in terms of various legal categories, as in the BREORGANIZATION example discussed earlier. 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 through the space of possible transaction patterns to suggest a course of action which satisfied additional constraints, e.g., minimal tax consequences. The techniques for searching for a "plan" of this sort have been studied extensively in the artificial intelligence literature, see, e.g., [Sacerdotti 77] [Stefik 81a] [Stefik 81b] [Sridharan and Bresina 82], and it is known that efficient solutions of the "planning problem" often depend critically on the way we construct the conceptual representation of our domain. However, it turns out that there are ways to write "legal analysis and planning systems" without the use of any conceptual models, and this raises questions about whether the more
complex artificial intelligence techniques are really necessary. In this section, I will examine these simpler approaches and compare them to the approach used in the TAXMAN project.

Let us first examine a legal analysis and planning system which makes no claim at all to be based on artificial intelligence techniques. A good example is Robert Hellawell’s CORPTAX system for analyzing the taxation of stock redemptions [Hellawell 80]. The CORPTAX system asks the user a series of questions about the taxpayer’s interest in the corporation which is redeeming its stock, and then provides an analysis of the legal consequences of the transaction. A typical question might be the following:

- Does TP and family actually own 50% or more (in value) of the stock of a corporation which is a partner in a partnership which owns Redcorp stock?

and after the user has answered a series of such questions, the system might respond as follows:

Redemption does not qualify under I.R.C. Sec. 302(b)(2).
Requirement of I.R.C. Sec. 302(b)(2) is not met.

As a technical matter, the program which produces this result is exceedingly simple. It is written in BASIC, and it consists almost entirely of PRINT statements to pose the questions, plus IF/THEN statements and GO TO statements to respond to the answers. The difficult task in writing such a program, though, as Hellawell points out, is the task of analyzing the legal rules in such detail that they can be cast into this form. For it is necessary for the author of the CORPTAX system to anticipate all possible answers to the questions, and all possible interactions among these answers, and then to provide an explicit logical branch within the program for each such combination of answers. In this respect, the CORPTAX system resembles the classical systems for computer-assisted instruction (CAI), in which the author is likewise forced to program explicitly all possible branches through an instructional curriculum.

Now I have no doubt that it will be possible to write programs like CORPTAX in various areas of the law, and it seems quite likely that some of these programs will be offered as commercial products in the foreseeable future. However, there are some limitations to these programming techniques which should be kept in mind. First, as the experience with the classical CAI systems has shown, the interaction between the user and the program here is extremely rigid. There are, as a practical matter, only a few fixed paths through the statute, and
the program controls absolutely the order in which these paths are explored. By contrast, in a system based upon artificial intelligence techniques, and built around a conceptual model of the legal domain, the user can initially provide whatever information seems relevant, and the system can ask for further information only when a preliminary analysis shows that it is needed. The TAXMAN system provides some hints about how this can be done, but for a complete example of the use of these techniques to construct a "mixed initiative" CAI system, see [Brown and Burton 75]. A second limitation of the CORPTAX system arises from the fact that it contains no explicit representation of the legal rules, but instead represents rules implicitly by the patterns of IF/THEN and GO TO statements. This means that it will often be difficult to determine if a legal rule has been correctly stated, or if the conclusion reached by the program in applying the rule in a particular case is correct. Finally, and closely related, there is the difficulty of modifying the CORPTAX system whenever the legal rules are modified. By contrast, in a system such as TAXMAN, the representation of legal rules is explicit, declarative, and modular, and a rule can be modified in most cases by modifying a single data structure.

Another system in existence today, in which the legal rules are also explicit, declarative, and modular, is the ABF Processor developed by James A. Sprowl at the American Bar Foundation in Chicago [Sprowl 79]. The ABF Processor can be understood initially as a sophisticated system for drafting form legal documents. An attorney using the system can prepare a form sales contract, for example, and incorporate instructions within the form itself to "fill in the blanks" with the names of the parties and the items being sold. The system is capable of much more than this, however, since the library of documents can include a set of IF/THEN rules approaching the complexity of a statute. To cite an example from Sprowl's original paper, consider a petition for probate which contains the following conditional clause:

[IF the estate may be probated without court appearance INSERT
   Petitioner requests permission to ... 
   ENDIF]

By itself, this clause would generate a query to the user to determine if "the estate may be probated without court appearance," and if the answer to the question was YES, the system would insert the brack-
eted text into the final document. But suppose the following rule had been stored in the probate library along with the form document:

IF

the decedent’s children and the named beneficiaries ARE of sound mind and over 20 years old

AND

the named executor IS a resident of the state of Illinois

AND

the decedent HAS waived security on the executor’s bond in the will

AND

the decedent’s children and the named beneficiaries HAVE consented to an out of court settlement

THEN

the estate may be probated without court appearance

The system would then notice that the consequent of this rule matched the antecedent of the rule appearing in the petition for probate, and it would proceed to ask the questions listed above: “Are the decedent’s children and the named beneficiaries of sound mind and over 20 years old?” and so on. Furthermore, since these IF/THEN rules can be chained together to an arbitrary depth, it is possible to build a system of substantial complexity using these techniques. For an example of such a system, capable of drafting security agreements under Article 9 of the Uniform Commercial Code, see [Boyd and Saxon 81].

To place the ABF Processor Language into perspective, it will be helpful to understand how the rule representation outlined above resembles the representation of the TAXMAN system, and also how it differs. The backward chaining from consequent to antecedent in the ABF Processor is similar to the top-down pattern matching from BREORGANIZATION, to ACQUISITION and CONTROL, and so on, in the TAXMAN system, and both of these are instances of a standard problem solving technique in the artificial intelligence literature, see, e.g., [Kowalski 79]. But the rules in the ABF system are entirely propositional. The pattern matching is performed, character by character, on the entire textual phrase appearing in the consequent of the rule. This means, for one thing, that the phrase “without court appearance” would fail to match a phrase reading “without a court appearance,” although this difficulty would rarely arise in practice because of the interactive manner in which these systems of rules are usually constructed. More significantly, this means that a rule in the ABF system has no internal semantic structure, and that means, in turn, that it will be very difficult to capture
the conceptual organization of a legal domain within this representation. There is a hint of this difficulty in Boyd and Saxon's discussion of several useful extensions to their program for drafting Article 9 security agreements [Boyd and Saxon 81]. After pointing out the importance of providing a system of “warning messages” to guide the user throughout the drafting process, Boyd and Saxon write (p. 655):

A truly effective assistance and caveat scheme must be reasonably complete. This means that a rather extensive data base will ultimately be called for. . . . The scheme that tends to fit most naturally into the A-9 program as augmented by the assistance and caveat concept is one that is constructed around the transactions involved. Information would be stored and retrieved according to an algorithm . . . which reflects the structure of the underlying transaction. The structure of the transaction, in turn, would be largely a function of the statutory framework that defines the secured transaction arrangement.

This idea of building a caveat scheme around the underlying statutory transaction patterns is an excellent one, in my opinion, and very much within the spirit of contemporary work on the applications of artificial intelligence techniques to database management. But it is difficult to see how this can be done without the construction of a rather deep conceptual model of the Article 9 domain.

This point brings us back to the contrast between deep models and shallow models which I introduced at the beginning of this paper. In these terms, Sprowl's ABF Processor, and even Hellawell's CORPTAX program, are examples of shallow rule-based expert systems, somewhat akin to the MYCIN system. The comparison is even clearer if we include here two additional legal systems, both of which were constructed by applying a general framework for expert systems research, originally developed in the medical field, to a legal problem domain. The first is a system called TAXADVISOR, developed by Robert H. Michaelson as part of his dissertation in Accountancy at the University of Illinois [Michaelson 81]; this is an application of the EMYCIN (or “Empty MYCIN”) framework to the problem of providing investment advice. The second is the Legal Decisionmaking System, LDS, developed by Don Waterman and Mark Peterson at the Rand Corporation [Waterman and Peterson 80] [Waterman and Peterson 81]: this is an application of the ROSIE programming language, a distant descendant of MYCIN, to the problem of settling claims in product liability cases. Since Waterman and Peterson are primarily interested in simulating a large collection of product liability decisions, in order to analyze the effects of alternative liability rules, they typically apply their rules bottom-up,
working forward from antecedent to consequent, rather than top-down, as in MYCIN. If the LDS system were to be used for an expert consultation, however, as Waterman and Peterson have occasionally suggested [Waterman and Peterson 80], then a top-down control structure would be essential, for a lawyer would never be able to enter the lower-level facts of a case, with enough precision to trigger the antecedent components of the rules, unless he was guided by a detailed conceptual model of the product liability domain. Thus the TAXADVISOR system, the LDS system running in a consultation mode, and the original MYCIN system are essentially equivalent. All three systems have the same expressive power and the same basic control structure as the ABF Processor.

There appears to be a paradox here, however. If the shallow rule-based systems have been so successful in the medical field, why do I insist that they have such serious limitations in the legal field? The answer has to do with some of the differences between medical reasoning and legal reasoning, and the different role played by a system of “rules” in each profession. The rules built into the MYCIN system are actually “rules of thumb”: they are empirical, associative, probabilistic, and they contribute to a medical diagnosis in a cumulative fashion. Since it is difficult for the human mind to weigh the effects of a large number of such rules in any particular case, the computer system performs, by comparison, surprisingly well. Presumably, a diagnosis that was based on a detailed causal model of the disease mechanisms would be more accurate than a diagnosis based on these judgmental rules, but it turns out that the mechanisms of bacterial infection are so poorly understood that no such causal model exists. For a discussion of this point, see [Kulikowski and Weiss 82]. By contrast, the only legal rules which exhibit some of the characteristics of the MYCIN rules are those which contain the most discretionary legal concepts, such as “reasonable care,” and these rules are unlikely to be the subject matter of an expert system. All other legal rules have a logical structure, rather than an associative structure; they are deterministic, rather than probabilistic; and they apply individually, rather than cumulatively. When these rules are encoded into the MYCIN framework, their internal logical structure is obscured, and the resulting system is identical to Sprowl’s ABF Processor: It becomes a top-down interpreter for a purely propositional logic.

Despite these limitations, the fact remains that systems such as Sprowl’s ABF Processor can be made available, today, as commercial products, while systems such as TAXMAN are still the subject of university and government sponsored research. The situation is the
same in the other areas of expert systems development: medicine, geology, and circuit design. Peter Hart writes:

Put differently, surface systems are the ones that currently "work." Most practical applications systems, or impressive demonstrations that appear to be close to applications, are likely to be surface systems. Deep systems tend to be the research vehicles of the field; they may be nameless, slow, and are likely to deal with very restricted problem domains. [Hart 82]

The challenge, then, is to transform the research vehicles of the field, the systems based on deep conceptual representations of a domain, into practical techniques for analysis and planning in the real legal world. In certain areas of the law, such as tax planning and estate planning, this seems to me to be a feasible goal for the very near future. I would thus expect to see a substantial amount of resources committed to this effort in the next several years.

Conclusion

This paper has proposed an ambitious research programme: To fully exploit the benefits of advanced computer science techniques in legal information systems, I have argued, it is necessary to build a conceptual model of the relevant legal domain. But this will not be an easy task. It will be necessary, at least initially, to make some compromises, as I have indicated, and to select a level of conceptual detail appropriate to each application. Indeed, an investigation of the tradeoffs between deep and shallow models, in each legal domain, would be an important aspect of the proposed research.

I have analyzed the prospects for this research programme in two kinds of applications: (1) conceptual legal retrieval systems, and (2) legal analysis and planning systems. But it is not necessary to keep these two systems separate. Implicit in my analysis is the fact that the same conceptual model could serve as the foundation for both. We have seen, for example, how the TAXMAN model could be used to describe a set of corporate tax cases in a conceptual data base, and how it could also be used to define the search space for a sophisticated tax planning system. There would be distinct advantages in combining these two functions. 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 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 conceptual rules of the legal analysis system. Neither of these advantages could be achieved in a
legal analysis system based on propositional rules, or in a retrieval system based on key words. Thus, despite the difficulties in constructing and applying the initial conceptual model, the ultimate benefits from an "intelligent" legal information system could be quite extraordinary. On these grounds alone, the research programme would appear to be justified.
References

[Allen 63]
Allen, L.E.
Beyond Document Retrieval Toward Information Retrieval.

[Bauer-Bernet 82]
Bauer-Bernet, H.
Legal Thesauri and Data Processing.

[Boyd and Saxon 81]
Boyd, W.E. and Saxon, C.S.
The A-9: A Program for Drafting Security Agreements Under Article 9 of the Uniform Commercial Code.

[Brown and Burton 75]
Brown, J.S. and Burton, R.R.
Multiple Representations of Knowledge for Tutorial Reasoning.

[Ciampi 82]
Ciampi, C.
Artificial Intelligence and Legal Information Systems.

(Council of Europe 81)
Council of Europe.
Proceedings of the Sixth Symposium on Legal Data Processing in Europe.

[Cullingford 78]
Cullingford, R.
Script Application: Computer Understanding of Newspaper Stories.
[DeJong 79]
DeJong, G.
*Skimming Stories in Real Time.*

[Duda, et al. 79]
Duda, R., Gaschnig, J. and Hart, P.
Model Design in the PROSPECTOR Consultant System for Mineral Exploration.

[Hafner 81]
Hafner, C.D.
*An Information Retrieval System Based on a Computer Model of Legal Knowledge.*

[Harris 79]
Harris, L.R.
Experience with ROBOT in 12 Commercial Natural Language Data Base Query Applications.

[Hart 82]
Hart, P.E.
Directions for AI in the Eighties.
ACM Special Interest Group in Artificial Intelligence.

[Hellawell 80]
Hellawell, R.
A Computer Program for Legal Planning and Analysis: Taxation of Stock Redemptions.

[Hohfeld 13]
Hohfeld, W.N.
Fundamental Legal Conceptions as Applied in Judicial Reasoning: I.

[Hohfeld 17]
Hohfeld, W.N.
1983] LEGAL INFORMATION SYSTEMS 291

Fundamental Legal Conceptions as Applied in Judicial Reasoning: II.

[Knapp 82]
Knapp, V.
Legal Thesauri.

[Kowalski 79]
Kowalski, R.
Logic for Problem Solving.

[Kulikowski and Weiss 82]
Kulikowski, C.A. and Weiss, S.M.
Representation of Expert Knowledge for Consultation: The CASNET and EXPERT Projects.
In Szolovits, P. (editor), Artificial Intelligence in Medicine, pages 21-55. AAAS, 1982.

[Loef 80]
Loef, S.
The POLYTEXT/ARBIT Demonstration System.

[Marcus 80]
Marcus, M.P.
A Theory of Syntactic Recognition for Natural Language.

[Martino 82]
Martino, A.A.
Deontic Logic, Computational Linguistics and Legal Information Systems.

[McCarty 77]
McCarty, L.T.
Reflections on TAXMAN: An Experiment in Artificial Intelligence and Legal Reasoning.
[McCarty 80]
McCarty, L.T.
Some Requirements for a Computer-Based Legal Consultant.

[McCarty 82]
McCarty, L.T.

[McCarty and Sridharan 80]
McCarty, L.T., and Sridharan, N.S.

[McCarty and Sridharan 81]
McCarty, L.T., and Sridharan, N.S.

[Michaelson 81]
Michaelson, R.H.
A Knowledge-Based System for Individual Income and Transfer Tax Planning.

[Niblett 80]
Niblett, B.
Computer Science and Law: An Advanced Course.

[Niblett 81]
Niblett, B.
Expert Systems for Lawyers.
[Sacerdotti 77]
Sacerdotti, E.D.
A Structure for Plans and Behavior.

[Shortliffe 76]
Shortliffe, E.H.
Computer-Based Medical Consultations: MYCIN.

[Skelly 83]
Skelly, S.K.
The Impact of Technology on Legal Information Processing and Some of the Ramifications.

[Sprowl 79]
Sprowl, J.A.
Automating the Legal Reasoning Process: A Computer that Uses Regulations and Statutes to Draft Legal Documents.

[Sridharan 81]
Sridharan, N.S.
Representing Knowledge in AIMDS.
Informatica e Diritto 7:201-221, 1981.

[Sridharan and Bresina 82]
Sridharan, N.S. and Bresina, J.L.
Plan Formation in Large, Realistic Domains.
University of Saskatchewan, 1982.

[Stefik 81a]
Stefik, M.J.
Planning with Constraints (MOLGEN: Part 1).

[Stefik 81b]
Stefik, M.J.
Planning and Meta-Planning (MOLGEN: Part 2).
Artificial Intelligence 16(2):141-170, 1981.
[Waterman and Peterson 80]
Waterman, D.A., and Peterson, M.A.
Rule-Based Models of Legal Expertise.

[Waterman and Peterson 81]
Waterman, D.A. and Peterson, M.A.
*Models of Legal Decisionmaking*.

[Weiss, et al. 78]
Weiss, S., Kulikowski, C., Amarel, S. and Safir, A.

[Winograd 83]
Winograd, T.
Addison-Wesley, 1983.