Information Technology and Legal Education: Towards 2000

9th & 10th April 1992
Sponsored by
Context Ltd; Linklaters & Paines; Needham & James

MASTER and ESCAPE
A Legal Expert System Shell and Application

P.T.J. Cattell and I.A. Wilson

Queensland University of Technology, Australia

Keywords: Expert Systems - Evidence - Knowledgebases - Shells - Commercial Law -Bills of Exchange

Abstract: This project was initially known as EXILE (Expert System in Legal Evidence). It was first reported in the paper, "EXILE - Expert Information in Legal Evidence" published by the Department of Computer Science, University of Reading (1988). It consisted of two distinct components - a specially developed expert system shell for legal applications and its first application to the law of evidence in Queensland. This project has now been substantially enhanced and these enhancements are described in this paper. The project is part of a continuing research program at the Queensland University of Technology and is a co-operative venture between the Faculties of Information Technology and Law.

1.0 Introduction

In 1988 Peter Cattell, Jon Machtynger and Ian Wilson published a research paper entitled "EXILE: Expert Information in Legal Evidence" (University of Reading, Department of Computer Science)1. The EXILE expert system shell and application at that time married an inference engine and user interface expressed in Prolog with a rulebase relating to documentary hearsay provisions in the Queensland Evidence Act. The system comprised a small number of rules, explanation and dictionary statements and operated only on Sun UNIX workstations.

The major goal of the EXILE project was to explore the issues involved in building a legal decision-making adviser. The participants set out to and did successfully construct a system which satisfied three criteria. It had to emulate an expert in his or her problem solving activity, seeking relevant information from the user. It had to explain how it went about the problem solving task, responding to enquiries as to why any given action was taken. Finally, it had to justify the conclusions it reached by reference to empirical data. The project to this point is described more fully in the authors' paper: "EXILE - Expert Information in Legal Evidence" presented at the 5th Bileta (British and Irish Legal Education Technology Association) Conference, University of Warwick, Coventry, UK (11-12

April, 1990)2.

From that time a number of improvements were made to EXILE. Principal among these was that the implementation language was altered from Prolog to C. This was to prove significant, for it enabled EXILE to reside entirely on a standard 720kb format 3.5' diskette which could be used on any IBM PC or equivalent machine. Thus, it became possible to project the development of EXILE into the commercial arena of the law office.

With that in mind, a series of projects was undertaken in order to expand the EXILE knowledge base. Additional evidence material was incorporated, so that from being confined to three sections of the Evidence Act the system became a comprehensive adviseron all issues related to hearsay, both statutory and common law, at Federal and Queensland State level. A new module was also created known as WASP (Wise Adviser on Suitable Procedure), which contains material related to obtaining, under the Queensland Supreme Court Rules, judgment in default of appearance and judgment in default of pleading in civil actions. WASP itself was in turn expanded, so that it also has the ability to advise on issues related to service of process out of the jurisdiction under both Federal and Queensland State rules.

During the last two years two significant projects have been completed which are the subject of this paper. Firsily, the EXILE shell has been revised and upgraded into a new system called MASTER. The first domain to be implemented using MASTER relates to actions on cheques under the Cheques and Payment Orders Act 1986 (Cth) and this application is titled ESCAPE.

2.0 MASTER: An Introduction

MASTER accesses a knowledgebase created using Boolean constructs: this operates in conjunction with explanation and dictionary functions. It is possible to construct rules for MASTER using IF, AND, AND NOT, OR, OR NOT, THEN and THEN NOT statements.

No attempt is made here to justify or explain the expert system approach to modelling legal expertise: this is covered in detail in our first BILETA paper (supra)3 and further again in (Walker,1983)4. A similar comment applies to the first application of EXILE to the law of evidence. Our aim in this paper is to describe subsequent enhancements and the current state of the project.

2.1 Re-Implementation of the Shell In C to achieve portability and speed-up

As one prong of development since the first paper was presented, the Expert System shell has been rewritten in C and considerably enhanced. The prototype shell was originally written in Prolog over a two year period to test the feasibility of the proposed model for implementing a system in the chosen field of law. At the time of the first paper, the testing of the model had reached a satisfactory stage and work then began towards production versions of the shell which would find acceptance in the legal community.

The prototype version suffered from two drawbacks. It was not adequately portable and the response time to enquiries made by a user was often too long.

Firstly, the implementation language chosen for the initial feasibility studies, Prolog, was not easily

portable between different machines. While there are a number of Prolog translators available for the PC/'x86 family and a number also available for UNIX-based machines, there does not exist a satisfactory well-documented standard for Prolog other than that of 'Edinburgh Prolog' which is not well supported.

Thus, the successful strategy used by the shell of EXILE was re-implemented in 1991/92 in the C language As C compilers are available for virtually every UNIX operating system environment, this opened up the major work-station markets for the final product. C is also available for the PC /'x86 family which, apart from system calls, conforms to a very similar standard.

Secondly, the response time to user enquiries was too slow, reflecting the interpretive nature of most Prolog translators, the inefficiencies of backtracking and the making and breaking of instantiations as the only equivalent of value assignment. This weakness was addressed by taking advantage of a fully compiled version of 'C' to achieve a much needed speed-up in execution. The new shell now gives immediate response for almost all user interactions.

2.2 lmprovements to the User Interface

The user interface allows communication between the system (typically the inference engine) and the user. The Menu system when accessed through the Rule Editor allows the developer to create the necessary menu selections. When accessed from the Inference Engine, it provides an easy means of communicating with the Rule Base.

On a colour display, the desired colours are user selectable from within the Setup program. This is a separate program used to control the colour selection and the items on the main system menu. It is not necessary to have a colour display, as the MASTER shell will work equally well on a monochrome display.

In addition to this, other major improvements were made to the user interface. For the package to be easily usable by legal professionals, it needed to have an interface which is not-only 'user-friendly' but also 'intelligent'. At the current phase of development, it is expected that a lawyer with little computer training should be able to generate new rules and modify and delete existing ones. This is not at all a trivial undertaking as every new rule (or variation of a rule) needs to be supported by the appropriate question-answering dialogue, the correct explanation support and correct inferencing strategy.

A utility program to achieve the above (i.e. manipulation of the knowledgebase) has been developed. This program allows users to examine, create or modify rules and text explanations via a full-screen interface. As such, maintenance of the knowledgebase is greatly simplified, and users do not need to have any knowledge of how the knowledgebase is constructed or stored on the disk.

The user interface has also been enhanced to present a standard "Look and Feel" throughout the system. This functionality allows new knowledgebases to be developed by users with little or no programming knowledge. The enhanced menu system allows the knowledgebase developer to create the menus dynamically from within the Rulebase Editor. This means that although the option to create the rule bases with any text editor or word processor still exists, the Rulebase Editor MUST be used to create the Menus for the knowledgebase.

As an aid for new users, on-screen 'Help' is provided wherever possible. This facility is implemented as a separate function and gives the shell an ability to have more expressive and more frequent on-line help. The rule editor help screen has also been changed to this method of operation.

2.3 Additional Functionality for the Inference Engine

The Inference Engine operates on a knowledgebase. The knowledgebase consists of the Rules and the Text Explanations. Also available, but not a critical part of the knowledgebase, is a Dictionary function, which provides elaborations on terms and phrases used.

2.3.1 Structure of Rules

The structure of the rules as stored in the rule file is described in this section. The rule file is a standard ASCII text file.

The first block of the rule file specifies all the root nodes of DisjointTrees. There is no limit to the number of Disjoint Trees in a given knowledgebase, with the only restriction being available memory. This first block takes the following format.

```
TREE

<name of root node for tree 1>

<name of root node for tree 2>

.

.

.

END TREE
```

For example, three of the disjoint trees in the expanded EXILE knowledgebase are 'statement is admissible', 'computer statement is admissible' and 'entry is admissible'. These would be expressed in the following form.

```
TREE

statement is admissible

computer statement is admissible entry is admissible

END TREE
```

The block on Disjoint Trees is followed by all the rules in the knowledgebase. The rules must be separated from the Disjoint Tree block by at least one empty line, and each rule must also be separated from the next by at least one empty line. Each rule can take on one of four structures.

The first AND structure is shown below. OR and NOT structures are also implemented.

```
 AND rule

     IF <condition-i>

     AND <condition-2>

     AND <condition-3>

    .

    .

     AND <condition-n>

     THEN <consequent-A>
```

<consequent-A> would only be true if all the conditions (1 to n) were true.

An example of an AND rule from the new EXILE knowledgebase follows.

```
IF prepared by public official

AND prepared in duty

AND to enable public to inspect

AND permanent record

THEN public document
```

This rule states that 'public document' is true only if all of 'prepared by public official', 'prepared in duty', 'to enable public to inspect' and 'permanent record' are true.

Any rule antecedent may also exist in a negated form.

2.3.2 Improvements to the 'How' Explanation System

After a conclusion is reached (ie: the goal is solved), it is desirable for the user to be able to request an explanation of how the solution was reached. This allows the user to check the reasoning process used by the inference engine (and to correct the knowledgebase if necessary).

The method used in this inference engine is to display one level of conclusions at a time. That is, information relating to the nodes that are children (first level children, not descendants) of the node being inspected will be shown, and further refinement will be under the control of the user.

The advantages of this method are
- The display does not become unnecessarily cluttered by pages of information;
- The user does not have to view undesired information.

If the user asks for an explanation of how the solution was reached, the Expert System Shell will display the relevant information about that node (initially the goal) and the node's children.

If the current node is TRUE, and it is an AND rule, then all the conditions for that node are listed; if it is an OR rule, the first TRUE condition for that node is listed.

If the current node is FALSE, and it is an AND rule, the first FALSE condition is listed; if it is an OR rule, all the conditions for that node are listed.

If the rule for that node is either an IF-THEN or a NOT rule, the only child is listed.

If further expansion on a listed child is possible (the child being derived from other rules), a letter is displayed next to that child explanation. This letter is used for examining that child further. If further expansion on a listed child is not possible (the child is a leaf (askable) node, and the value for that child was supplied by the user), then the words (was told to me) are listed next to that child.

The user then has the option of expanding a child (in which case the above process is repeated with the child as the new current node), going up a level (examining the parent of the node now under examination) or quitting. Sample dialogue from an EXILE session follows.

```
The statement is admissible under the Evidence Act 1977-89 (Qld).

Do you want to see how I arrived at that conclusion? y

the statement is admissible under the Evidence Act 1977-89 (Qid), because

        A) s.92(1)(a) and the exemptions in s.92(2) apply


If you want to expand on any of the above, enter the letter, or press U to go up

or Q to quit: A

s.92(1)(a) and the exemptions in s.92(2) apply, because

        A) s.92 of the Evidence Act is relevant

AND B) the statement in question is contained in the form of a document

AND     the subject matter of the statement tends to establish a fact (was told t

AND     the statement is relevant to a fact in issue (was told to me)

AND     direct oral evidence of that fact is admissible (was told to me)

AND     the maker had personal knowledge of the matter(s) dealt with in the state
      or document (was told to me)

AND C) the maker of the statement need not be called because the exemptions in s.

If you want to expand on any of the above, enter the letter, or press U to go up
   or Q to quit:
```

2.3.3 Sample Displays of Parts of the new EXILE knowledgebase.

Below are sample displays of the screen for various parts of the EXILE knowledgebase.

```
DISJOINT TREES

    statement is admissible

    computer statement is admissible

    entry is admissible

    statement produced by computer

    subs2 conditions satisfied

    common law options

    statement admissible cth
```

This display shows the list of disjoint trees.

```
prove1 is ok IF

            dealing with s.92

AND statement is in document

AND statement tends to establish fact

AND statement is relevant to fact in issue

AND direct oral evidence is admissible

AND maker has personal knowledge

AND maker will be called

    ------prove1 is ok ------

    TRUE :s.92(1)(a) applies

    FALSE:s.92(1)(a) does not apply
```

This display shows an AND rule, with consequent 'prove1 is ok' and several conditions for that rule. Notice the explanations associated with the consequent provel is ok at the bottom of the screen.

```
maker will be called IF
```

This is an askable question (LEAF node)

```
   ------ maker will be called ------

    TRUE :the maker of the statement will be called as a witness

    FALSE:the maker of the statement will not be called as a witness

    QUES: Will the maker of the statement be called as a witness?
```

This display shows an askable node. There is no rule with 'maker will be called' as a consequent, and as such, the user must supply the answer. Notice that the display of explanations at the bottom of the screen includes a question.

In the displays above, notice that when the current node displayed on the screen has one or more children, one of the children is highlighted.

The highlight pointer may be moved from child to child by means of the up and down arrow keys. Pressing the Enter key will make the highlighted child the new current node, and display the child's information on the screen.

Pressing U will move upwards in the tree (ie: to the rule that has the current node as one of its conditions). A leaf node will have no children (obviously) and when a leaf node is the current node, the only way to walk is Up, and the Enter key has no effect.

3.0 The ESCAPE Application

3.1 Aims of the project

ESCAPE was designed to demonstrate the conversion of aspects of the Cheques and Payments Orders Act 1986 (Cth) to an expert system. The Act operates like a code to the extent that it covers this area of the law, and thus lends itself to interpretation as an expert system. Cheques and bills of exchange as a whole comprise a difficult area of commercial law, and one that practitioners who do not always practice in the area would have difficulty with. Having access to ESCAPE would allow them to consult an expert system with their problems of recovery under the Act. Although some familiarity with the law is assumed, ESCAPE is intended to have application beyond lawyers to such entities as accountancy firms and banks.

3.2 The Cheques and payments Orders Act 1986 (Cth)

Not all aspects of the law relating to cheques are dealt with under the Cheques and Payments Orders Act. Thus ESCAPE is limited to advising on recovery under the Act, not under the general law.

3.3 Design of the Knowledge Base

The structure of the Act is long and difficult, meaning that an expert system which attempted to cover it entirely would be large and complex, and probably beyond the range of one person to produce. It was therefore decided to concentrate on those sections of the Act dealing with the ability of a holder to recover on a cheque.

The sections dealt with in the system run through an action where the holder of a cheque seeks to recover against one of the parties to the cheque, but is unsure against whom this can be done. The system firstly establishes whether the person is in fact a holder of the cheque, and therefore whether they have the prima facie standing to sue on the cheque. Once this has been established, the next disjoint tree and chain of enquiry is whether the party against whom the holder is attempting to recover has any defences to the action. Five possible defences under the Act are explored. The final disjoint tree looks at whether the holder car overcome any of the prima facie defences by virtue of the type of holder which he or she is. For example. if the holder is not just a mere holder, but is a holder for value, or a holder in due course, he or she car overcome the defence of lack of consideration. Thus the system gives initial advice on the possible actions available to a holder of the cheque.

The sections of the Act generally lent themselves readily to the construction of rules. Each section dealt with under the system is broken down into component parts, each of which became the antecedent of a rule. Where further explanation was necessary, it was generally dealt with by way of a consequent addition to the rule base. Sometimes this was not necessary, and entries in the explanation file were sufficient. Using an addition became essential when the antecedent needed to be broken down into further sub-rules, which was often the case.

Because the Act operates as a Code, most sections caused no problem with a logical breakdown into rules. The system was engineered so that each set of sections was dealt with together and flowed from one to the next.

It was not possible however to deal with the three main lines of enquiry, each a root node of disjoint trees, other than by making them separate menu items. This was so because they were distinct from each other, dealing with separate issues, and could not have been dealt with one after the other in the same line of enquiry. The disadvantage of this approach is that a user of the system must go to the main menu three times if the system has determined that the potential plaintiff is a holder, and the potential defendant has one or more prima facie defences available. The advantage is that specific

lines of enquiry can be pursued without addressing material which the user already knows.

3.4 Construction of me Disjoint Trees

There are three root nodes of disjoint trees in ESCAPE:

```
TREE

holder can recover

defence available

holder - type may avoid defence

END TREE
```

These are explained individually below.

3.4.1 Holder Can Recover

The first menu selection advises whether the person is a holder of the cheque, and if so whether they can prima fade sue on the cheque, and recover under the Act. This is called "holder can recover" in the tree.

It is represented by the question "Can the holder recover on the cheque?" in the main menu. There are a number of sub-goals which must be proved in order for the expert system to conclude that

the holder can prima facie recover on the cheque under the Act:

```
IF instrument is a cheque

AND person is a holder

AND holder may sue in own name

AND recovery OK

THEN holder can recover
```

Each of these is broken down into further rules.

3.4.2 Defence Available

The second menu option builds on the first and gives advice as to whether any defences are available under the Act to the prima facie action by the holder of the cheque. It is represented in the main menu as "Are there any prima facie defences available?". The system then moves to the defences menu which lists the following:

1. Lack of consideration
2. Lack of due presentment
3. Cheque discharged
4. Cheque discharged through fraud
5. Forgery

The defences available are expressed in the rule file as follows:

```
IF lack of consideration

OR lack of due presentment

OR cheque discharged

OR cheque discharged - fraud

OR forgery

THEN defence available
```

Again, as with the first top-level goal, this rule is further broken down into component parts.

3.4.3 Holder - Type May Avoid Defence

The third top-level goal establishes which category of holder the potential plaintiff falls into, and therefore whether any of the defences can be overcome. The sub-goals of this disjoint tree are as follows:

```
IF holder

AND NOT holder for value

AND NOT holder in due course

AND NOT taking from holder in due course

THEN holder - type can avoid defence
```

4.0 Conclusion

MASTER is a versatile and useful tool for creation and maintenance of expert systems in the legal domain. It has been developed to operate within the computing constraints of even the smallest law office. Expert systems have potential for widespread application amongst legal practitioners, provided the latter can have confidence in the expert knowledge of the system builders.

University law schools are obvious sources of legal expertise. University computer science schools are obvious sources of computing expertise.

The MASTER project and the continuing collaboration between the authors and their respective faculties at the QUT in Brisbane provide an illustration of what can be achieved.

Law has been the last of the professions to embrace computer technology, and particularly expertsystems applications. Experience in other domains suggests there may be a substantial market opportunity for productive technology transfer.

References

1. Cattell,P.T.J., Machtynger,J.Y., Wilson,I.A,"EXILE: Expert Information in Legal Evidence" University of Reading, Department of Computer Science Research Paper, 1988

2. Cattell,P.T.J., Machtynger,J.Y., Wilson,I.A. "EXILE -Expert Information in Legal Evidence", Conference proceedings, 5th Bileta (British and Irish Legal Education Technology Association) Conference, University of Warwick, Coventry, UK, 11-12 April, 1990,

3. supra p.23

4. Walker,A., An Inference Engine Which Explains Both Yes and No Answers , in proceedings of the 8th IJCAI, Los Altos, CA: William Hoffman, Inc., 1983, pp 526-528

Authors
Peter Cattell is a senior lecturer in the department of computing science,
Queensland University of Technology,
Brisbane Australia.
Ian Wilson is a senior lecturer in the faculty of law at QUT.

The authors acknowledge the contributions to this paper made by:
Peter Chadwick
Mark Looi and
Geraldine Mackenzie
all of whom are postgraduate students at QUT.