Artificial Intelligence Review **6**, 129—143, 1992.

The maintenance of legal knowledge based systems

TREVOR BENCH-CAPON AND FRANS COENEN

Department of Computer Science, University of Liverpool, Liverpool, U.K.

Abstract. Legal knowledge based systems (KBSs) are, by definition, grounded on law. Very often the relevant law is subject to routine amendment and repeal, such changes occurring at irregular and unpredictable intervals. These systems are thus particularly affected by significant problems of adaptation as a result, a fact which has limited their practical take-up. If they are to be of more practical use the maintenance issues associated with these systems must be taken seriously. In this paper we discuss the issues associated with the maintenance of legal KBSs and describe a suite of maintenance tools designed to address these issues.

Key Words: knowledge based system maintenance, legal knowledge, adaptation.

1. INTRODUCTION

Researchers in knowledge based systems and law are often asked why, given the success claimed for their research prototypes, there is not a greater take up of such systems in practice. This is an important question, and one which merits an answer. The question can, of course, be directed also at knowledge based systems in other domains, but we believe that there are considerations peculiar to the legal domain which mean that the question can be usefully addressed with reference to that domain. The answer is not that there is no potential demand for legal knowledge based systems: nor is that demand confined to lawyers lacking the financial clout to support the development of such systems. The most significant impact is likely not to be on lawyers at all, but on those whose jobs are governed by law (or law-like regulations). As evidence for this claim we may consider the following quotation written by Paul Duffin, a prominent member of the UK Central Communications and Telecommunications Agency (CCTA):

> The UK Civil Service is the largest single user of conventional IT equipment and services in the UK . . . The CCTA has a specific responsibility to research and then encourage the use of appropriate IT to assist in the administrative mechanisms of Government. KBS represents one such technology which CCTA has identified as being of particular benefit . . . In terms of government administration, KBS may be the single most significant development to emerge since the computer itself, for it offers a means of streamlining and improving decision-making to an unprecedented degree. (Duffin 1988)

The activities that he saw being particularly influenced by such systems go to the heart of administration

> Much of government 'mainline business' involves the administration of regulations or the following of set procedures or, frequently, both. These areas of application are amenable to computerised assistance using ES [Expert Systems] techniques, as has been demonstrated. (Duffin 1988)

If we widen our notion of a 'legal' KBS to include not only the 'laws of the land' but also the internal procedures and guidelines used by companies to direct the activities of their employees, Duffin's remark about the Civil Service of the UK becomes relevant to any large organisation which performs a good deal of administration — that is, any large organisation whatsoever. All administrators must make decisions within the policies and guidelines of their employers, and this activity is an informal analogue of legal decision making, and susceptible to the application of similar KBS techniques. As an example of this law-like activity, banks have policies on lending, and issue guidelines to their staff to realise these policies. A system which supported a loan scheme would be able to employ much the same techniques as a truly legal system, such as a system to support the adjudication of claims to welfare benefits.

Thus legal KBS are wanted as practical systems, and it is this very potential for use that has attracted many researchers to the area. So is the problem that they are infeasible? Much research has been devoted to showing that this is not the problem either: the British Nationality Act project (Sergot *et al.* 1986) has shown how legislation can be represented in an executable form, and further related work, such as (Bench-Capon 1988) has explored the relation of such a formalism to a practically useful system. Many other examples of successful, in the sense of feasible, systems exist. And indeed there are practical examples of such systems in use, perhaps most notably the Retirement Pension Forecast and Advice System (RPFA) (Spirgel-Sinclair 1988) and the VATIA system (Susskind 1988). Of course, there remains a gap between the demonstration of a feasible research prototype and a demonstration that such systems could be of real practical utility in an operational situation. A demonstration of the latter can, however, be achieved only by a rather greater number of live systems coming into use.

Thus it can be said both that there is a great potential demand for such systems, and that it has been shown that it is possible, and, in a few cases, profitable to build such systems. But our original question remains: why are such techniques not part of the routine armoury of large organisations? Why is it that those with the power to commission such systems do not have sufficient confidence in the viability of the KBS solution?

Part of the answer lies in organisational issues. The traditional consultative model of an expert system is simply not appropriate to support many of the tasks which need to be addressed. The RPFA, mentioned above, does not follow this model, and the need to take the task seriously and to tailor the support provided to the particular task is well documented in (Bench-Capon 1991b) where the same legislation is shown to give rise to very different systems when these systems are directed to different tasks founded on that legislation.

These issues, however, mean that it is that much more time consuming and

difficult to build such a system, not that it is impossible. It is our belief that the greatest barrier to the routine use of KBS techniques for practical legal applications lies not so much in the problems of building the systems, since this process is becoming better understood and methodologies for knowledge engineering are becoming established, as in the problems associated with the maintenance of such systems. Building a KBS requires a substantial investment, and such an investment will only be forthcoming if the expected benefits outweigh the envisaged costs. The calculation depends critically on the expected life of the system. Now while, to take a simple example, a system to diagnose faults in a machine can be expected to last until the machine becomes obsolete without much need for change (except to correct errors in the program), this is not at all the case with a system in the domain of law. One thing that is certain about law (except in some freak domains such as Nervous Shock (Smith 1987)) is that it will change over time. This is especially true of those kind of regulation-orientated domains where we expect KBS techniques to be most useful. This means that the life expectancy of the system is wholly unpredictable unless there is some clear strategy to enable the system to cope with these changes. It would be the purest folly to invest in a legal KBS unless there was some assurance that it would be maintainable, and this is an issue which has received far too little attention. For a clear discussion of these issues, a description of the inadequate way in which they are addressed by current approaches, see (Bratley *et al.* 1991). So our answer to the original question about practical take up is that there can be no confidence in the applicability of KBS until a convincing answer to the maintainability of such systems can be given. It is our intention in this paper to make an attempt to provide the beginnings of such an answer.

2. CHANGES IN REGULATION BASED KBS

The way in which the knowledge relevant to a Regulation Based KBS changes is different from many of the areas to which KBS techniques are applied. In the area of fault diagnosis, for example, where the subject of the domain is some machine designed and constructed by humans and the malfunctions that may occur, and the remedies that may cure them, the knowledge that a system must use is relatively stable. The system may require debugging, but this is often a matter of extending the knowledge that needs to be included without invalidating what already exists. While the machine remains in use, the core of the system needs little attention. This happy state of affairs does not exist in a regulation based domain. For regulations are repealed and amended as well as added to, and a decision in a landmark case may necessitate revision of existing interpretations of the law. This is a significant difference, posing significant problems. The situation is analogous to the well known problem of truth maintenance in KBS: so long as information is simply increased there is no problem, but when an additional piece of information requires existing beliefs to be revised the matter is no longer simple, as the variety of truth maintenance systems and non-monotonic logics found in the AI literature demonstrates.

Thus while the incremental refinement of the knowledge base as a solution to maintenance, often cited as a strength of classic expert systems, whereby rules are 'simply' added to the knowledge base, may be a feasible strategy for some domains; such a strategy is certainly inappropriate to regulation based KBS, where the existing knowledge may become useless overnight. In practice this incremental approach is not without its problems in other domains also, as the ever growing team maintaining XCON will testify.

Problems arising out of the changes in regulations are well known in conventional data processing: changes in tax law, for example, must be announced well in advance of coming into effect so as to allow time for the considerable task of altering the programs which have to apply these laws in payroll and other applications. However the problems associated with KBSs are greater than with a conventional system. In the conventional system the limited range of tasks which such a system can perform tends to restrict the knowledge represented. Thus a payroll system will need to have recorded within it such things as the rates at which tax is paid and the thresholds at which these rates come into effect, but it will not record the sort of expertise and knowledge of precedent that we would expect from a tax lawyer. The kinds of thing which are recorded tend to change at regular intervals, are signaled well in advance, and change in relatively predictable ways. The regulation based KBS, in contrast, will typically be expected to incorporate some elements of expertise as well, and this will change in an irregular and unpredictable manner as decisions are made, or as external circumstances change. This means both that the detecting of such changes, and the decision of the appropriate response and incorporating them into the knowledge base may be problematic.

3. MAINTENANCE ASSISTANCE FOR KNOWLEDGE ENGINEERS (MAKE)

The MAKE project, a collaboration between the University of Liverpool, ICL and British Coal, is investigating the issues connected with maintenance of regulation based KBS. The specimen application being considered concerns claims for compensation for work related injuries made by employees of British Coal (BC). Such an application is fairly representative of the sorts of application claimed in the introduction to offer the greatest potential for the exploitation of KBS.

The BC application bears out the expectation that there is a considerable need for maintenance. Regarding the law itself, each year, there are between 10 and 20 court judgements in British Coal cases and another 5 relating to other employers, but with significance for British Coal. There are up to 20 new relevant Statutory Instruments, and 10 technical instructions issued. In addition the policy of British Coal is modified from time to time, and some 10—15 such policy decisions are made in a typical year. All of these alterations need to be assimilated by the clerks dealing with the claims. Other changes in the expertise of these employees arise out of changes in medical views, for example the acceptance that a particular substance can cause dermatitis; policy changes by

other bodies, as when a particular firm of solicitors may start to issue writs if the claim is not settled in a certain period of time; and changes in the perception of methods of work or occupations. British Coal estimate that these will require another 30 changes per year. Some of these changes will be relatively minor, but none the less the cumulative effect of these changes indicates the rapidity with which a knowledge base dealing with this sort of application would go out of date. If the advantages cited for using KBS to support such tasks are to be realised, it is essential that the knowledge be kept up to date, and so a practical system would require continuous updating.

The principal aims of the MAKE project are thus to produce a KBS development environment, MADE (Make Authoring and Development Environment), that encourages the production of maintainable regulation based KBSs and a set of tools to support the maintenance of such systems. The BC application is intended simply act as a test bed for the methodology and the tools. In this paper the maintenance tools developed, or under development, as part of the MAKE project are described. For further details of the MADE development environment and methodology interested readers are referred to (Coenen 1991).

4. ISOMORPHISM AND MAINTENANCE

We take as a starting point that maintenance is something which needs to be taken seriously throughout the development of the KBS: it is not an issue that needs attention only when the system is complete and the maintenance phase commenced. For if a system is to be maintainable, this will not occur by chance, but needs to be ensured by the way the system is built. In particular the way the knowledge is represented is critical.

One factor that makes the maintenance of regulation based KBS difficult is that when the Knowledge Engineer encodes the knowledge that he has elicited from the expert he will often bring together separately presented items in a single rule. We can illustrate the effects of this with the following simple example concerning the increasingly obsolescent Category C Retirement Pension.

The UK Social Security Act states:—

> 39(1) Subject to the provisions of this Act –
> (a) a person who was over pensionable age on 5th July 1948 and satisfies such other conditions as may be prescribed shall be entitled to a Category C retirement pension at the appropriate weekly rate.

To interpret this we need also to bear in mind

> 27(1) In this Act 'pensionable age' means –
> (a) in the case of a man, the age of 65 years; and
> (b) in the case of a woman, the age of 60 years.

Now if we consider the kind of knowledge that an expert adjudicator might apply to decide claims for this benefit, we might see him allowing claims of men

aged 108 and women aged over 103. This would certainly pick out the correct group of people and would be the most convenient expression of the knowledge if the claim form gave the age of claimant. It does, however, 'compile in' both a certain amount of arithmetical expertise and knowledge of the current date, as well as the interaction between 27(1) and 39(1). Moreover, such a rule would need to be amended every 5 July. If the claim form contained not the age of the claimant but the date of birth, however, this would not be the most convenient expression of the knowledge, since a calculation would now be required to get the age from the date of birth, and the expert would be likely instead to operationalise the knowledge as 'men born before 5/7/1883 and women born before 5/7/1888'. This still conflates 27(1) and 39(1). An expert will inevitably operationalise his knowledge in a way that is suited to the task he is required to perform, but this operationalisation may well remove distinctions which are important when we come to maintain the system.

Suppose for example 27(1) was amended, perhaps to equalise pensionable ages. The impact of this on the interpretation of 39(1) could not be recognised in a conflated representation. This suggests that the representation used for a regulation based KBS should avoid conflating disparate items of knowledge into single structures. This can be achieved if we mirror the structure of the knowledge sources in our representation so as to attain a degree of isomorphism between the representation and what is represented. A fuller discussion of this need is to be found in (Bench-Capon 1991c) and (Routen 1991). This can often be achieved by a disciplined use of a representation rather than use of a distinctive representation, although certain extensions are required to Prolog (or any first order formalism) if this is to be possible with regard to legislation: again this is fully discussed in (Bench-Capon 1991a) and (Routen 1991). Further, we can note that achieving a structural correspondence here will also enable us to record the provenance of all the items of knowledge in our intermediate representation, which is not a simple matter in the absence of such isomorphism, but which is vital if changes are to be followed through from source to knowledge base.

Thus one thing that must be done to ease the problems of maintenance is to use a representation that enables the knowledge base to maintain a close structural correspondence with the original source documents, so that it is possible to identify the parts of the knowledge base which are jeopardised by a given change. Moreover, for this to have its best effect, statements in the representation must be truly declarative. While almost all knowledge representation paradigms have declarativeness as an aspiration, in practice the use of, for example, conflict resolution strategies in production rule systems, means that it is not possible to detach a piece of a knowledge base from its context and consider its correctness in isolation. If we want to ensure that localised changes to the source material result in correspondingly localised changes to the knowledge base, we must be sure that there are no ramifications of changes resulting from a subtle alteration of the meaning of the statement deriving from its context in the knowledge base.

We therefore conclude that the form of representation used is a crucial factor

in the production of maintainable systems. The tools developed on the MAKE project are consequently targeted upon a form of representation which has the properties described above. This formalism is the representation and inference Toolkit developed on the Alvey-DHSS Demonstrator project, particularly for the representation of legislation (Bench-Capon 1991a). In brief, these facilities comprise an inheritance hierarchy, with the classes viewed as logical types, their slots as attributes of these types, and the possibles values of these slots specified in the class description. Inheritance is by strict specialisation. This hierarchy represents a vocabulary in which constraints expressing the relations between slots can be expressed. These constraints are expressed in a typed logic extended to include arithmetic.

It is worth briefly noting the objection to this approach to the maintenance problem given in (Bratley 1991), namely that for a system to be useful the representation must be augmented with the expertise to provide for the interpretation of the law and the resolution of vague concepts. They claim that such an augmentation will necessarily destroy the correspondences which it was argued above would facilitate (or make possible) maintenance. We disagree: in the kinds of domains we are interested in this expertise is also available in written form, as guidance to the adjudicating clerks. All that we require is that this guidance is also represented in an isomorphic manner, and that a clear separation between the various knowledge sources is observed.

5. REGULATION BASED KBS MAINTENANCE

It is not the intention of the MAKE project to address major maintenance tasks which may necessitate the entire rebuilding of the system. Of course, if the law changes root and branch, there is little that can be done to accommodate this. The aim is to address minor adaptive maintenance only, i.e. maintenance resulting from the day to day changes in the source material due to changes in regulations and legal texts, the application and operation of the law etc. In this context the maintenance required can be considered under a number of headings: (a) Rule Base (RB) maintenance, (b) Class Hierarchy (CH) maintenance, (c) changes to the source data, and (d) validation.

In the following subsections each of these headings is discussed in further detail. In each case the nature of the associated maintenance is described and appropriate tools to assist in the maintenance task identified. In the following section the tools are described in further detail. It should be noted that this catalogue of tools expresses some possibilities and areas for work. Within the MAKE project not all of the tools described have been fully developed and implemented. Some are currently in operation, others are in the process of development and some exist only as a rough specification.

5.1. *RB maintenance*

The maintenance of the RB will involve one or more of the following activities:—

M1 The introduction of a new Rule.
M2 The modification of an existing Rule.
M3 The removal of an existing Rule.

The effect of introducing a new Rule may be unwanted redundancy or subsumption (so that the RB contains a rule which has no effect), or the creation of a missing branch (so that there is no linkage from an intermediate conclusion to an ultimate goal), a hard contradiction or soft inconsistency. A hard contradiction is simply a that a logical contradiction, i.e.:—

$$A \ \& \ \text{not} \ A$$

is derivable from the KB. What we term a 'soft inconsistency' occurs when some proposition is a consequence of the KB where as it is in fact known that its negation is possible. In the simplest possible case we may have two Rules:—

$$P \Rightarrow Q$$
$$P \Rightarrow \text{not} \ Q$$

There is no logical contradiction here, but not P is a logical consequence of the KB. If, however, P represented something which we knew to be sometimes true and sometimes false, this would indicate that our KB was in error.

The introduction of a new Rule will thus involve checking for the following:—

C1: Redundancy or subsumption.
C2 Missing Rules or branches.
C3 Hard contradiction.
C4 Soft inconsistency.

Considerable work has been done on the development of suitable ways addressing these structural defects of a knowledge base. This work has usually been directed towards some constrained representation. In the MAKE project we have produced algorithms to address C1 and C2 for the formalism we use, although some problems remain. Our algorithms cannot currently ensure that all problems of this sort are detected: our view is, however, that the detection of some defects is a help to the maintainer of the system. Algorithms for C3 and C4 exist in other systems such as (Rousset 1988) although these do not address all aspects of inconsistency. Further algorithms to address contradiction or inconsistency have been proposed as part of the MAKE project, but none of these can be shown to be complete. Detecting all such inconsistencies may well be computationally intractable, but defects that are detected can be brought to the attention of the maintainer and so fixed.

The removal of a Rule may also result in the creation of a missing branch or cause a section of the KB to become redundant. Therefore when removing a Rule checks C1 and C2 should be implemented.

The modification of a Rule has the same effect as removing a Rule and

introducing another. Hence the methods outlined above for the introduction and removal of Rules can be used in sequence:—
— Remove old Rule
— check for C1 and C2
— Introduce new Rule
— check for C1, C2, C3 and C4.
It should be noted that as a result of removing the Rule in this case some acceptable redundancy and/or missing branches may temporarily be created until the new Rule is added.

It would be rare for a maintenance session to consist of only the removal or introduction of a single Rule. In most cases a maintenance session will involve all three types of KB maintenance, i.e. M1, M2 and M3. It is therefore proposed that on completion of any KB maintenance checks for C1 to C4 should always be carried out. Two RB maintenance tools may therefore be identified:—
T1 The Rulemap.
T2 Hard Contradiction and Soft Inconsistency Identification.
T1 can be implemented on the RB in its static form and incorporate checks for redundancy and subsumption. It can also be considered to be an RB navigation tool that will allow the user to move through the RB at the intermediate representation level and the fine grain, executable, level so that missing Rules and branches can be identified and facilitate verification 'by eye'.

T2 is designed to address the dynamic aspects of the RB and will be implemented at both the intermediate and executable representations as appropriate. The existence of redundant and missing branches will only be significant in a task dependent RB.

5.2. *CH maintenance*

In the representation used on the MAKE project (see Bench-Capon 1991a, for a fuller description) the CH plays an important role as the means by which the vocabulary to be used in writing the Rules is defined, and as the means of recording the state of an application at any particular time. The discipline that this imposes is important if the representation is to be a faithful reflection of the domain, and hence keep its structure through a period of maintenance, whilst remaining an adequate vocabulary for modelling the domain. The maintenance associated with the CH may involve:—
M4 The modification of an existing Slot by introducing a new Value.
M5 The modification of an existing Slot by removing an existing Value.
M6 The modification of an existing Class by introducing a new Slot.
M7 The modification of an existing Class by removing an existing Slot.
M8 The introduction of an entire new Class.
M9 The removal of an existing Class.

5.2.1. *Introducing or removing a value*

One of the most basic actions in the maintenance of the CH is the addition of a possible Value to a Slot. In practice a Value will be added to a Slot as a con-

sequence of the introduction or modification of a Rule which necessitates an extension to the vocabulary. The allocation of this addition Value to an existing Slot will not generally effect the operation of any established Rules or the existing CH. The exception to this is if Rules exist that use the possible Values of a Slot to express negation. Thus the Rules that contain the Attribute to which a Value is to be added need to be identified so that the effect of introducing this Value can be determined. For this purpose it will, in some cases, be necessary to go down to the fine grain level of representation.

Removing a Value from a Slot will jeopardise all Rules which make use of that Value either in the Head or the Body of the Rule. These Rules must therefore be identified and presented to the maintenance engineer so that a decision can be made on whether it is appropriate to remove the Rule, remove the atom containing the removed Value, or modify the Rule.

A tool to allow the identification of jeopardised Rules as a result of removing and introducing Values to and from Slots in the CH is therefore desirable. However a more general tool to identify jeopardised Slots and Rules as a result of changes to the Source data or changes to the Rule Base or CH would be more beneficial. Thus:—

T3 Jeopardy Tool.

Because of the inheritance mechanism used, the CH insists on strict specialisation, so that a Value can only be added to an Attribute at the highest level at which that Attribute appears. If the Rule which motivates the introduction of the new Value refers to a Class which inherits the Attribute from a Super-Class, either the Value must be added to the Super-Class, or some new Attribute must be created in the Class in question. If the Value is added to the Super-Class, of course, the Rules for that Class in which the Attribute appears are jeopardised. Therefore, before a new Value can be added, the user should be confronted with the Class which introduces the Attribute to the CH, which may not be the Class mentioned in the Rule which motivated the introduction of the Value, and the Value added to this Class. If the Class to which the Attribute is added is not a leaf Class, this process would be facilitated by a tool which walks down the CH so that the user is able to determine the correct point, on each path, at which the new Value should cease to apply.

There is thus a need for a tool to provide the maintenance engineer with the facility to walk systematically up or down the CH, focusing on particular Attributes. Thus:—

T4 CH Navigation Tool.

5.2.2. *Introducing or removing an attribute*

An Attribute can be added to a Class in two ways. Either it can be added directly to the Class, or it can be added to a Super-Class, and so added to the Class in question indirectly by inheritance. Thus if a Rule needs to mention a new Attribute for some Class, the first step should be to walk up the CH to determine the appropriate point at which the Attribute should be introduced into the hierarchy. Note, however, that adding it to a Super-Class will cause all the Sub-Classes of that Class to take on the Attribute, not only those on the

path walked up. Once introduced into the hierarchy Rules may be written using the Attribute. These can then be subject to the usual checks for new Rules already described. The final stage will then be to walk down the CH specialising the Values of the Attribute as appropriate, until a point range is reached on every downwards path. The process of adding a new Attribute to an existing Class will thus also involve the use of the CH Navigation Tool (T4).

The process of removing an Attribute from an existing Class can be regarded as removing a set of Values. Inheritance, however, means that the Attribute will also be removed from all the Sub-classes up to the point in the hierarchy at which it was introduced. The best approach would therefore be to commence the removal at this point, and then to walk down the CH to determine at which point the Attribute should be reintroduced into the hierarchy if necessary. Class(es) at which it is now introduced and their Sub-Classes will not be affected. Thus the CH Navigation Tool will also be of relevance here.

5.2.3. *Introducing or removing a class*

The point at the hierarchy in which the Class should be introduced will be best determined by the Attributes that need to be associated with the Class. If an existing Class contains a subset of the desired Attributes then it is a potential Super-Class for the new Class. Clearly the Class with the largest such subset (i.e. the lowest such Class in the hierarchy) is the logical Super-Class to choose. Next it must be determined which existing Classes should be Sub-Classes of the new Class. The answer here is that existing Classes with Attributes which are a superset of the new Class should be Sub-Classes of the new Class. A tool to determine the relations between sets of Attributes is clearly suggested. This may be incorporated into the CH Navigation Tool. If a suitable super Class is not identified, the Class can be considered to represent a leaf node, a Sub-Class of the Class with the largest sub-set of desired Attributes.

Removing a Class is, as far as the KB is concerned, effectively like removing a set of Attributes. As far as the Class Hierarchy is concerned, existing Sub-Classes of the Class need to become Sub-Classes of its immediate Super-Class. Problems still arise if any specialisation of Attribute Values, or addition or Attributes, were made in the removed Class. Clearly such Attributes must be re-introduced, or specialisations made, either in the Sub-Classes or the immediate Super-Class, as seems to be most appropriate.

5.3. *Changes to the source data*

At a higher level, Rules will also be jeopardised by changes in the source material, as when legislation is amended. A feature of the MADE development methodology is that a linking facility is provided to link individual sections in the source material through the various analysis stages to the resulting CHs and RBs in the target representation. This provides a useful basis for identifying Rules and Classes that may be affected as a result of changes in the source material and can be automated as part of the Jeopardy Tool (T3).

5.4. *Validation*

So far only the verification of the KB and CH have been considered. However it is also necessary to validate the KBS after maintenance has taken place. This can be carried out by peopling the Rule Base and determining what conclusions can be arrived at or tracing how inferences are made. Thus:—
T5 Rule Base Animation Tool.

6. THE MAKE SUITE OF MAINTENANCE TOOLS

In the previous Section a number of maintenance tools were identified to address different aspects of KBS maintenance. These are summarised below:—
T1 The Rulemap.
T2 Hard Contradiction and Soft Inconsistency Identification.
T3 Jeopardy Tool.
T4 CH Navigation Tool.
T5 Rules Base Animation Tool.
In the following Sub-Sections each of these tools will be described in greater detail. An indication will also be given expressing the state of development which each tool has reached.

6.1. *The rulemap*

The Rulemap, although still under going modification, has been in operation for some time now. It consists of a directed (from left to right) bipartite graph which graphically displays the Rule Base either at the Attributes-Rules (intermediate representation) level or the Proposition-Clause (executable representation) level. A number of options are provided to allow the user to walk up and down the Rule Base. By following a path through the Rulemap it is possible to determine the Leaf Attributes and Propositions into which a Root Attribute ultimately unfolds and vice versa. This gives the user a clear visual view of the rules in the knowledge base, from the various perspectives of source, intermediate representation and executable representation. Utilities are also provided to allow the user to interrogate the Rulemap to display the Rule and Clauses in which Attribute and Propositions appear or to inspect the Values or Entities associated with Attributes and Propositions. It is intended that the facilities to identify redundant or subsumed Rules or sub-sets of Rules will be accessed from this tool.

6.2. *Hard contradiction and soft inconsistency identification*

When adding or modifying rules in a KB during a maintenance session a hard contradiction may be introduced. In logical terms this means that there can be no model for the knowledge base, so the knowledge base cannot be correct. Soft inconsistency is a modified phenomenon and occurs when some Proposi-

tion is a consequence of the KB when it is in fact known that its negation is possible. This means that the knowledge base excludes some models which are known to occur, and again suggests a defect in the knowledge base. Thus a minimal validation of the knowledge base will involve ensuring that neither of these situations exist. Tools are under development in the MAKE project which will allow such inconsistencies in a knowledge base to be detected. The algorithms used will only serve to detect a proportion of the contradictions and inconsistencies contained in a Rule Base. However it is claimed that some detection is better than none at all.

6.3. *Jeopardy tool*

This is a general purpose maintenance tool to identify jeopardised Slots and Rules as a result of changes to the source data or changes to the Rule Base or Class Hierarchy. The tool will incorporate the following facilities:—

(a) Rules Jeopardised by Slot Changes Identification.
Facility to identify Rules jeopardised as a result of changes to Slots in the Class Hierarchy.

(b) Class Definitions Jeopardised by Class Deletions Identification.
Facility to identify the Slots Jeopardised by the removal of a Class because they are typed to that Class.

(c) Rules Jeopardised by Source Changes Identification.
Facility to identify the Rules that are effected by changes in the source material.

(d) Slots Jeopardised by Source Changes Identification.
Facility to identify the Slots in the Class Hierarchy that are effected by changes in the source material.

(e) Rules Jeopardised by Rule Changes Identification.
In a task dependent Rule Base Rules above and below an altered Rule may be jeopardised. This facility will identify the sub-set of Rules which have been affected by a KB maintenance session.

The jeopardy tool operates using the links that should be included by the knowledge Engineer during system development. This is an essential part of the MADE methodology. A change in the source can then be linked through to the Rule Base and Class Hierarchy. Some automation has been introduced here resulting in 'warning triangles' being placed at the heads of Rules when changes to the sources are made. Work is still in progress on this tool.

6.4. *The class hierarchy navigation tool*

This is a Class-Instance Browser designed to allow the user to navigate through a Class Hierarchy. The tool is intended to give visibility to the author of not just the Class Hierarchy, but also where Slot definitions come from, and will enable the user to determine the best location for new Classes, Sub-Classes and Attributes related to those Classes, and specialisations of attribute values. A

version of this tool has also been in operation for some time although some of the facilities have yet to be added.

6.5 *The rule base animation tool*

This tool is similar to the Rulemap but addresses the dynamic aspects of the Rule Base. It allows the user to people the Rule Base by creating Instances and asserting Propositions and then to determine what inferences can be made as a result. When the behaviour is unexpected, either because an inference which should not be made is made, or because an inference which was expected fails to be made, this tool will enable the user to locate the precise clause which caused the failure, and from this the rule, analysis and source from which it was derived. Such animation is a necessary adjunct to the 'by eye' validation supported by the static tools, since the practical consequences of a given fragment of the KB may be hard to envisage in the abstract.

7. CONCLUSION

In this paper we have identified the maintainability of legal KBS as an important factor in their successful exploitation. Unfortunately this issue has attracted too little attention to date. In the MAKE project we are producing a coherent strategy for the maintenance of such systems, embracing a methodology for knowledge analysis, recommendations for representation principles, and a set of tools to support the amendment of the knowledge base. Some useful tools for the maintenance of a KB have been sketched in this paper.

ACKNOWLEDGEMENT

The work described above was carried out as part of the MAKE Project, supported by the Information Engineering Directorate of the UK Department of Trade and Industry and the UK Science and Engineering Research Council. The project collaborators are ICL, the University of Liverpool and British Coal. The views expressed in this paper are those of the authors and may not necessarily be shared by the other collaborators.

REFERENCES

Bench-Capon, T. J. M. and Forder, J. M. (1991a) Knowledge representation for legal applications, in T. J. M. Bench-Capon (Ed.), *Knowledge Based systems for Legal Applications*, Academic Press, 1991, pp. 245—264.

Bench-Capon, T. J. M. (Ed.) (1991b) *Knowledge Based Systems for Legal Applications*, Academic Press, 1991.

Bench-Capon, T. J. M. and Coenen, F. P. (1991c) Exploiting isomorphism: development of a KBS to support British Coal Insurance claims, in *Proceedings of the 3rd International Conference on AI and Law*, Oxford 1991, ACM Press, pp. 62—68.

Bench-Capon, T. J. M. (1988) Applying legal expert systems techniques: practical considerations, in Duffin, P. H. *KBS in Government 88*, On Line Publications, 1988, pp. 205—214.

Bratley, P., Fremont, J., Mackaay, E., and Poulin, D. (1991) Coping with change, in *Proceedings of the 3rd International Conference on AI and Law*, Oxford 1991, ACM Press, pp. 62—68.

Coenen, F. P. and Bench-Capon, T. J. M. (1991) KBS development using X windows: the MADE development methodology. To be presented at UKUUG, 1991.

Duffin, P. H. (Ed.) (1988) *Knowledge Based Systems: Applications in Administrative Government*, Ellis Horwood, Chichester, 1988.

Rousset, M. (1988) On the consistency of knowledge bases: The COVADIS system, *Proceedings of ECAI 88.*

Routen, T. W. and Bench-Capon, T. J. M. (1991) Hierarchical formalisations, *International Journal of Man-Machine Studies*, July 1991.

Sergot, M. J., Sadri, F., Kowalski, R. A., Kriwaczek, F., Hammond, P. and Cory, H. T. (1986) The British Nationality Act as a logic program, *Communications of the ACM* **29**(5), 370—386.

Spirgel-Sinclair, S. and Trevena, G. (1988) The retirement pension forecast and advice system, in Duffin, P. H. *op. cit.* pp. 34—40.

Smith, J. C. and Deedman, C. (1987) The application of expert systems technology to case-based law, *Proceedings of the First International Conference of Artificial Intelligence and Law*, Boston, 1987, pp. 84—93.

Susskind, R. and Tindall, C. (1988) VATIA: Ernst and Whinney's VAT expert system, *Proceedings of the Fourth International Expert Systems Conference*, London, 1988.