** Note this document is not in the same format as the published version which was in two columns and with the ** dimensions and shape of the figures altered. However the text and content of the figures is identical.

A GRAPHICAL INTERACTIVE TOOL FOR KBS MAINTENANCE.

Frans Coenen and Trevor Bench Capon.

Department of Computer Science,
University of Liverpool,
Chadwick Building,
PO BOX 147
Liverpool,
L69 3BX,
United Kingdom.

ABSTRACT

Although KBSs are now well established, very little work has been carried out concerning the maintenance of these systems. In this paper a KBS maintenance tool is described designed to assist maintenance engineers in the task of maintaining the Rule Bases (RBs) of KBSs. The tool is directed at KBSs built using an intermediate representation, between the source and the final Rule base, which is isomorphic with the source i.e. that reflects the structure of the source data. Essentially, the Datamap tool described allows the maintenance engineer to navigate round the intermediate representation to identify suspect Rules. This is achieved using two directed biparteid graphs which graphically display the connections between elements within individual Rules and the connections between the Rules themselves. In addition a number of facilities are provided to allow the user to interact with the graphical displays and obtain answers to questions concerning KB maintenance. These facilities support maintenance activities where judgements are required on behalf of the maintenance engineer.

June 16, 1998

A GRAPHICAL INTERACTIVE TOOL FOR KBS MAINTENANCE.

Frans Coenen and Trevor Bench Capon.

Department of Computer Science,
University of Liverpool,
Chadwick Building,
PO BOX 147
Liverpool,
L69 3BX,
United Kingdom.

1. INTRODUCTION.

Knowledge Based Systems (KBSs) have been in existence for several decades now, but their general acceptance is still limited. It is suggested that one principal reason for this is the difficulty in maintaining such systems due to the nature of the expert knowledge used [1]. This tends to be dynamic, in that the knowledge will change as new discoveries are made and expert opinions alter over a period of time. This means that KBSs, if they are not maintained, quickly become obsolete or inaccurate. This is especially the case in legal domains where the knowledge may change overnight if legislation is altered.

The maintenance of KBSs tends to be adaptive maintenance rather than the corrective or perfective maintenance as associated with conventional systems, a categorisation first proposed in [2]. The approach to maintenance is therefore not the same since the emphasis is on responding to external changes. Further the structure of KBSs is such that traditional maintenance techniques are not applicable. Given the above a surprisingly small amount of research work has been carried out to investigate the maintenance of KBSs. What work has been carried out has largely been concerned with formal consistency checking of Rule Bases (RBs) such as in the COVADIS system [3], the debugging of rule bases such as provided by the TEIRESIAS system [4], or the syntactical editing of RBs within KBS development toolkits.

In this paper, a proposed support tool to assist the KBS maintenance engineer in the maintenance of Rule Bases (RBs) is described. The tool is part of a suite of tools under development as part of the Maintenance Assistance for Knowledge Engineers (MAKE) Project, further details of which can be found in 1. The MAKE project itself is a two year collaborative project between Liverpool University, ICL and British Coal. The tools are essentially designed to be used in conjunction with KBSs built using MADE (Make Authoring and Development Environment). This is a development technique aimed at producing an isomorphic intermediate representation [5], a brief overview of which is given in Section 2. Currently this intermediate representation is translated into an RB represented in Clausal Form. However the target representation could just as easily be the Horn Clause Subset used by Prolog or straight Production Rules popular in many expert system shells. The Datamap tool described here thus has potential for implementation on any Rule based representation.

An overview of the Datamap is given in Section 3 and some possible uses with respect to maintenance are suggested. To increase the usefulness of the Datamap the user will need to access the underlying information. To this end a number of additional facilities and an alternative view option are proposed. This interaction is described in detail in Section 4 and the proposed additional facilities in the following Sub-sections. The discussion includes suggestions concerning the usefulness of each facility. Essentially the Datamap tool gives a graphical view of the RB and allows the maintenance engineer to pose certain questions to it. We may distinguish two categories of problem that may arise with respect to RB maintenance, problems of structure and problems of content. Structural problems relate to the logical structure of the KB and are concerned with missing and redundant Rules, Clauses, circular Rule paths, etc. Problems of content are associated with the actual contents of the Rules. Essentially there are two possibilities with regard to mistakes of content: either the KB licences an inference which is not in fact legitimate or the KB fails to licence a desired inference. The proposed facilities provided with the Datamap are designed to address both questions of structure and content. The interactive facilities in particular support content maintenance where judgements are required on behalf of the maintenance engineer.

In the Final Section of this paper some conclusions are drawn including the results of the demonstration of a proto-type Datamap that has been developed.

2. THE MADE ISOMORPHIC DEVELOPMENT METHODOLOGY.

MADE is a KBS development environment based on KANT (Knowledge engineers assistANT). This is a hypertext like knowledge analysis tool originally built to assist in the development of a KBS to provide decision support for Department of Social Security (DSS) Adjudication Officers in the assessment of clams for benefits in local DSS offices [6]. The system was called the Local Office Demonstrator (LOD) System and was one of three applications built as part of the Alvey DHSS Large Demonstrator Project aimed at demonstrating the viability of KBS decision support in large, legislation based organisations [7]. It is ideally suited to the construction of KBSs in domains, such as legal domains, where the source knowledge is comprised of a significant amount of textual material, by assisting the knowledge engineer in the analysis of these source documents. The philosophy behind MADE is that knowledge can be viewed at different levels and that this ability will greatly enhance system maintenance. An advantage that is utilised by the Datamap.

The first level at which knowledge can be viewed is at the source level. KANT allows the knowledge engineer to browse one or more source documents in individual windows so that relevant Sections can be identified. These can then be analysed. The Result of the analysis is a set of Rules and a hierarchical Data Base, called the Class Hierarchy. The Rules in the Rule Base typically all branch out from a single top level node. This represents the Root Proposition which the KBS is intended to establish. The Class Hierarchy represents the vocabulary of the domain and consists of a top level Class with Sub-Classes branching from it describing different Object types. Each Object has a unique Class name and a number of slots describing Attributes of that Object and the possible Values that these Attributes may have for that Object. A feature of the Class Hierarchy is that Sub-Classes inherit Attributes of Super-Classes so as to ensure that a Sub-Class is a strict specialisation of its Super-Class. In KANT, inherited Attributes cannot be cancelled as is the case with some other development environments such as KEE. The Datamap described here is designed to assist in the maintenance of RBs and is not primarily concerned with the nature of the Class Hierarchy.

After analysis the Rule Base and Class Hierarchy are both represented using an intermediate representation. called MIR (Make Intermediate Representation). This is the second level view of the knowledge. The nature of this intermediate representation and the advantages to be gained from this type of representation have been discussed elsewhere ([8], [5] and [9]). It is thus not the intention of this paper to continue the discussion. Suffice to say that the more flexible syntax of the intermediate representation assists the knowledge engineer by allowing him to express the knowledge in the most natural way, and by maintaining a clear correspondence with the source.

The MIR Rule Base and Class Hierarchy are compiled into a target representation, the third level view of the knowledge. Currently this consists of a Clausal form referred to as CMIR (Compiled MIR). The Datamap allows the maintenance engineer to intercat and explore these second and third levels of knowledge representation in association with the RB.

The design of MADE revolves around three base windows, the KANT, MADE and MAPPE (Make APPlication Environment) Windows. From the KANT base window structures and sources can be selected and analysed. This is where the Knowledge Analysis for the Application is carried out. The MADE base window incorporates a results area in which sequences of messages produced during compilation of the MIR into CMIR are displayed. The MADE window is also the user interface through which the MAKE maintenance tools are invoked, including the Datamap described here. A full discussion of the tools available under the MAKE project is given in 1. The MAPPE Base Window provides the end user view of the application. It is divided into a hierarchy of TOPIC windows. These consist of forms to filled in by the user and can be viewed as Finite State Machines (FSM) in that Forms pass from one state to another depending on the operations performed on them. The states change according to either the user supplying an answer or by Rule initiated answers. The user can choose the order in which to address the various Sub-tasks and links to the written sources mean that the appropriate sections of legislation and guidance can be consulted at any time. This aspect is discussed in ([9]).

Finally, with respect to the RB and the intermediate representation, the following definitions, used in Sections 3 and 4 below, should be noted:-

(a) Rule.

This is an expression comprising Propositions, typically EAVs, and logical connectives. It consists of a head and a tail separated by an if or iff (read as if and only if). The head contains a single Proposition the truth or falsity of which is determined according to the truth or falsity of the Propositions contained in the body and their connectives.

(b) Clause.

This is an expression in standard Clausal Form, with a Proposition as head and a set of conjoined Propositions as tail (body). The Head Proposition is true if all the Propositions in the body are true. A single Rule will typically be equivalent to several Clauses.

(c) Proposition.

A Proposition is a triple comprising an Entity, an Attribute and a Value. The Entity may be a constant (the name of an instance) or a variable (typed to some Class in the Class Hierarchy). The Attribute will be a slot associated with the Class of which the

Entity is a member. The Value will be a sub-range of the possible Values for the slot as defined in the Class of which the Entity is a member. Note that the Class Hierarchy constrains what Propositions are possible.

(d) Root Propositions and Attributes.

A Root Proposition is a Proposition which appears in the head of some Clause, and does not appear in the tail of any Clause. A Root Attribute is an Attribute associated with one or more Root Propositions. Thus a Root Proposition represents the task which the RB is designed to establish.

(e) Leaf Propositions and Attributes.

A Leaf Proposition is a Proposition which appears in the body of one or more Clauses, and in the head of no Clauses. A Leaf Attribute is an Attribute associated with one or more Leaf Propositions. The significance of leaves is that Values associated with them must be supplied to the system, not deduced. Values may be supplied either by explicit user input, or by the association of an instance with a Class the definition of which makes the Proposition true.

An example of MIR is given in the Appendix. This fragment of RB has been taken from a genuine KBS designed to provide British Coal's accident claims officers and investigators with decision support. This was also developed as part of the MAKE project because it was felt that a real application was required on which proposed maintenance tools could be tested.

3. OVERVIEW OF THE DATAMAP.

The Datamap is a directed (from left to right) biparteid graph which links Attributes and Rules. On the left hand side we will find the start Attribute(s), with arcs to the Rules in which the start Attribute(s) act as the head. Arcs from these Rules will connect to Attributes in the body of these Rules. Further arcs will link to Rules in which these Attributes appear as the head (i.e. the Attribute concerning which the Rule permits inference), and so on until the Leaf Attributes are reached. It is proposed that the Datamap be invoked by a top level command selected from a KBS maintenance tool menu and that it be drawn, by default, starting with the Root Attributes. An option will be provided however to allow the maintenance engineer to select an alternative start Attribute, should he wish to investigate a sub-tree. Figure 1 shows a Datamap drawn in default mode representing the example Rule Base given in the Appendix. Note that only three levels of Attribute-Rule pairs are drawn. This is because of the limited amount of screen space that it is envisaged will be available. The information displayed in these three levels, although limited, is enough to allow the user to navigate through the RB. Using the Unfold/walk Down and the Fold/Walk Up Options described below further levels may be accessed.

The thing shown most clearly by the Datamap in Figure 1 is the relationship between Attributes and Rules. It shows in which Rules Attributes appear (through the arcs), and the role that they play in those Rules (through the direction of the arcs). By following a path through the Datamap it is possible to determine the Leaf Attributes into which a Root Attribute ultimately unfolds using the Unfold/Walk Down Option (see Sub-section 4.1). This then makes it relatively easy to find a Rule which may be problematic if the behaviour of an Attribute is not as expected.

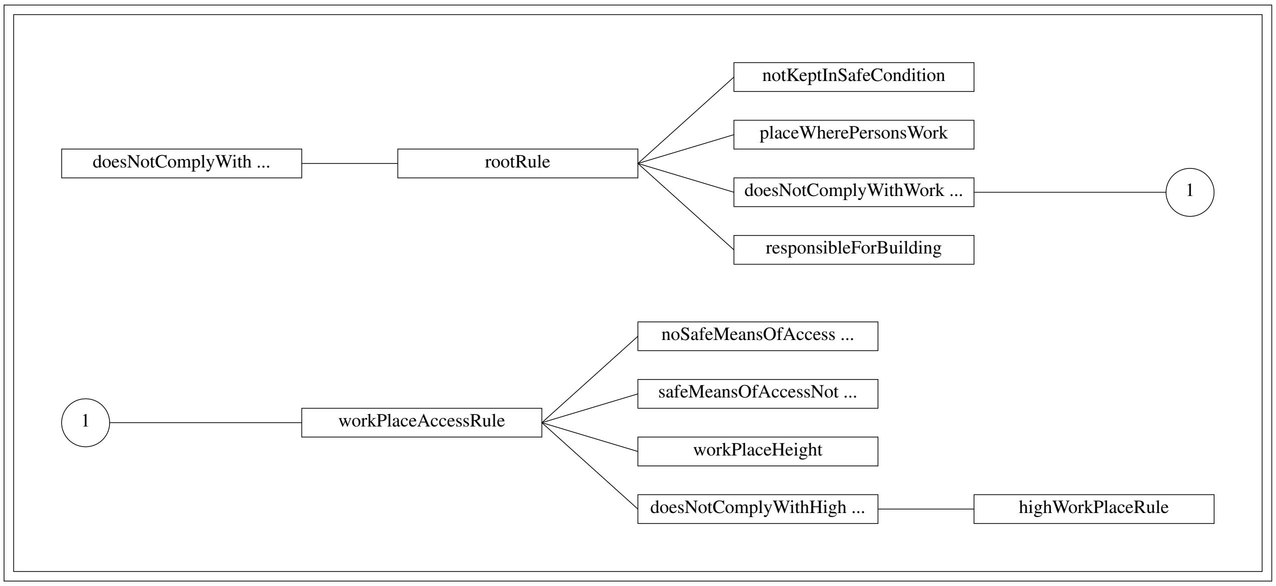
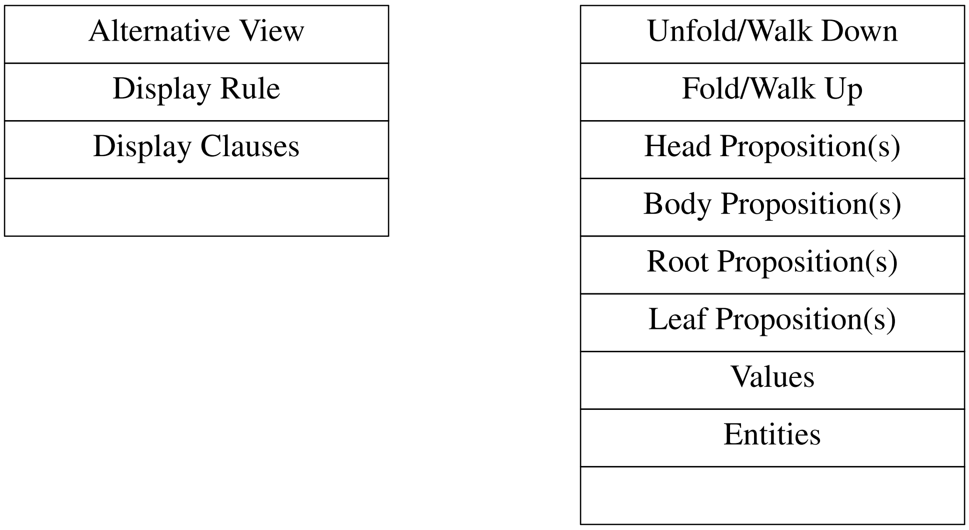


FIG 1: DATA MAP EXAMPLE 1



(a) Rule Menu (b) Attribute Menu

FIG 2: DATA MAP MENUS

4. USER INTERACTION.

Having produced a Datamap as described in the previous Section, the tool provides for a large amount of user interaction by "clicking" on any Attribute or Rule Identifier of interest. The result will be "pop up" menus of the form shown in Figure 2 giving the user the choice of a number of options. Each of the options available is described in further detail in the following Sub-Sections.

4.1. THE UNFOLD/WALK DOWN AND FOLD WALK UP OPTIONS.

The Datamap as described above is only drawn to three levels. If the user wishes to display further levels he/she can click on an Attribute and select the "Unfold/Walk Down" Option (Figure 2). For example if, in Figure 1, the user selects the Attribute "doesNotComplyWithHigh ... " and the "Unfold/Walk Down" Option then the result will be as depicted in Figure 3. Alternatively if the user wishes to walk back up the Datamap to find defined Attributes the the "Fold/Walk Up" Option can be selected.

These Options allow the user to navigate through the Datamap. Firstly to explore how Attributes and Rules are connected to each other and secondly to trace a Root Attribute through to the Leaf Attributes associated with it and vice versa.

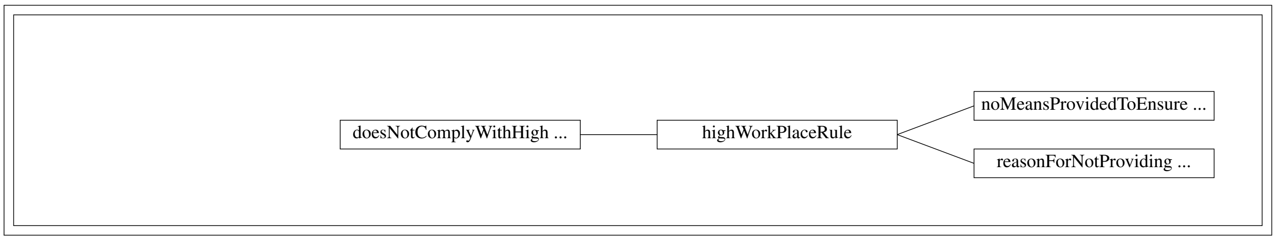


FIG 3: DATA MAP EXAMPLE 2

4.2. ALTERNATIVE VIEW OPTION.

There is a need for the maintenance engineer to also be able to inspect the relationships between Clauses and Propositions. This is the lowest level at which the knowledge displayed can be viewed and governs the "fine grain" behaviour of the system. An Alternative Datamap is therefore required and is produced in a separate window by selecting the Alternative View Option. This Datamap is drawn starting with the Head Proposition(s) contained in the Rule Identifier selected. The Clause Identifiers in which this Proposition appears are then produced and displayed. This is then followed by the tail Propositions associated with these Clauses, followed by the Clauses in which these tail Propositions appear as the head. Again because of the limited amount of space available this process only continues until three levels of Proposition-Clause pairs have been displayed or until a Leaf Proposition is reached. Thus if for example the "rootRule" Identifier is selected in Figure 1 and the Alternative View Option selected from the resulting menu, an Alternative Datamap of the form given in Figure 4 will be produced.

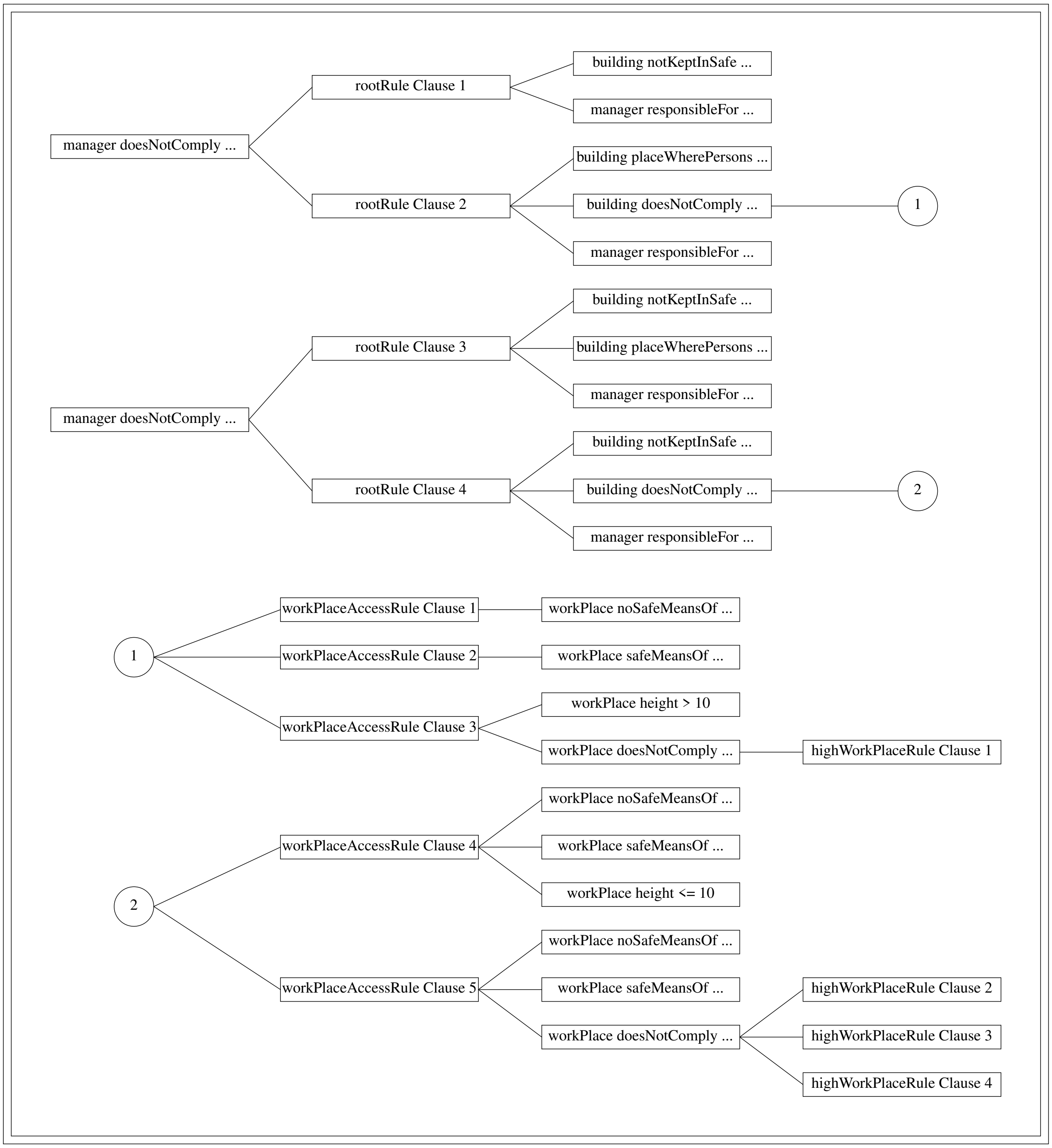


FIG 4: ALTERNATIVE DATA MAP EXAMPLE 1

As discussed previously the initial Datamap can be used to identify suspect Rules. These represent the second level view of the knowledge below the source documents. When identified these Rules can be examined to determine why their behaviour is unexpected. At this stage, viewing the Rule as its set of consequent Clauses is probably to be desired; typically the aberrent behaviour will be as a result of either some unwanted, malformed or missing Clause, a situation which may be difficult to identify by inspection of the

Rule alone. However the provision of an Alternative Datamap Option will alleviate this situation by allowing the user to switch to the detailed Clausal Representation. In contrast if we consider Horn Clause or Prolog representations where Rules or Clauses are syntactically the same, then appreciation of the RB is at a single level: knowledge of the source structure provides no guidance. In the Datamap we can navigate at a more abstract level, calling up the detail when it is needed.

As with the initial Datamap each Proposition Identifier has a pop up menu associated with it as illustrated in Figure 5. By selecting the "Unfold/Walk Down" Option we can unfold the data map in the same way as described for the initial Datamap. Similarly we can walk back up the Datamap using the "Fold/Walk Up" Option.

Unfold/Walk Down
Fold/Walk Up

FIG 5: ALTERNATIVE DATA MAP MENU

4.3. HEAD PROPOSITION(S).

The selection of the Head Propositions Option produces a list of Head Propositions and their associated Clause Identifiers in a separate window. The Option may be used to allow the maintenance engineer to inspect the Head Propositions associated with Attributes and identify the Clauses in which they are contained without referring to the Alternative Datamap.

4.4. BODY PROPOSITION(S).

The Body Propositions Option displays the Body Propositions in which a selected Attribute appears and the Clause Identifiers associated with these Propositions. The Option will allow the user to inspect the Propositions contained in the body of a selected Rule, without referring to the Alternative Datamap.

4.5. ROOT PROPOSITION(S).

The Root Propositions Option is similar to the Head Propositions Option except that it returns "NONE" if the selected Attribute can not be linked to a Root Proposition. This Option could be used to determine whether an Attribute, and hence the associated Proposition, is a Root Attribute or Root Proposition.

The Option would also assist the maintenance engineer when determining whether the RB is adequate to establish the information that the application requires by inspecting the Root Propositions. The Root Propositions of the RB must be exhaustive with respect to the possible Values of the Root Attributes required by the task, and must partition the possible Values in the desired way.

Lastly the Root Proposition(s) Option would allow the maintenance engineer to determine whether the RB establishes anything not required. It may be that an Attribute we are not interested in appears as a Root with respect to the RB, in which case we will waste inference in establishing it.

4.6. LEAF PROPOSITION(S).

The Leaf Proposition(s) Option is similar to the Body Proposition(s) Option described above except that the Datamap will return "NONE" if the selected Attribute is not a Leaf Attribute.

This Option would allow the maintenance engineer to determine whether the information required by the RB to establish a desired Propositions can be obtained. By examining the Leaf Propositions the engineer can see whether they are things that the system user can be expected to supply. If they are not, Rules must be added to the RB to unfold them into ascertainable information.

The Option will also be of use in determining whether all Leaf Propositions in the KB are required to establish a desired conclusion i.e. some Leaf Proposition may be redundant. This will involve finding all Leaf Propositions who appear only on paths leading to non-desired Root Attributes.

Finally this Option can be used to check for missing branches. The structure of the RB and Class Hierarchy must be such that all Values for Leaf Propositions can be obtained either by direct reference to the the Class Hierarchy or by explicit input by the user i.e. the completion of appropriate forms. If not further Rules may have to be added or maintenance carried out on the Class Hierarchy and/or forms available. In the MAKE application user input is defined through a task description. Linking this to the Datamap automates the finding of missing branches.

4.7. VALUES.

When selected the Values Option is used to display the Values that an Attribute may have by referring to the Propositions in which the Attribute of interest is contained. This Option would also assist in checking that there are no missing branches in the RB as described in the previous Section by allowing the engineer to match possible Attribute Values with the contents of the Class Hierarchy. There may be a possibility of introducing some automation here.

4.8. ENTITIES.

The Entities Option is similar to the Values Option described in the previous Section except that it returns the Entities associated with an individual Attribute, again by reference to the Propositions in which the Attribute is contained.

4.9. DISPLAY RULE.

When carrying out RB maintenance it will eventually be necessary for the engineer to inspect individual Rules as they are represented in the intermediate representation. This is facilitated by the "Display Rule" Option. The Option simply displays an entire Rule, in a separate window, by selecting a Rule Identifier of interest.

A possible use of this Option would be to allow the maintenance engineer to check that individual Rules represent the source data. Using an isomorphic approach to KBS development, as espoused by KANT, individual Rules should represent individual blocks of source, which can be inspected at this point by links established in the analysis phase.

This Option will also allow the maintenance engineer to inspect the "and" and "or" connections in the body of a Rule. Something that is not identifiable by inspection of the Datamap itself.

4.10. DISPLAY CLAUSE.

The Display Clause Option is similar to the previous Option except that all the Clauses that a Rule compiles into are displayed. This may be required during a maintenance session again to allow the maintenance engineer to inspect the affect of the "and" and "or" connections in Rules or simply to examine specific RB Clauses.

5. CONCLUSIONS.

A proto-type Datamap has been in operation on a pilot version of the British Coal KBS which has been developed as part of the MAKE project. This proto-type was of the form of the basic Datamap described in Section 3 and did not display all of the additional facilities described in Section 4. However in this basic form the proto-type Datamap has been demonstrated to all project members and has been very well received. It was generally considered to be a essential maintenance tool and that the additional facilities described here would make it into a very effective tool. We therefore feel that the Datamap would be a useful addition to any KBS toolkit or development methodology.

It should however be noted that redundant Rules or Clauses will not appear in the Datamap and that subsumed Clauses will not be identifiable. Additional tools have been specified as part of the MAKE project to be appended to the Datamapand to allow the maintenance engineer to identify such Rules and Clauses [10]. Finally it should also be noted that the Datamap would also allow for easy identification of cyclic paths, both at the Rule and Clause level, although this is not really an issue in systems built using the MADE methodology.

6. ACKNOWLEDGEMENTS.

The work described here is being 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. In addition, the authors would like to express special thanks to Chris Burton for development and demonstration of the original proto-type Datamap.

7. APPENDIX (MAKE RB FRAGMENT).

```
"rootRule"

(manager doesNotComplyWithRegulations is true

iff

(((building notKeptInSafeCondition is true)

or

((building placeWherePersonsWork is true)

and

(building doesNotComplyWithWorkPlaceAccessRule is true)))

when

(manager responsibleForBuilding includes building)))
```

```
"workPlaceAccessRule"

(workPlace doesNotComplyWithWorkPlaceAccessRule is true

iff

((workPlace noSafeMeansOfAccessProvided is true)

or

(workPlace safeMeansOfAccessNotMaintained is true)

or

((workPlace height > 10)

and

(workPlace doesNotComplyWithHighWorkPlaceRule is true))))


"highWorkPlaceRule"

(highWorkPlace doesNotComplyWithHighWorkPlaceRule is true

iff

((highWorkPlace noMeansProvidedToEnsureSafety is true)

and

(highWorkPlace reasonForNotProvidingMeansToEnsureSafety

                                    is other)))
```

REFERENCES.

[1] Bench-Capon, T.J.M. & Coenen.Practical Application of KBS To Law: The Crucial Role of Maintenance in van Noortwijk, C., Schmidt, A.H.J. & Winkels, R.G.F. (Eds) Legal Knowledge Based Systems: Aims for Research and Development, Koninklijke Vermande BV, pp 5-17,1991.

[2] Swanson, E.B. The Dimensions of Maintenance. Proceedings, 2nd International Conference on Software Engineering, IEEE, Oct. 1976.

[3] Rousett, M. On The Consistency of Knowledge Bases: The COVADIS System. Proceedings of ECAI 1988.

[4] Davis, R. Interactive Transfer of Expertise. In Buchannan, B.G. & Shortliffe, E.H. Rule-Based Expert Systems: The MYCIN Experiments of The Stanford Heuristic Programming Project, 1984. Addison-Wesley, 1984.

[5] Routen, T.W. & Bench-Capon, T.J.M. Hierarchical Formalisations. Forthcoming in International Journal of Man-Machine Studies, 1991.

[6] Storrs, G.E. & Burton, C.P. KANT, A Knowledge Analysis Tool. ICL Technical Journal, Vol 6, No 3, May 1989.

[7] Forder, J.M. & Taylor, A.D. The Local Office System. Forthcoming in Bench-Capon, T.J.M. (Ed), Knowledge based Systems for Legal Applications. Academic Press, pp 139-164, 1991.

[8] Bench-Capon, T.J.M. Deep Models, Normative Reasoning and Legal Expert Systems Proceedings of the Second International Conference on AI and Law, Vancouver. ACM Press, 1989.

[9] Bench-Capon, T.J.M. & Forder, J.M. Knowledge Representation for Legal Applications. Forthcoming in Bench-Capon, T.J.M. (Ed), Knowledge based Systems for Legal Applications. Academic Press, pp 245-264, 1991.

[10] Coenen, F. and Bench-Capon, T.J.M. Specification For Redundancy and Subsumption ID Tool. Make Report 3/91, Deliverable D5a[1]. Department of Computer Science, Liverpool University, 1991.