Verification of rule-bases using incidence matrices: the IMVER system

F. Coenen, A. Taleb-Bendiab & R. Forster

\textit{Department of Computer Science, Liverpool University, Chadwick Building, P.O. Box 147, Liverpool L69 3BX, UK}

\textit{Department of Mechanical Production and Chemical Engineering, Manchester Metropolitan University, John Dalton Building, Chester Street, Manchester M1 5GD, UK}

ABSTRACT

The verification of Knowledge Base Systems (KBSs) used in the engineering industry should be a vital part of the KBS life-cycle. However the operation of KBS is often found to be, at best, ineffective and at worst wrong. As a result increased maintenance costs are experienced. Although it is not possible to absolutely guarantee that a KBS is error free we can gain confidence in its operation using verification tools. Here we describe the IMVER (Incidence Matrix VERification) system, a system for verifying rule-bases associated with KBSs. As the name suggests the system is based on the concept of incidence matrices, a technique where by the number of propositions occurring in individual rules within a rule-base can be represented using a matrix. Essentially the system takes, as input, a rule-base, converts this into an incidence matrix, checks for anomalies and errors and produces a commentary as output. The current version of the system checks for redundancy, subsumption and inconsistency. As a result these anomalies can be put right, thus leading to an enhanced degree of confidence that can be placed on the system when delivered to the end user as well as a corresponding decrease in future maintenance costs.
1. INTRODUCTION

The erroneous operation or failure of a Knowledge Based System (KBS) used in the engineering industry, whether it be manufacturing or production engineering, will at best result in lost time and hence increased costs. In a "worst case scenario" it may result in the complete shutdown of plant. Further, in some manufacturing industries, such as chemical production, the ecological ramifications can be far reaching; consider the event where a KBS fails to apply an emergency shut down procedure. The verification of KBS prior to delivery is therefore vital not only to prevent local financial losses but also to address the, far wider reaching, global consequences of failure. By verifying a KBS we cannot of course guarantee that it is error free. However we can increase the confidence level that we may have in a KBS, we may even attempt to qualify this numerically.

Verification is an integral part of the the development of all software systems and there are many techniques available, aimed at a variety of different types of system. Traditional verification techniques are generally concerned with software testing. These techniques can be conveniently divided into black box and white box techniques. The first includes experiences such as function testing and equivalence testing; and the second techniques such as statement testing, branch testing and path testing. However traditional software verification techniques do not translate easily to KBS. We can point to a number of reasons for this, two of the most significant are:

- The program flow and control in KBSs is hidden.
- The nature of KBSs is such that exhaustive verification is impractical.

At present there is a great deal of research activity concentrated on the field of KBS verification; this is well illustrated by the attendance at the recent EUROVAV-91 and -93 conferences. Well documented approaches include normal form techniques (Charles\(^1\)), decision table methods (Cragen and Steude\(^2\)), KB reduction (Ginsberg\(^3\)), generic rule systems (Chang et al.\(^4\)), and the use of static inspection tools (Coenen\(^5\)). A further interesting development is the use of incidence matrices. The general principle behind incidence matrices is well outlined in Landauer\(^6\). There are a number of examples where the use of incidence matrices has been incorporated into a validation and verification systems. One such system is described by Agarwal and Tanniru\(^7\) who incorporate the concept into a Petri-net based approach to rule-base verification. The system interprets the incidence matrix to demonstrate "subsumption", "redundancy" and "completeness" (the terms are given in quotes because
Agarwal and Tanniru do not define subsumption and redundancy in precisely the same way as used in this document).

Currently a collaborative research project is underway between the computer science department at Liverpool University and the department of mechanical, production and chemical engineering at Manchester Polytechnic to develop and build a system, based on the work carried out by Agarwal and Tanniru concerning incidence matrices, to address the verification of KBSs used in manufacturing engineering. The result is the Incidence Matrix Verification (IMVER) system.

2. NOTE ON VERIFICATION

There are many definitions of verification, Satre and Massey\(^8\) gave an extensive review of these definitions at AIENG91. We have adapted the definition, with respect to traditional software systems, originally given by Boehm\(^9\). Boehm defines verification by posing the question:

*Are we building the product right?*

Thus verification is the process of checking that the system meets the requirements definition/specification.

With regard to KBSs, where (in the traditional sense) no meaningful requirements specification exists, the process of verification described in these terms becomes meaningless. We have thus defined KBS verification as the process of checking that the system's operation is right, i.e. that results are arrived at in the correct manner. The work described here is thus concerned with the verification of rule-bases.

3. AN EXAMPLE INCIDENCE MATRIX

Consider the following rule taken from a rule-base used for the selection of machine tools used in manufacturing operations:-

```
RULE 1
--------
IF Workpiece isa roundObject
    AND Workpiece has rotationalWorkMotion
    AND Tool has noWorkMotion
THEN Lathe could be selected
```

This rule consists of four propositions, we can label the conditional
propositions \( b, c \) and \( d \) and the concluding proposition \( a \). Thus:

\[
\text{IF} \ b \ \text{AND} \ c \ \text{AND} \ d \ \text{THEN} \ a
\]

More concisely we can express this as:

\[
b \land c \land d \Rightarrow a
\]

Where the \( \Rightarrow \) symbol is read as "implies". Thus \( b \) equates to the proposition \( \text{Workpiece is a roundObject} \), \( c \) to the proposition \( \text{Workpiece has rotationalWorkMotion} \) and so on. The consequent \( \text{Lathe could be selected} \) is represented by the letter \( a \). This will be the rule notation used in the rest of this document. Note that only propositional conjunctions are considered therefore rules containing disjunctions will require translation into a conjunctive form using standard logical equivalences.

The above notation can be represented as a matrix:

\[
\begin{pmatrix}
1 & 1 & 1 & 1
\end{pmatrix}
\]

Where each column represents a proposition and each row a rule. The matrix indicates the number of occurrences of each proposition in the rule.

If we now consider two further rules taken from the machine tool selection rule-base:

RULE 2
-----
\[
\begin{array}{l}
\text{IF} \quad \text{Workpiece has axisOfSymmetry} \\
\quad \text{AND} \quad \text{Workpiece isKindOf solidObject} \\
\text{THEN} \quad \text{Workpiece isa roundObject}
\end{array}
\]

RULE 3
-----
\[
\begin{array}{l}
\text{IF} \quad \text{Tool has horizontalMotion} \\
\quad \text{AND} \quad \text{Tool has SmallFeedRate} \\
\text{THEN} \quad \text{Tool has noWorkMotion}
\end{array}
\]

or using our rule notation:
These three rules can be expressed as a single matrix of the form:-

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

If we now multiply this matrix by the transpose of itself:

\[
\begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1
\end{bmatrix}
= \begin{bmatrix}
4 & 1 & 1 \\
1 & 3 & 0 \\
1 & 0 & 3
\end{bmatrix}
\]

The resulting matrix shows the number of shared propositions between the rules in the rule-base and is referred to as the incidence matrix. The concept of incidence matrices is used in the IMVER system to verify KBSs by identifying anomalies within the associated rule-base.

4. ERRORS IN RULE-BASES

There is much discussion concerning the errors and anomalies that can occur in a rule-base with many authors presenting "definitive" lists (see Preece and Shinghal\(^\text{10}\), Ginsberg\(^3\), Cragen and Steudel\(^2\)). We do not wish to repeat the exercise in this paper, however to provide a common forum of understanding it is essential that we present definitions for at least the principle anomalies addressed by IMVER.

**Redundancy (Missing Links)**

A rule-base contains redundancy if a rule, or group of rules, plays no part in establishing any root proposition(s). This situation will exist if the rule is not "connected" to the rest of the rule-base.

**Duplication**

We define duplication (absolute subsumption) as the situation where a rule-base contains two or more rules which have identical antecedents and consequents except, perhaps, for the order of the propositions.
Subsumption

Complex subsumption occurs when one rule is a more specialised case of another. Thus when two rules have duplicate consequents and the antecedent of one is a subset of the antecedent of the other, or when two rules have duplicate antecedents and the consequent of one is a subset of the consequent of the other, or a mixture of these states.

Inconsistency

An inconsistency occurs in a rule-based system when we can draw contradictory conclusions from the same set of facts.

These definitions have deliberately been kept short, readers requiring a more complete explanation for their derivation are referred to Coenen and Bench-Capon.

5. SYSTEM ARCHITECTURE

Figure 1 shows the top level architecture and operation of the IMVER systems. If we view the systems simply as a "black box" the input will be a rule-base and the output a "commentary". If we consider the system as a "white box" we can identify a number of top-level parts, namely a translator and a number of anomaly testing functions. The translator takes a "raw" rule-base and translates it into an incidence matrix, as described above, and also calculates a number of additional parameters for each rule (see below). The anomaly testing functions are then applied to the incidence matrix in sequence. Currently we test for redundancy, missing links, duplication, subsumption and inconsistency.

Figure 1: Block Diagram Illustrating Architecture and Operation of IMVER
6. REDUNDANCY AND MISSING LINKS

The first anomaly identified by the IMVER system is redundancy. If we return to the incidence matrix derived in section 3 above:

\[
\begin{pmatrix}
4 & 1 & 1 \\
1 & 3 & 0 \\
1 & 0 & 3
\end{pmatrix}
\]

What can we determine from this matrix? Firstly the leading diagonal indicates the number of propositions in each rule, i.e. 4 in Rule 1 and 3 in Rules 2 and 3. Thus the columns and rows represent Rules 1, 2 and 3 respectively. The remaining elements in the incidence matrix then indicate the number of shared propositions between rules, thus Rules 2 and 3 share one proposition each with Rule 1 while Rules 2 and 3 share no propositions with each other. We will identify individual elements in the resulting matrix by the notation \(R(i,j)\) where \(i\) equates to the rule number represented by a particular row and \(j\) the rule number equating to a particular column. Thus the first element, \(R(1,1)\) is equal to 4, while element \(R(1,2)\) is equal to 1.

The significance of the number of propositions shared by a rule is that if a rule shares no propositions with any other rule it is disconnected, i.e. some error exists. If a rule shares only one proposition with another rule, and it is not a root or a leaf rule, then the rule-base is also disconnected, i.e. a missing link exists. Consider the following disconnected rule-base:

\[
\begin{align*}
b & \Rightarrow a \\
e & \Rightarrow b \\
g & \Rightarrow i
\end{align*}
\]

Multiplying the matrix representing this rule-base by the transpose of itself we get an incidence matrix of the form:

\[
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 \\
1 & 1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 0
\end{pmatrix}
= \begin{pmatrix}
3 & 1 & 0 \\
1 & 3 & 0 \\
0 & 0 & 3
\end{pmatrix}
\]
Inspection of the result shows that Rule 3 only shares propositions with itself, we can therefore say that Rule 3 is redundant.

If we consider another (connected) example rule-base:-

\[
\begin{align*}
  b & \Rightarrow a \\
  c & \Rightarrow b \\
  d & \Rightarrow c \\
  e & \Rightarrow d
\end{align*}
\]

Then:

\[
\begin{bmatrix}
  1 & 1 & 0 & 0 & 0 \\
  0 & 1 & 1 & 0 & 0 \\
  0 & 0 & 1 & 1 & 0 \\
  0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  1 & 1 & 0 & 0 \\
  0 & 1 & 1 & 0 \\
  0 & 0 & 1 & 1 \\
\end{bmatrix} =
\begin{bmatrix}
  2 & 1 & 0 & 0 \\
  1 & 2 & 1 & 0 \\
  0 & 1 & 2 & 1 \\
  0 & 0 & 1 & 2 \\
\end{bmatrix}
\]

Here we can determine that the rule-base is connected because the root and leaf rules (Rules 1 and 4) share at least one proposition with another rule and the body rules at least two. If we consider a similar but disconnected rule-base:-

\[
\begin{align*}
  b & \Rightarrow a \\
  c & \Rightarrow b \\
  d & \Rightarrow f \\
  e & \Rightarrow d
\end{align*}
\]

Then:

\[
\begin{bmatrix}
  1 & 1 & 0 & 0 & 0 & 0 \\
  0 & 1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 1 & 0 & 1 \\
  0 & 0 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
  1 & 0 & 0 & 0 \\
  1 & 1 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 1 \\
  0 & 0 & 0 & 1 \\
  0 & 0 & 1 & 0 \\
\end{bmatrix} =
\begin{bmatrix}
  2 & 1 & 0 & 0 \\
  1 & 2 & 0 & 0 \\
  0 & 1 & 2 & 1 \\
  0 & 0 & 1 & 2 \\
\end{bmatrix}
\]

Here rule 3, which (say) we know is a body rule, shares only a single proposition with another rule hence we can say that the rule-base is also disconnected.

NOTE: A root rule is a top level rule designed to establish some end proposition. A leaf rule is a rule which does not obtain data from other
rules but from some other source, for example user input. A body rule is then any other rule that is not a root or a leaf, i.e. a body rule is a rule contained in the body of a rule-base.

7. SUBSUMPTION AND DUPLICATION

Agarwal and Tanniru used the incidence matrix technique to identify subsumption (including duplication as defined here). This approach has been adopted, and refined, in the IMVER system. If we have an existing rule-base:-

\[
\begin{align*}
  b & \Rightarrow a \\
  d & \Rightarrow b
\end{align*}
\]

and we wish to add a rule:-

\[
\begin{align*}
  e & \Rightarrow c
\end{align*}
\]

We can multiply the matrix representing the existing rule-base by the transpose of the matrix representing the new rule to produce an incidence matrix as follows:-

\[
\begin{pmatrix}
  1 & 1 & 1 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 1 & 1 & 0 & 0
\end{pmatrix}
\times
\begin{pmatrix}
  0 \\
  0 \\
  1 \\
  0 \\
  0 \\
  1 \\
  1
\end{pmatrix}
= \begin{pmatrix}
  1 \\
  0
\end{pmatrix}
\]

From the above this tells us that the new rule shares a single proposition with Rule 1, assuming the new rule is a leaf rule this means that the rule is connected.

If we now add another rule, Rule 4, which is a duplicate of Rule 3, and carry out the same matrix multiplication the resulting matrix will be:-

\[
\begin{pmatrix}
  1 & 1 & 1 & 0 & 0 & 0 & 0 \\
  0 & 1 & 0 & 1 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 1 & 1
\end{pmatrix}
\times
\begin{pmatrix}
  0 \\
  0 \\
  1 \\
  0 \\
  0 \\
  1 \\
  1
\end{pmatrix}
= \begin{pmatrix}
  1 \\
  0 \\
  3
\end{pmatrix}
\]
From this we can only tell that the new rule shares 3 propositions with Rule 4. If we now consider two variables (also identified by Agarwal and Tanniru), \( N(i) \) and \( N(j) \) which represent the total number of propositions in an existing rule \( i \) and the total number of propositions in a new rule \( j \). We can then demonstrate duplication using the following identity:

\[
\text{if } N(i) = N(j) = R(i,j) \text{ then duplication exists.}
\]

In our example \( N(1), N(2), N(3) \) and \( N(4) \) and the element \( R(3,4) \) in the resulting incidence matrix are all equal to 3. Applying the identity:

\[
N(3) = N(4) = R(3,4)
\]

we can conclude that rules 3 and 4 are duplicates.

We can apply similar identities to subsumption. If we again consider the two rules:

\[
\begin{align*}
\text{b c } & \rightarrow \text{ a} \\
\text{d e } & \rightarrow \text{ b}
\end{align*}
\]

and add the rule:

\[
\text{d } \rightarrow \text{ b}
\]

which subsumes rule 2 the resulting calculation is as follows:

\[
\begin{pmatrix}
1 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
0 \\
1 \\
0 \\
1 \\
0
\end{pmatrix}
= \begin{pmatrix}
1 \\
2
\end{pmatrix}
\]

From the resulting matrix we can firstly discern connectivity (i.e. no missing links and no redundancy) and no duplication. Further, we can construct an identity:

\[
\text{if } N(i) > N(j) \text{ and } R(i,j) = N(j) \text{ then rule (j) subsumes rule (i)}
\]
Conversely we can also construct an identity:

\[
\text{if } N(j) > N(i) \text{ and } R(i,j) = N(i) \\
\text{then rule (i) subsumes rule (j)}
\]

For example if we add the rule:

\[
d \rightarrow e \rightarrow b
\]

to the rule-base:

\[
b \rightarrow c \rightarrow a \\
d \rightarrow b
\]

We can multiply the matrix representing the existing rules by the transpose of the matrix representing the new rule to produce an incidence matrix of the following form:

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
0 \\
1 \\
0 \\
1 \\
1
\end{pmatrix}
= 
\begin{pmatrix}
1 \\
2
\end{pmatrix}
\]

Here \( N(3) \) is greater than \( N(2) \) and \( R(2,3) \) is equal to \( N(2) \), therefore we can say that Rule 2 subsumes Rule 3.

8. INCONSISTENCY

Having considered subsumption and redundancy the third type of anomaly identified by the IMVER system is inconsistency. If we consider a (deliberately flawed) rule-base of the form:

\[
a \rightarrow b \rightarrow c \\
a \rightarrow b \rightarrow d \\
e \rightarrow f \rightarrow b
\]

Note that given \( a \) and \( b \), we can derive both \( c \) and \( d \), then, assuming \( c \) and \( d \) cannot both be true at the same time, we can state that an inconsistency exists. Carrying out the multiplication we produce an incidence matrix of the form:

\[
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
0 \\
1 \\
0 \\
1 \\
1
\end{pmatrix}
= 
\begin{pmatrix}
1 \\
2
\end{pmatrix}
\]
If we now consider a variable $A(i)$ which equates to the number of antecedent propositions in a rule $i$ we can define an identity:

$$
\text{If } A(i) = A(j) = I(i,j) \text{ inconsistency exists}
$$

In the above example $A(1) = A(2) = A(1,2) = 2$, thus we can say that an inconsistency exists between Rules 1 and 2.

9. RELATED WORK

There is not sufficient space in this document to give an extensive overview of alternative existing verification systems and other related work. Examples of well known systems include Suwa's ONCOCIN rule checking program (Suwa\textsuperscript{12}) CHECK (Nguyen et al.\textsuperscript{13} 1987) and EVA (Chang et al.\textsuperscript{4}) to mention a few of the more notable. A brief overview of these systems is given below, readers requiring further information are referred to the indicated references.

Suwa's rule checking program is one of the earliest examples of a KBS verification system. It checks for a number of anomalies using decision table techniques and was used successfully on the ONCOCIN system, an EMYCIN-like KBS (for oncology) that used attribute-value rules. CHECK is an automated rule verifier that is also based on ideas about decision tables. It was developed out of Suwa's rule checking program and operates on KBSs developed using the LES shell. This employs a combination of goal driven and data driven rules and allows rules to contain variables that may be instantiated at run-time. EVA (Expert systems validation Associate) was, in turn, developed out of CHECK by a research group headed by Stachowitz and Chang at the Lockhead Artificial Intelligence Centre in Austin, Texas. The long range goals of the EVA project, which started in 1986, are to build an integrated set of generic tools to "validate" any KBS written in any expert system shell. EVA is written in PROLOG and consists of a wide range of validation tools that enables the user to check for a number of anomalies including those identified here. EVA can be viewed as a metashell consisting of a unifying architecture that uses a single inference strategy, a single meta-KB and a common language for specifying requirements, constraints and
models for domain knowledge, hence making it independent of any specific shell.

Systems such as EVA are very much experimental, furthermore they are extremely complex. We maintain that verification need not be complex, and hence high cost, but both simple to implement and at the same time use.

10. CONCLUSIONS

In this paper we have described the IMVER software system. Of particular significance is that verification of KBSs, whether they be engineering systems or designed to address other domains, need not be complex. The incidence matrix technique provides us with a simple, and mathematically elegant method, of verifying rule-bases that works well on medium sized KBSs.

However a number of problems have been encountered which have yet to be addressed. Firstly the IMVER system can only test rule-bases represented using production rules; further we cannot distinguish between entities, values and variables contained in propositions. Secondly the technique requires the matrix multiplication of every rule in the rule-base with every other rule in the rule-base.

Current work is therefore concentrated on addressing these problems. Consideration is being given to developing the system so that propositions can be "unpacked". In addition a more efficient mechanism for storing and manipulating the matrix is under investigation. We are also considering addressing a number of further anomalies, particularly those resulting from chained inferencing.

REFERENCES.


