Rule-base content verification using a digraph-based modelling approach

G.S. Gursaran\textsuperscript{a,*}, S. Kanungo\textsuperscript{b}, A.K. Sinha\textsuperscript{a}

\textsuperscript{a}Department of Mathematics, Faculty of Science, Dayalbagh Educational Institute, Dayalbagh, Agra 282 005, India
\textsuperscript{b}Department of Management, Indian Institute of Technology, Hauz Khas, New Delhi, India

Received 11 November 1998; accepted 8 March 1999

Abstract

Ensuring that the content of a rule-base, which is being encoded, is free from problems of consistency, completeness, and conciseness, is necessary to avoid any performance errors that might occur during consultation sessions with the rule-based system. In this paper we have described, formally, content verification of a specific type of rule-base using a digraph-based modelling approach. Through analytic formulations it is demonstrated that problems in the rule-base lead to the existence of certain properties in the digraph and various rule-base model representations that have been devised in this work. These properties, in turn, as is also shown through an example, can be examined for rule-base content verification.

Keywords: Rule-base verification; Rule-based systems

1. Introduction

Rule-based (expert) systems (RBSs) have now been applied to assist in a variety of engineering tasks ranging from scheduling production [1], to assisting in effective product life-cycle implementations, e.g. design for manufacturing [2]. However, the quality of assistance provided by the rule-based system depends not only on the appropriateness and comprehensiveness of knowledge encoded in the rule-base, but also on the fact that the rule-base is free from errors, either domain-dependent or domain-independent. This makes verification of rule-based systems an extremely important issue.

The rule-base of a RBS is constructed through a collaborative effort of the knowledge engineer and the application-domain experts. Problems, however, can occur at any stage in the knowledge encoding process which are then reflected in the content of the rule-base, i.e. in the domain-independent structural characteristics of the rules actually stored in the rule-base [3]. Since a rule-base cannot be tested, even on simple cases, until much of it is encoded, most researchers [4–8], contend that the ability to verify content, during rule-base construction, would be an invaluable aid in ensuring that it is free from the problems of consistency\textsuperscript{1}, completeness\textsuperscript{2}, conciseness\textsuperscript{3} that usually remain dormant until performance errors occur. In fact the ability to content verify rule-base, as Nazareth [7] argues, would enhance the viability of rule-based system technology itself.

TEIRESIAS [9], represents the first attempt to automate rule-base verification. It examines the completed MYCIN [10] rule set and builds rule models against which new rules or refined rules can be verified. The Rule Checker Program of Siuva [4,11], was built to check consistency and completeness of rules in the ONCOCIN system (a rule-based system for clinical oncology). The rule checker works as the system is being built and also hypotheses missing rules and thus addresses an important aspect of completeness. CHECK [5] and ARC [12] are rule-base content verification tools that have been developed for the LES (Lockheed's rule-based expert system) and ART expert system development packages, respectively. CHECK is an extension of the rule checking program of [4]. It differs from the ONCOCIN rule checker in that it is applied to the entire set of rules for a goal and not just subsets that determine the value of each attribute. It can handle both backward
chaining and forward chaining rule forms. ARC converts all rules to forward chaining rule forms first.

UVT [6] is a unification-based tool for rule-base content verification. Apart from the usual checks, it checks for problems caused by inferred rules by computing the transitive closure of the rule-base first. The method does not assume any control strategy on the part of the rule-based system. Nazareth [7] shows how to model a rule-base, used with forward chaining, with petri-nets. Rigorous proofs certifying content verification capabilities have been presented by him. PREPARE [8] is an automated tool for detecting potential errors in a rule-base. PREPARE models the rule-base using a Predicate/Transition net (assumes rule elements to be first-order logic based predicates). Inconsistent, redundant, subsumed, circular and incomplete rules are detected through a syntactic pattern recognition method.

Nazareth [7] notes that most methods for content verification of rule-bases can be placed in the following three categories: (i) pairwise comparison methods (e.g. Refs. [5,6]), (ii) decision context based methods (e.g. Refs. [11,13]), and (iii) model based methods (e.g. Refs. [7,8]). Numerous problems associated with the methods in the first two categories make their application difficult [7]. However, the alternative model based methods hold much promise. Methods in this category usually transform the rule-base into a model in another format. For example, rule-bases may be modelled as petri-nets [7], or as inference nets [12]. Modelling is usually based on the structural connectivity of rule components and the verification process is reduced to specific within the model. One definite advantage, though not exploited by most researchers, is that the applicability of a model based method to content verification can be formally proved.

Since most of the available methods do not present formal mathematical proofs certifying applicability, we in this paper have used a digraph-based modelling approach to describe a formal content verification of a specific type of rule-base. Through analytic formulations it is demonstrated that problems in the rule-base lead to the existence of certain properties in the digraph and various rule-base model representations that have been devised in this work, which, in turn, can be examined for rule-base content verification. The work is different from Ref. [7] in that it examines the rule-base of an RBS that employs a backward chaining inference control strategy.

2. The assumed rule-based system

The essential characteristics of the RBS, on which content verification is researched, is described in this section. All references to a "rule-based system" in this paper are taken to be a reference to the rule-based system described in this section unless stated otherwise.

In the rule-based system, atomic propositions are boolean valued statements and clauses are atomic propositions or the negation of atomic propositions. A rule is of the form

\[ \text{IF } c_1 \& c_2 \& \ldots \& c_n \text{ THEN } c_{n+1} \]  

and is simply written as

\[ c_1 \& c_2 \& \ldots \& c_n \rightarrow c_{n+1} \]  

where \( c_1, \ldots, c_{n+1} \) are distinct clauses with \( n \geq 1 \). In a rule, the set of clauses in the antecedent is called the premise set of the rule, while each clause in the antecedent is called a premise of the rule. Sometimes, a premise of a rule may be referred to as IF condition and the consequent as conclusion or THEN condition. The consequent is always a single clause and not a conjunction of clauses. This is a restriction. Note that in the system, the use of the disjunctive form is explicitly disallowed. Furthermore, it is assumed that rules are strictly deterministic in nature and do not involve any probabilities.

The finite collection of all rules that constitute the rule-base of the system is called rule-collection which is also denoted \( \mathcal{R} \). Furthermore, in the system, inferencing is strictly through backward chaining. The clauses are thus classified as:

Fact clauses (or external clauses)

These are clauses whose truth values are determined by querying the user or from a database of facts known to the system. Such clauses do not constitute the consequent of any rule.

Goal clauses

These are clauses that match system goals. Such clauses do not appear as a premise in the antecedent of any rule in the rule-base. It is assumed that a system goal must always match a goal clause.

Intermediate clauses

These are the remaining clauses. They are chained to the consequent of other rules, in the rule-base, in backward chaining.

It may be noted that modelling and verification has been limited to a system without uncertainty. This is because a general verification of a rule-base can then be described. The domain knowledge is also taken to be of the propositional form. This is because knowledge in propositional form can be easily obtained and encoded. Furthermore, it ameliorates error detection. These views are not any different from those in Ref. [7].

3. Basic modelling concepts

In this section we define the basic concepts necessary for modelling the rule-base, i.e. an element set \( \mathcal{E} \), called clauseset, and the modelling relation \( R_{pc} \), called pc-relation.

Definition 1. Rule-collection \( \mathcal{R} = \{r_1, r_2, \ldots, r_t\} \) is an ordered collection of rules where \( t \) is the number of rules in rule-collection. The ordering of rules in \( \mathcal{R} \), denoted rule-order, is defined by the sequence in which rules are physically
stored in the rule-base of the rule-based system. For rule \( r_k \), \( k \in \{1, \ldots, r\} \), in rule-collection \( \mathfrak{R} \), the premise set of \( r_k \) is denoted \( P_k \) and the consequent is denoted \( C_k \) which is a single clause.

**Definition 2.** The set \( \mathcal{E} = \{c_1, c_2, \ldots, c_s\} \), called clauseset, is the set of clauses such that for each clause \( c \in \mathcal{E} \) there exists at least one rule in rule-collection \( \mathfrak{R} \) in which clause \( c \) is either a premise or the consequent.

Definitions 1 and 2 together imply that
\[
\mathcal{E} = \bigcup_{k=1}^{r} (P_k \cup \{C_k\}).
\] (3)

**Definition 3.** The modelling relation \( R_{pc} \) called pc-relation, is a binary relation in clausesset \( \mathcal{E} \) such that for clauses \( c, c' \in \mathcal{E} \), \( (c, c') \in R_{pc} \) if and only if clause \( c \) is a premise in a rule in rule-collection \( \mathfrak{R} \) which has clause \( c' \) as its consequent.

In the case of a content verified rule-collection \( \mathfrak{R} \), \( R_{pc} \) is clearly irreflexive\(^4\) and asymmetric\(^5\).

**Definition 4.** A finite sequence of distinct clauses
\[x_1, x_2, \ldots, x_k, x_{k+1} \in \mathcal{E}, \quad 1 < k \leq s, \quad \text{such that} \quad x_1 R_{pc} x_2, x_2 R_{pc} x_3, \ldots, x_k R_{pc} x_{k+1}, \]
called a complete clause chain. A finite sequence of clauses
\[x_1, x_2, \ldots, x_k \in \mathcal{E}, \quad 1 < k \leq s, \quad \text{such that} \quad x_1 R_{pc} x_2, x_2 R_{pc} x_3, \ldots, x_k R_{pc} x_k, \]
called a circular clause chain if \( x_1 = x_k \) and clauses \( x_2, x_3, \ldots, x_{k-1} \) are all distinct.

**Definition 5** [14]. In a partially ordered set \((B, P)\), the reachability set \( R(x) \) of an element \( x \in B \) is defined as \( R(x) = \{y \in B \mid x P y\} \) and the antecedent set, \( A(x) \), of an element \( x \in B \) is defined as \( A(x) = \{y \in B \mid y P x\} \).

In Definition 5, as \( P \) is reflexive, \( x \in R(x) \). This, together with antisymmetry of \( P \) ensures that reachability sets of no two distinct elements in \( B \) are equal, although the reachability set of one element may be a proper subset of the other. Thus, for all \( x, y \in B \), \( x \neq y \), \( R(x) \neq R(y) \). Also, for all \( x, y \in B \), \( x \neq y \), \( A(x) \neq A(y) \).

With respect to \( P \), above, set \( B \) can be partitioned into a level hierarchy such that all the elements into a given level are contained in the same block\(^6\) [14].

4 Any rule \( r_k \) in which clause \( C_k \) is an \( R_{pc} \), leading to an infinite inference loop when backward chaining is attempted with it. Also follows from the definition of the rule which is given in Section 2.

5 For clauses \( c, c' \in \mathcal{E} \), if \( (c, c') \in R_{pc} \), then \( (c', c) \notin R_{pc} \). This also follows from the arguments presented for irreducibility.

6 If \( 0 \) is the level which is the empty set, \( L_0 = \emptyset \), the iterative level identification algorithm may be written as [14]
\[L_0 = \{c \in B \mid c \in -L_{k-1} \}, R_{pc}(-L_{k-1} \cap c) \notin A(c)\},
\] where \( R_{pc}(-L_{k-1}) \) and \( A(c) \) are the reachability and antecedent sets determined for the elements in \( B \). Let \( \leq \) be a strict partial order relation corresponding to the partial order relation \( \leq \). If \( \theta \) is the transitive reduction of \( \leq \) then for every relation \( \sigma \) such that the transitive closure \( \sigma' \) is \( \leq \), we have \( \theta \subseteq \sigma \). In other words, the transitive reduction is the least relation whose transitive closure is \( \leq \) [16].

4. The modelling and verification steps

Content verification of a rule-base is integrated into the framework of modelling and model property identification. The basic steps in modelling and verification are:

1. Construction of rule matrix \( N = [n_{ij}] \), redundancy set (RS) and conflict set (CS).

   The rule matrix \( N \) is a matrix of order \( |\mathcal{E}| \), where \(|\mathcal{E}|\) denotes cardinality of clauseset \( \mathcal{E} \). The \( i \)th row and \( j \)th column in \( N \) correspond to clause \( c_i \) in \( \mathcal{E} \). The matrix entries are made by reading rules from rule-collection \( \mathfrak{R} \) in rule-order. The exact process is outlined in Procedure 1 below.

   The redundancy set \( RS \) is a set of 3-tuples \((x,y,z)\) in \( R \), a clause that is in both the premise sets \( P_x \) and \( P_y \) of rules \( r_x \) and \( r_y \), respectively. Furthermore, in the 3-tuple \((x,y,z)\), \( x < y \), i.e., rule \( r_x \) precedes rule \( r_y \) in rule-order.

   The conflict set \( CS \) is a set of ordered pairs. In an ordered pair \((x,y)\) in \( CS \), \( x \) is a clause whose negation is also in \( \mathcal{E} \) and \( x = y \) is a rule index which represents the position of a rule in rule-order and that is determined as described in Procedure 1 below.

   The following procedure is adopted to make entries in \( N \) and construct \( RS \) and \( CS \). Statements enclosed within "*" denote comments.

   It can be noticed in Procedure 1 that steps 3 through 14 are repeated for every rule, read in rule-order, in rule-collection \( \mathfrak{R} \). Steps 7 through 10 ensure that for every pair of rules \( r_i, r_j \in \mathfrak{R} \), where \( i < j \), with the same consequent clause and a common clause, say \( c \), in their premise sets, the 3-tuple \((i,j,c)\) is included in the redundancy set \( RS \). Furthermore, it follows from step 7 and step 9 that Procedure 1 does not store any 3-tuple in \( RS \) that violates
Procedure 1.

1. BEGIN
2. Initialise \( N = 0 \), \( RS = \emptyset \), and \( CS = \emptyset \);
3. FOR \( k = 1, \ldots, t \) DO BEGIN
4. Read the \( k \)th rule \( r_k \) in rule-order, from rule-collection \( \mathcal{R} \);
5. FOR every clause \( c_i \in P_k, C_k = c_i \) do BEGIN
6. /* make an entry in cell \( n_{kq} \) in rule matrix \( N \) as follows: */
7. IF \( n_{kq} = 0 \) THEN assign \( k \) to \( n_{kq} \);
8. IF \( n_{kq} = q, q < k \) THEN BEGIN
9. /* leave \( n_{kq} \) unchanged; */
10. include the 3-tuple \((q,k,c_i)\) in \( RS \);
11. FOR each 3-tuple \((p,q,c_i)\), where \( p < k \), in redundancy set \( RS \), DO
12. include the 3-tuple \((p,k,c_i)\) also in \( RS \);
13. END; /* of IF \( n_{kq} = q \) */
14. IF clause \( c_j \) is the negation of clause \( c_i \) THEN
15. include the ordered pair \((c_i,k)\) in the conflict set \( CS \);
16. END; /* for every clause in premise set \( P_k \) */
17. FOR every clause \( c \in C \),
18. IF the negation, \( \neg c \), of clause \( c \) also belongs to clauseset \( \mathcal{C} \) THEN
19. IF \((c,x) \notin CS \) and \((\neg x,y) \notin CS \), where \( 0 \leq x,y \leq t \), THEN
20. include the ordered pair \((c,0)\) in \( CS \);
21. END. /* Procedure 1. */

the definition of \( RS \). Fig. 1 illustrates the rule matrix \( N \), redundancy set \( RS \), and conflict set \( CS \) for an example.

The worst-case time complexity is clearly decided by the loop 3–15 with all \( t \) rules, in rule collection \( \mathcal{R} \) having the same premise set, of say \( n - 1 \) clauses, and the same conclusion, i.e. all the \( t \) rules are redundant. Let us assume that the redundancy set is indexed by the rule number pair \((i,j)\) in 3-tuple \((i,j,c)\) and that clauses are stored in linear lists in the indexed positions with insertions taking place in front of the list. For the first rule there are \( n - 1 \) entries to be made in the rule matrix. For the second rule, there are \( n - 1 \) insertions made in the clause list at index position \((1,2)\) in the redundancy set. For the third rule, in the redundancy set, there are a total of \( O(n^2) \) comparisons made at index position \((1,2)\) along with the insertions at index positions \((1,3)\) and \((2,3)\) which is \( O(1) \) time for each insertion. For the fourth rule, in the redundancy set, there are \( O(n^2) \) comparisons made at index position \((1,2)\) followed by \( O(n^2) \) comparisons made at index position \((1,3)\) along with the insertions at index positions \((1,4)\), \((2,4)\) and \((3,4)\), i.e. the time taken is \( 2O(n^2) \). For the fifth rule it is \( 3O(n^2) \) and so on. Since there are \( t - 1 \) rules for which insertions are made in the redundancy set, the time complexity for the loop in lines 3–15 is clearly \( O(n^2 t) \).

2. Construction of an adjacency matrix \( A = [a_{ij}] \). The entries in the adjacency matrix \( A \) are derived directly from \( N \) as follows:

\[
a_{ij} = \begin{cases} 
1 & \text{if } n_{ij} \neq 0 \\
0 & \text{if } n_{ij} = 0
\end{cases}.
\]  

(6)

The adjacency matrix represents the pc-relation \( R_{pc} \).

3. Development of a reachability matrix \( X \) which represents the transitive–reflexive closure, \( R^*_{pc} \), of pc-relation \( R_{pc} \) in clauseset \( \mathcal{C} \).

4. Identification of various partitions induced by the reachability matrix on the set and subsets of clauseset \( \mathcal{C} = \{ c_i \} \). These partitions are: (i) the relation partition; (ii) the level partition; (iii) the separate parts partition; (iv) the disjoint and the strong subsets partition of each
level identified in (ii); and (v) the strongly connected subsets partition as per identification in (iv). They are described in detail in Section 6. Identification of each partition is followed by verification.

5. Construction of the structural model of $R_{p,c}^{*}$ in clauseset $\mathcal{C}$. This may be accomplished by first constructing a minimum edge adjacency matrix $M = [m_{ij}]$ that represents the transitive reduction of $R_{p,c}^{*}$ in clauseset $\mathcal{C}$, and then using the connectivity information in the minimum edge adjacency matrix to derive the minimum edge digraph structural model. Construction is followed by a correction procedure, which is described in Section 6.5, and then by verification.

5. Problems in content of rule-collection $\mathcal{R}$

The subsequent subsections describe how consistency, completeness, and conciseness may be compromised in rule-collection $\mathcal{R}$. The description is strictly within the framework of the chosen rule-based system.

5.1. Consistency

Consistency in rule-collection $\mathcal{R}$ is compromised by circularity and conflicts.

Conflicts: A conflict may occur in the form of:

Self-Contradictory Rule
A rule in rule-collection $\mathcal{R}$ is a self-contradictory rule if its consequent clause is the negation of a premise in its antecedent.

Directly Contradictory Rules
Two rules in rule-collection $\mathcal{R}$ are directly contradictory if they have the same antecedent and the consequent clause of one rule is a negation of the consequent clause of the other.

Contradictory Clause Chain
A clause chain $x_{1}, x_{2}, ... , x_{k} \in \mathcal{C}, k > 2$, is a contradictory clause chain if clause $x_{k}$ is the negation of clause $x_{1}$.

Circularity: Circularity may occur in the form of:

Self-Referent Rule
A rule in rule-collection $\mathcal{R}$ is a self-referent rule if its consequent clause is also a premise in its antecedent.

Circular Clause Chain
See Definition 4 in Section 3.

Circularity is undesirable for the simple reason that it moves a rule-based system into an infinite inference loop when backward chaining is attempted.

5.2. Completeness

Compromises in completeness refer to knowledge gaps in the rule-collection. Such gaps usually lead to incomplete chains of inference thereby leading to inconclusive inference sessions. The situations that are indicative of knowledge gaps are as follows:

Unreachable Conclusion: In backward chaining, the consequent clause of a rule in rule-collection $\mathcal{R}$ should either match a system goal or match a premise in the antecedent of another rule in rule-collection $\mathcal{R}$. If there are no matches for the consequent clause, it is said to be an unreachable conclusion.

Dead End Goal: In order to achieve a system goal (or a sub-goal) in the backward chaining system, it is necessary that either the truth of the goal be determined by querying the user or it should match the consequent clause of some rule in rule-collection $\mathcal{R}$. If neither is achievable, then the goal is said to be a dead end goal.

Dead End IF Conditions: For a rule premise in the antecedent of a rule in rule-collection $\mathcal{R}$, it is required that either.

- the truth of the premise be determined by querying the user or through a database of facts, or
- the premise should match the consequent clause of some other rule in rule-collection $\mathcal{R}$.

If neither is achievable then the premise is said to be a dead end IF condition.

Missing Rules: There are missing rules, if a situation exists in which a particular inference is required but there is no rule in rule-collection $\mathcal{R}$ to produce the desired inference, i.e. there are knowledge gaps in rule-collection $\mathcal{R}$.

As clauses have been assumed to be boolean valued, the problems of unreferenced attribute values, and illegal attribute values (e.g. as in Refs. [5,12]) are not considered. These are specific to first-order predicate logic-based rule representations.

In the knowledge engineering process a large number of rules may be overlooked by the expert or by the knowledge engineer. Rules that have been overlooked are also said to be missing rules. As Sunu et al. [4] note “missing rules can be detected if it is possible to enumerate all circumstances in which a given decision should be made or a given action should be taken.” However, the problem of detecting missing rules, in general, is known to be hard and it is for this reason that most verification methods have avoided addressing it [7].

5.3. Conciseness

Conciseness deals with situations in which rules are present in rule-collection $\mathcal{R}$ that logically serve no purpose. Compromises in conciseness may not really pose problems in rule-based systems, in general, in which rules are deterministic in nature and only the first applicable rule is guaranteed to succeed. However, if rules involve probabilities, conciseness errors might pose considerable problems [17]. For example, if two rules have the same antecedent, but rule probabilities that are different, then they may cause the same
evidence (factual or conclusive) to be counted twice [4,17].

Conciseness is compromised in the following situations:

Redundant Rules
Two rules in rule-collection \( \mathcal{R} \) are said to be redundant rules if they have the same antecedent and the same consequent, i.e., there are two instances of the same rule.

Subsumed Rules
A rule \( r_i \) is subsumed by another rule \( r_j \) in rule-collection \( \mathcal{R} \) if both the rules have the same consequent and the premise set of rule \( r_j \) is a proper subset of the premise set of rule \( r_i \).

Unnecessary IF Conditions
Two rules in rule-collection \( \mathcal{R} \) contain unnecessary IF conditions if the rules have the same consequent, a premise in one rule is in conflict with a premise in the other rule, and all the other premises in the two rules are the same. The two rules in fact should be replaced by a single rule that does not contain either of the conflicting premises. A special case occurs when two rules have the same consequent, one rule has a single premise in its antecedent that is also in conflict with a premise in the other rule which in turn has two or more premises in its antecedent. For example, if \( A \rightarrow B \) and \( C \& D \rightarrow B \) are two rules in rule-collection \( \mathcal{R} \) with \( A \equiv C \), then clause \( C \) is an unnecessary premise. But, both the rules \( A \rightarrow B \) and \( D \rightarrow B \) are still required in rule-collection \( \mathcal{R} \).

5.4. Rule-base content verification: a case study

Content verification is illustrated with a small example rule-base. The example has been kept small so as to be able to display the matrices generated through the modeling process. The rules in the rule-base, called ER-BASE, are

\[
\begin{align*}
1: & \quad A \rightarrow B \\
2: & \quad A \& C \rightarrow B \\
3: & \quad B \rightarrow D \\
4: & \quad E \rightarrow F \\
5: & \quad F \rightarrow D \\
6: & \quad H \rightarrow I \\
7: & \quad J \rightarrow K \\
8: & \quad L \rightarrow M \\
9: & \quad M \rightarrow N \\
10: & \quad N \rightarrow L
\end{align*}
\]

Let \( A, C, E, H, L \) be fact clauses, and let \( D, K, N \) be goal clauses. Further, let \( A \equiv D \), i.e., they are contradictory clauses. The following errors are then present in ER-BASE:

1. Rule 1 subsumes rule 2.
2. The backward chaining of rules 1 and 3 gives rise to a contradictory clause chain.
3. The backward chaining of rules 8, 9 and 10 gives rise to a circular clause chain.

\[
\begin{align*}
\text{(a) Adjacency Matrix} & \quad A & B & C & D & E & F & H & I & J & K & L & M & N \\
A & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
D & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
E & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
H & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
J & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
K & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
M & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
N & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{(b) Reachability Matrix} & \quad A & B & C & D & E & F & H & I & J & K & L & M & N \\
A & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
B & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
C & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
D & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
E & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
H & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
I & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
J & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
K & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
M & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
N & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{align*}
\]

Fig. 3. ER-BASE matrices.
4. Clause 1 in rule 6 represents an unreachable conclusion.

5. In rule 7, the goal corresponding to goal clause K represents a dead end goal and clause J is a dead end IF condition.

In the modelling of ER-BASE,

\[ \mathcal{R} = \{ \text{rule 1, rule 2, rule 3, rule 4, rule 5, rule 6, rule 7, rule 8, rule 9, rule 10} \} \]

and

\[ \mathcal{C} = \{ A, B, C, D, E, F, H, I, J, K, L, M, N \} \]  

(8)

Fig. 2 shows the rule matrix \( X \) for ER-BASE. The adjacency matrix derived from \( X \) is shown in Fig. 3, along with the reachability matrix. Fig. 4 illustrates the interpretive structural model of ER-BASE. Content verification of ER-BASE is discussed in the subsequent sections.

6. Partitions and content verification

This section describes the content verification of rule-collection \( \mathcal{R} \) through the partitions induced by the reachability matrix \( X \) on the set and subsets of clauseset \( \mathcal{C} = \{ c \} \).

6.1. The level partition

Clauseset \( \mathcal{C} \) is partitioned into a level hierarchy such that all the clauses at a given level are contained in the same block. For each clause \( c \), the reachability set \( R(c) = \{ c \in \mathcal{C} \mid c \rightarrow^* R \} \), and the antecedent set \( A(c) = \{ c \in \mathcal{C} \mid c \rightarrow^* R \} \).

A clause is a top level clause if and only if \( R(c) \cap A(c) \). The level partition is written as

\[ \pi_l(\mathcal{C}) = \{ L_1; L_2; \ldots; L_l \} \]  

(9)

where \( l \) is the number of levels, \( L_1 \) is the top level block, and \( L_l \) is the lowest level block.

Lemma 1. Let \( c \) be a top level clause. \( R(c) \cap A(c) = R(c) = \{ c \} \) if and only if there is no circular clause chain \( x_1, x_2, \ldots, x_k = x_1 \), \( k > 2 \), in any backward chaining of rules in rule-collection \( \mathcal{R} \).

Proof. Let us assume that \( R(c) \cap A(c) = R(c) = \{ c \} \) for \( c \) a top level clause. Now let us suppose there exists a circular clause chain \( x_1, x_2, \ldots, x_k = x_1 \), \( k > 2 \). From the transitivity of \( R^* \), it follows that \( R(c) \cap A(c) \supseteq \{ c, x_1, x_2, \ldots, x_k \} \). This contradicts our assumption that \( R(c) \cap A(c) = R(c) = \{ c \} \). Conversely, let us assume that there is no circular clause chain \( x_1, x_2, \ldots, x_k = x_1 \), \( k > 2 \), in any backward chaining of rules in \( \mathcal{R} \). Then for top level clause \( c \), it follows from the definition of \( R(c) \) that \( R(c) = \{ c \} \). Also from the definition of \( A(c) \) it follows that \( c \in A(c) \). Hence, \( R(c) \cap A(c) = R(c) = \{ c \} \). This proves the lemma. ■

Lemma 1 shows that the level partition does not give a true picture of the level distribution of clauses in the presence of circular clause chains. Hence any backward chaining of rules that leads to the creation of a circular clause chain must be corrected to eliminate this circularity problem.

In the level partition it is always the case that the number of levels \( l > 1 \), as \( l = 1 \) implies one or more of the following: (a) the rule-base is nothing but a collection of clauses and no rules which then violates our assumption that rule-collection \( \mathcal{R} \) is a finite nonempty collection of rules only; (b) all the clauses in \( \mathcal{C} \) form a clause chain (or may be a set of disjoint clause chains) that is also circular (this follows from Lemma 1); (c) all rules in rule-collection \( \mathcal{R} \) are self-referent rules and each rule has a single clause in its antecedent and that this clause is also its consequent.

The detection of circular clause chains is described in Section 6.3. It is, however, hereafter, assumed for the rest of this paper that \( \mathcal{R} \) is free of all circularity problems\(^9\) (see Section 5.1). This assumption is also referred to as no-circular-cause-chain assumption.

Set \( B \) is the set of bottom level clauses. A clause \( c \in \mathcal{C} \) belongs to set \( B \) if and only if \( A(c) = R(c) \cap A(c) \). In the level partitioning of clauses \( L_i \subseteq B \).

Lemma 2. in a content verified rule-collection \( \mathcal{R} \), if clause \( c \in \mathcal{C} \) is a fact clause then \( c \in B \).

Proof. Clause \( c \) is a fact clause implies that there exists at least one rule \( r_j \in \mathcal{R} \) such that \( c \in P_j \). Furthermore, there exists no rule \( r_k \in \mathcal{R} \) with consequent \( C_k = c \). This implies that clause \( c \), in rule \( r_j \), cannot be chained to the consequent of any rule in \( \mathcal{R} \) in backward chaining. Thus \( A(c) = \{ c \} \), and \( A(c) \cap R(c) = A(c) \). ■

From Lemma 2 we know that clauses in \( B \) should all be fact clauses. This set of clauses can be checked against the actual set of facts known to the system. If the truth of a particular clause at this level cannot be determined by querying the user or through a database of facts then the clause represents a dead end IF condition. This clause should have matched the consequent clause of some rule

\(^9\) It may be noted that this assumption also implies that there are no self-referent rules in rule-collection \( \mathcal{R} \).
in rule-collection \( \mathcal{R} \) which it cannot being in the bottom level set \( B \). Formally,

**Corollary 1 (of Lemma 2).** If clause \( c \in \mathcal{C} \) is a dead end IF condition then \( c \in B \).

**Proof.** Since clause \( c \) is a dead end IF condition there does not exist a rule \( r_j \in \mathcal{R} \) such that \( C_j = c \). Thus \( A(c) = \{ c \} \subseteq R(c) \) and hence \( c \in B \).

Similarly,

**Lemma 3.** In a content verified rule-collection \( \mathcal{R} \), if clause \( c \in \mathcal{C} \) is a goal clause then \( c \in L_1 \).

**Proof.** Clause \( c \) is a goal clause implies that there exists no rule \( r_k \in \mathcal{R} \) such that \( c \in P_k \), and there exists at least one rule \( r_j \in \mathcal{R} \), such that \( C_j = c \). Thus \( R(c) = \{ c \} \subseteq A(c) \) and hence \( c \in L_1 \).

**Corollary 2 (of Lemma 3).** If a system goal \( g \) does not match any clause in \( L_1 \), then \( g \) is a dead end goal.

Lemmas 2 and 3 suggest that a fact or goal clause should belong its appropriate level partition block. If this is not the case, i.e., a known goal clause does not belong to \( L_1 \) or a known fact clause does not belong to \( B \), then the system knowledge is not correct. This follows as rule-collection \( \mathcal{R} \) includes rules that specify these clauses as intermediate clauses. Moreover, if there are system goals that do not match any clause in \( L_1 \) and there are known facts that do not match any clause in \( B \), then it may imply that there are missing rules. Furthermore,

**Lemma 4.** The system goal corresponding to consequent clause \( c_j \), of rule \( r_l \in \mathcal{R} \), is a dead end goal if \( r_l \) is the only rule with \( c_j \) as its consequent clause, \( c_j \in L_1 \), and there is a clause \( c_i \in P_k \) that is a dead end IF condition.

**Proof.** Since the truth of \( c_i \) cannot be determined, the truth of \( c_j \) also cannot be determined. Hence, the system goal corresponding to goal clause \( c_j \) is a dead end goal.

The top level partition block \( L_1 \) comprises goal clauses. If a clause in this partition cannot be matched to a system goal then the clause represents an unreachable conclusion. The clause should have matched a premise of some rule in rule-collection \( \mathcal{R} \) which it cannot being in the top level block \( L_1 \).

**Lemma 5.** If clause \( c \in \mathcal{C} \) is an unreachable conclusion then \( c \in L_1 \).

**Proof.** Since clause \( c \) represents an unreachable conclusion, there does not exist a rule \( r_j \in \mathcal{R} \) such that \( c \in P_j \). Thus \( R(c) = \{ c \} \) and hence \( c \in L_1 \).

Clauses in partitions other than \( L_1 \) and \( B \) are intermediate clauses.

In **ER-BASE**, level partition is

\[
\pi_1 = \{ \{ D, I, K, L, M, N \}; \{ B, F, H, J \}; \{ A, C, E \} \},
\]

where \( L_1 = \{ D, I, K, L, M, N \} \) and \( L_2 = \{ B, F, H, J \} \). Level partition block \( L_1 \) contains all the goal clauses \( D, K, \) and \( N \). Additionally it contains the clause \( I \), which does not correspond to a system goal and thus represents an unreachable conclusion. Clauses \( L \) and \( M \) are fact and intermediate clauses, respectively, but both belong to block \( L_1 \) as they are contained in the circular clause chain \( L, M, N, L \) which is formed in a backward chaining of rules 8, 9 and 10. The bottom level set \( B = \{ A, C, E, H, J \} \). The circular clause chain, as one can see, has corrupted the bottom level set also. Clause \( J \) in set \( B \) is not a fact clause and represents a dead end IF condition and hence the system goal corresponding to clause \( K \) is a dead end goal.

It is obvious now that the level partition has an important role to play, not only in the verification of the rule-base but in also organizing clauses for effective visual presentation.

6.2 The separate parts partition

The separate parts partition identifies the set of clauses in clause set \( C \) that constitute a smaller digraph and which is separate. The partitioning begins with the identification of the bottom level set \( B \). Any two clauses \( c_i, c_j \in B \) are in the same block if and only if \( R(c_i) \cap R(c_j) \neq \emptyset \). The remaining clauses of the reachability sets for each block are then appended to the block. Thus, the separate parts partition is

\[
\pi_1(\mathcal{C}) = \{ D_1; D_2; \ldots; D_d \}
\]

where \( d \) is the number of disjoint digraphs.

The separate parts partition delineates blocks of rules such that within each block rules are intertwined. Based on this partitioning the knowledge engineer can suggest meta-rules that can help the system identify a block of rules first and then begin its reasoning task. This will help reduce search, especially in comparison to an exhaustive goal hypothesising scheme. If the partition is not correct and rules in two or more blocks are in fact intertwined then it implies that there are missing rules. If the partitions are correct then they can also be used to create smaller reachability matrices for the purpose of further content verification.

In **ER-BASE** the separate parts partition is \( \pi_1 = \{ \{ A, B, C, D, E, F \}; \{ I, H \}; \{ L, K, J \}; \{ L, M, N \} \} \). Using this partition, the rule collections \{ rule 1, rule 2, rule 3, rule 4, rule 5 \}, \{ rule 6 \}, \{ rule 7 \}, and \{ rule 8, rule 9, rule 10 \} can be delineated.

Since it has been assumed that no self-referent rules exist in rule-collection \( \mathcal{R} \), no partition block in the separate parts partition can be a singleton set. The conditions that may lead to the existence of a singleton block are formally characterised as follows.
Lemma 6. If in \( \pi_6(\emptyset) \) there exists a singleton block \( D = \{ c \} \), for some clause \( c \in \emptyset \), then there exists a self-referent rule in rule-collection \( \mathfrak{R} \) with its consequent clause as clause \( c \) and with a premise set that is the singleton \( \{ c \} \).

**Proof.** The existence of a singleton block \( D = \{ c \} \) implies that \( R(c) = A(c) = \{ c \} \). Since only rules belong to rule-collection \( \mathfrak{R} \), as per the rule-base structure assumed in Section 2, and since each rule in rule-collection \( \mathfrak{R} \) at least one premise in its antecedent, this implies that there exists a rule in rule-collection \( \mathfrak{R} \) with its consequent clause as clause \( c \) and with a premise set that is the singleton \( \{ c \} \). In other words there exists a self-referent rule with a single premise in its antecedent, which is clause \( c \), and with a consequent clause which also is clause \( c \).

In Lemma 6, if clause \( c \) is a goal clause then the truth of clause \( c \) cannot be determined and the system goal corresponding to clause \( c \) is a dead end goal.

6.3. The disjoint and strongly connected subsets partition

The reachability matrix induces a two block partition \( \pi_4(L_k) \) on each level partition block, \( L_k \), depending on whether a clause belongs to or does not belong to a strongly connected subset. If a clause \( c \) is not a part of a strongly connected set then \( RL(c) = \{ c \} \) where \( RL(c) \) denotes reachability of \( c \) with respect to the elements of \( L_k \). The two block partition is denoted by

\[
\pi_4(L_k) = [I; S].
\]

(11)

A clause \( c \in L_k \) is contained in block \( I \) if it satisfies the equation \( RL(c) = \{ c \} \), otherwise it is contained in block \( S \).

The reachability matrix also induces a partition \( \pi_4(S) \) such that a group of clauses are in the same block if every clause in the group is reachable from and antecedent to every other clause in the group. Thus

\[
\pi_4(S) = \{ MC_1; MC_2; \ldots; MC_y \}
\]

(12)

where \( MC \) denotes a maximal cycle set and \( y \) is the number of maximal cycle sets.

Given the no-circular-clause-chain assumption, the \( S \) block at each level should be empty. Furthermore.

**Corollary 3 (of Lemma 1).** Let \( x_1, x_2, \ldots, x_k = x_1, k > 2 \), be a circular clause chain then there exists a maximal cycle set \( MC \) at some level such that \( MC \supseteq \{ x_1, x_2, \ldots, x_{k-1} \} \).

**Proof.** The circular clause chain \( x_1, x_2, \ldots, x_k = x_1, k > 2 \), implies that \( R(x_1) \cap \ldots \cap R(x_{k-1}) \cap A \) \( \implies (x_1) \cap \ldots \cap A \) \( \{ x_{k-1} \} \supseteq \{ x_1, x_2, \ldots, x_{k-1} \} \). Thus the set \( \{ x_1, x_2, \ldots, x_{k-1} \} \) is a subset of a maximal cycle set at some level.

The knowledge engineer can, and should, assure that the no-circular-clause-chain assumption is satisfied. This is possible with the help of the \( \pi_4 \) and \( \pi_5 \) partitions at each level with which the knowledge engineer can detect those circular clause chains that do not result from self-referent rules.

Self-referent rules can be detected in rule matrix \( \mathfrak{R} \) and the redundancy set \( RS \) as follows. Let rule \( r_i \in \mathfrak{R} \) be self-referent in clause \( c \in \emptyset \). Let \( r_i \) be the first rule in rule-order that is self-referent in clause \( c \). It follows from Procedure 1 that \( r_1 \) has \( k \) rule matrix \( \mathfrak{R} \), and, furthermore, for each rule \( r_i \) that is redundant with \( r_j \) in clause \( c, q > k \), the 3-tuple \( (k, q, c) \) exists in redundancy set \( RS \).

In \( \text{ER-BASE} \), \( \pi_5(L_k) = \{ D, I, K; \{ L, M, N \} \} \), \( \pi_4(L_2) = \{ B, F, H, J; \emptyset \} \), and \( \pi_5(L_5) = \{ A, C, E; \emptyset \} \).

It can be seen that \( \pi_5(L_1) \) indicates the presence of a circular clause chain which is corroborated by the \( \pi_3 \) partition of its \( S \) block. Partition \( \pi_3(L, M, N) = \{ L, M, N \} \). In a backward chaining of rules 8, 9, 10, the clauses \( L \), \( M \), and \( N \) form a circular clause chain \( LMNL \) which leads to the maximal cycle set \( \{ LMN \} \). In Fig. 4, it can be seen that all the three clauses \( L \), \( M \), and \( N \) appear at level 1 thereby giving an incorrect picture of the level distribution of clauses.

6.4. The relation partition

The reachability matrix induces a partition on the ordered pairs of \( \emptyset \times \emptyset \) into two blocks \( Z \) and \( Z' \). An ordered pair \( (c_i, c_j) \) belongs to \( Z \) if \( c_i, R^*, c_j \), else \( (c_i, c_j) \) belongs to \( Z' \). The relation partition is written as

\[
\pi_5(\emptyset \times \emptyset) = [Z; Z'].
\]

(13)

The relation partition together with the level partition can help in hypothesising missing rules in rule-collection \( \mathfrak{R} \).

An attempt at hypothesising missing rules was made by Suwa et al. [4] in their rule checker program for the ONCOCIN system—a rule-base system for clinical oncology. The rule checker supposes that there should be a rule for every combination of values of the condition parameters of the premises of a rule. However, this supposition results in the rule checker program hypothesising rules that at times have semantically impossible combinations of condition parameter values. But as Suwa et al. [4] note, the method was extremely helpful in helping them debug the developing knowledge-base.

Since in our system, clauses are boolean valued propositions or their negations, the method of enumerating rules for missing condition parameter value combinations does not make sense. However, if it is assumed that there exists a requirement of hypothesising rules for every 2- combination of clauses in clause set \( \emptyset \) and that, at the same time, do not introduce any of the problems described in Section 5, then such rules can be hypothesised, to some extent, with the help of the relation and the level partition.

Initially an ordered pair in \( Z' \) is taken as a candidate for
hypothesising a rule for inclusion in rule-collection $\mathcal{R}$. It is then verified that the rule hypothesised from the ordered pair does not fall in any of the problem categories described below. On successful verification the rule is considered to be a candidate for inclusion in rule-collection $\mathcal{R}$. The final inclusion of the hypothesised rule in $\mathcal{R}$ is, however, contingent to its acceptance by the expert.

Let us assume, for this section, that rule-collection $\mathcal{R}$ is completely content verified, i.e. $\mathcal{R}$ has none of the problems mentioned in Section 5. For each ordered pair $(c_i, c_j) \in Z'$, the rule $c_i \rightarrow c_j$ is hypothesised and considered as a candidate for inclusion in $\mathcal{R}$ if $(c_i, c_j)$ does not fall in any of the following categories:

1. **Circular Rule Inconsistency Category:** The ordered pair $(c_i, c_j)$ belongs to this category
   - if $(c_j, c_i) \in Z$. The inclusion of the hypothesised rule $c_i \rightarrow c_j$ leads to the creation of the circular clause chain $c_i, \ldots, c_j, c_i$ in a backward chaining of rules in rule-collection $\mathcal{R}$, or else
   - if both $(c_i, c_j)$ and $(c_j, c_i)$ belong to $Z'$ and $(c_i, c_j)$ is more likely to be accepted as a valid rule. Since for every $c \in \mathcal{R}$, the ordered pair $(c, c)$ belong to the Z block, self-referential rules are not hypothesised.

2. **Level Inconsistency Category:** The ordered pair $(c_i, c_j)$ belongs to this category
   - if both $c_i, c_j \in B$ or both $c_i, c_j \in L_1$, i.e. both the clauses are either fact clauses or goal clauses, respectively. In rule-collection $\mathcal{R}$ a fact clause cannot be the consequent of any rule and a goal clause cannot be a premise in the antecedent of any rule. Moreover, if both $c_i$ and $c_j$ belong to $B$ and the hypothesised rule $c_i \rightarrow c_j$ is acceptable then $c_i$ is actually a dead end IF condition and not a fact clause. This contradicts our assumption that rule-collection $\mathcal{R}$ is completely content verified. Furthermore, if both $c_i$ and $c_j$ belong to $L_1$ and hypothesised rule $c_i \rightarrow c_j$ is acceptable then $c_i$ is actually an unreachable conclusion and not a goal clause. This again contradicts our assumption that rule-collection $\mathcal{R}$ is completely content verified, or else
   - if $c_i \in B$ and $c_j \in (\mathcal{R} - L_1 - B)$, i.e. clause $c_i$ is a fact clause and clause $c_j$ is an intermediate clause. This is because a fact clause would then be the consequent in the hypothesised rule, implying that it was actually a dead end IF condition. Furthermore, as mentioned above, clause $c_i$ cannot be a goal clause (i.e. in $L_1$) as it would then imply that it was actually an unreachable conclusion, or else
   - if $c_i \in L_1$ and $c_j \in (\mathcal{R} - L_1 - B)$, i.e. $c_j$ is a goal clause and $c_j$ is any intermediate clause. This is because a goal clause would then be a premise in the hypothesised rule implying that it was actually an unreachable conclusion. Furthermore, as mentioned above, $c_j$ cannot be a fact clause (i.e. in $B$) as it would then imply that it was an dead end IF condition.

3. **Contradictory Rules Inconsistency Category:** The ordered pair $(c_i, c_j)$ belongs to this category
   - if clause $c_i$ is the negation of clause $c_j$. The rule $c_i \rightarrow c_j$ then implies a self-contradictory rule, or else
   - if the inclusion of the hypothesised rule, $c_i \rightarrow c_j$, leads to the creation of a clause chain in which either clause $c_i$ or clause $c_j$ is a negation of some other clause in the clause chain.

In both the cases backward chaining leads to contradictory conclusions.

4. **Rule Premise Redundancy Category:** The ordered pair $(c_i, c_j)$ belongs to this category, if clauses $c_i$ and $c_j$ are premises in the same rule in rule-collection $\mathcal{R}$. For example, if rule $A \rightarrow B$ is hypothesised and rule-collection $\mathcal{R}$ contains the rule $A \& B \rightarrow C$, then the two rules tautologically imply the rule $A \rightarrow C$. Clause $B$ is thus a redundant premise (this may be proved by simply constructing a truth table). It is not claimed that testing for the membership of an ordered pair, in $Z'$, in the above categories would completely prevent a rule from being hypothesised whose inclusion in rule-collection $\mathcal{R}$ leads to problems. However, by not hypothesising rules from ordered pairs in $Z$ and by not hypothesising rules that fall in any of the above categories, most problems are avoided.

Fig. 5 illustrates rule hypothesising. In ER-BASE, rule hypothesising can proceed only after its complete
content verification. Each rule that is hypothesized must be verified by the expert also. Such a verification is necessary as our rule hypothesizing scheme is based on syntactic considerations only. However, the suggested method can be an important step in filling knowledge gaps in rule-collection \( \mathcal{R} \).

6.5. The structural model

The above partitions are used to create the minimum edge adjacency matrix \( M \). However, since \( M \) is minimum edge, edges that are present in the adjacency matrix \( A \), but not in \( M \), have to be restored in the structural model that is derived from \( M \). Only then will the structural model be a true digraph representation of the relation \( R_{pe} \) in clause set \( \mathcal{R} \).

The interpretive structural model is derived from the corrected structural model.

The interpretive structural model is a visual model that provides an insightful look at the inference structure of the rule-base. For example, see Fig. 4. It is possible to detect some of the problems, visually, in rule-collection \( \mathcal{R} \), by comparing the resultant interpretive structural model with an ideal interpretive structural model. Any deviation from the ideal then implies the possible existence of problems. This is quite obvious in the interpretive structural model of ER-BASE in Fig. 4. The circular clause chain is clearly visible and so is the contradictory chain of inference (also discussed in a later section).

7. Further content verification

This section describes the detection of the problems of redundant rules, subsumed rules, conflicting rules, and unnecessary IF conditions. The no-circular-clause-chain assumption, as already stated in Section 6.1, holds good for the discussion in this section.

7.1. Redundancy

The detection of redundant rules in any rule-base, in general, is difficult. The ordering of premises in the antecedents of redundant rules may be different and the rules themselves may be physically placed at different locations in the rule-base. However, in rule-collection \( \mathcal{R} \), redundant rules can be detected with the help of rule matrix \( N \) and redundancy set \( RS \).

Let rules \( r_{k_1}, r_{k_2}, \ldots, r_{k_n} \in \mathcal{R} \) be redundant rules such that \( 1 \leq k_1 < k_2 < \ldots < k_n \leq \ell \). Furthermore, let the consequent of each of the redundant rules \( r_{k_1}, r_{k_2}, \ldots, r_{k_n} \) be the clause \( c \in \mathcal{C} \), say. It follows from Procedure 1 that

1. in redundancy set \( RS \) there exists 3-tuple \( (k_1, k_2, c) \) for \( x \in P_{k_1} = P_{k_2} = \ldots = P_{k_n} \) and \( k = 2, 3, \ldots, n \),
2. if \( r_{k_1} \) is the first rule in rule-order with consequent clause \( c \) then in rule matrix \( N \) we have \( n_{x c} = k_1 \) for \( x \in P_{k_1} \), and
3. there do not exist any of the rule indexes \( k_2, k_3, \ldots, k_n \) in the column corresponding to clause \( c \) in rule matrix \( N \).

This follows as \( P_{k_1} = P_{k_2} = \ldots = P_{k_n} \) in redundant rules \( r_{k_1}, r_{k_2}, \ldots, r_{k_n} \).

As a special case.

Lemma 7. Let rules \( r_i, r_j \in \mathcal{R}, i < j \), have the same consequent clause and let this consequent clause be \( c \in \mathcal{C} \). Furthermore, let \( r_i \) and \( r_j \) be the first two rules in rule-order in rule-collection \( \mathcal{R} \) with clause \( c \) as their consequent. The two rules \( r_i \) and \( r_j \) are redundant if and only if in redundancy set \( RS \) there exists 3-tuple \((i, j, c)\) for \( x \in P_i \), and there does not exist an entry of rule index \( j \) in the column corresponding to clause \( c \) in rule matrix \( N \).

Proof. Let us assume that rules \( r_i \) and \( r_j \) are redundant. Thus \( P_i = P_j \) and by Procedure 1 it follows that in \( RS \) there exists 3-tuple \((i, j, c)\) for \( x \in P_i = P_j \). Since \( P_j - P_i = \emptyset \), it follows from Procedure 1 that there is no clause \( y \in P_i \) for which \( n_{y c} = j \) in \( N \). Conversely, the existence of 3-tuple \((i, j, c)\) in \( RS \) for every clause \( x \in P_i \) implies \( P_j \subseteq P_i \). Since the column corresponding to clause \( c \) in \( N \) does not contain an entry of rule index \( j \) and since both rule \( r_i \) and rule \( r_j \) have the same consequent in clause \( c \), it implies that \( P_j \subseteq P_i \), \( P_i \subseteq P_j \) and \( P_j \subseteq P_i \) together imply \( P_i = P_j \). From \( P_j = P_i \) and \( C_i = C_j \) together imply that \( r_i \) and \( r_j \) are redundant rules.

It may be noted that in Lemma 7 the assumption that rule \( R_j \) precedes rule \( R_i \) in rule-order and that the two rules are the first two in rule-collection \( \mathcal{R} \) with the same consequent clause \( c \), always implies, by Procedure 1, that in rule matrix \( N \) we have \( n_{x c} = i \) for \( x \in P_i \). Lemma 7 is illustrated with the help of an example in Fig. 6.

7.2. Subsumption

Let rules \( r_j, r_i \in \mathcal{R} \) be two rules, and let rule \( r_j \) subsume rule \( r_i \). From the definition of subsumption we know that \( P_j \subseteq P_i \) and the consequent clause of both the rules \( r_i \) and \( r_j \) is the same. Now let \( r_j \) precede \( r_i \) in rule-order in rule-collection \( \mathcal{R} \). It follows from Procedure 1 that in redundancy set \( RS \) there exists 3-tuple \((i, j, c)\) for \( x \in P_i \), and there exists
at least one clause, say \( b \in P_i \), for which the 3-tuple \((i,j,b)\) does not belong to RS.

As a corollary to Lemma 7, there can be two cases. These are as follows.

**Corollary 4 (of Lemma 7).** Let rules \( r_i, r_j \in R, i < j \), have the same consequent clause and let this consequent clause be \( c \in C \). Furthermore, let \( r_i \) and \( r_j \) be the first two rules in rule-order in rule-collection \( R \) with clause \( c \) as their consequent. Rule \( r_j \) subsumes rule \( r_i \) if and only if in redundancy set RS there exists 3-tuple \((i,j,x)\) for \( x \in P_i \), and in rule matrix \( N \) we have \( n_{xc} = j \) for \( y \in P_j - P_i \) where \( P_j - P_i \neq \emptyset \).

**Corollary 5 (of Lemma 7).** Let rules \( r_i, r_j \in R, i < j \), have the same consequent clause and let this consequent clause be \( c \in C \). Furthermore, let \( r_i \) and \( r_j \) be the first two rules in rule-order in rule-collection \( R \) with clause \( c \) as their consequent. If rule \( r_j \) subsumes rule \( r_i \), then

1. in rule matrix \( N \) we have \( n_{ic} = i \) for \( x \in P_i \),
2. there is no entry of \( j \) in the column corresponding to clause \( c \) in \( N \),
3. in redundancy set RS there exists 3-tuple \((i,j,x)\) for \( y \in P_j \), and
4. there exists at least one clause, say \( z \in P_i \), such that \((i,j,z) \notin RS\).

The proofs of the above corollaries follow from the proof of Lemma 7. Corollary 4 is applicable to the example ER-BASE. The rule matrix \( N \) for ER-BASE (see Fig. 2) has the entries of 1 and 2 in the column of clause \( B \), and \((1,2,A) \in RS \). Both the rules, 1 and 2, have a common premise in clause \( A \) and conclude \( B \). Rule 2 has an additional premise in clause \( C \), giving \( n_{CB} = 2 \), and is thus subsumed by rule 1.

### 7.3. Conflicts

Throughout the discussion in this section, in addition to the no-circular-clause-chain assumption, it is also assumed that there are no consen欣 problems in rule-collection \( R \). Conflicts may then be detected as follows:

1. **Self-Contradictory Rule:** Let rule \( r_k \in R \) be a self-contradictory rule and let clause \( c \in C \) be its consequent clause. Since rule \( r_k \) is self-contradictory, there exists clause \( b \in P_k \) such that \( c \Leftarrow b \). From Procedure 1 it follows that ordered pair \((k,b)\) exists in conflict set CS.

2. **Self-Contradictory Clause Chain:** Let \( x_1, x_2, \ldots, x_k \), \( k > 2 \), be a clause chain formed in a backward chaining of rules in rule-collection \( R \) and let \( x_k \Leftarrow \neg x_1 \), i.e. the clause chain \( x_1, x_2, \ldots, x_k \) is a self-contradictory clause chain. Then,
   - it follows that in the relation partition, \( R(\mathcal{C} \times \mathcal{C}) \) (see Section 6.4), \((x_1, x_k) \in \mathcal{Z} \). Furthermore, there exists a path from clause \( x_1 \) to clause \( x_k \) in the interpretive structural model; and
   - if there is no self-contradictory rule in rule-collection \( R \) with clause \( x_1 \) as a premise and clause \( x_k \) as the consequent, then from Procedure 1 it follows that either ordered pair \((x,0)\) or else \((x_k,0)\) exists in conflict set CS.

3. **Directly Contradictory Rules:** Let rules \( r_i, r_j \in R \) be two directly contradictory rules. From definition of directly contradictory rules we have \( P_i = P_j \) and the consequent clause of \( r_i \), say clause \( a \in C \), is the negation of the consequent clause of rule \( r_j \), say clause \( b \in C \). Now, let us assume that \( r_j \) is the first rule in rule-order in rule-collection \( R \) with consequent clause \( a \), and \( r_j \) is the first rule in rule-order in rule-collection \( R \) with consequent clause \( b \). It follows from Procedure 1 that
   - in rule matrix \( N \) we have \( n_{ic} = j \) for \( x \in P_i \), and \( n_{ic} = i \) for \( y \in P_j \), and
   - either one of the following is applicable:
     - (a) If no self-contradictory rule exists in \( R \) that is self-contradictory in clause \( a \) and clause \( b \), then either \((a,0)\) or else \((b,0)\) exists in conflict set CS.
     - (b) If there exists a self-contradictory rule \( r_k \) in \( R \) that is self-contradictory in clause \( a \) and clause \( b \), then either \((a,k)\) or else \((b,k)\) exists in conflict set CS (depending on which clause is in the premise of the self-contradictory rule \( r_k \)).

It is also possible to detect conflicts in a visual presentation of the interpretive structural model. For example, a self-contradictory clause chain can be detected by tracing clause chains in the presentation. Furthermore, the knowledge engineer can look for certain graph structures, in the presentation, to detect possible directly contradictory rules, e.g. see Fig. 7.

In ER-BASE, clauses \( A,B,D \) constitute a contradictory clause chain in a backward chaining of the rules 1, and 3 as well as the rules 2 and 3. It can be seen that \((A,0) \in CS \) and there exists a path from clause \( A \) to clause \( D (= \neg A) \) in the interpretive structural model of Fig. 4.

### 7.4. Unnecessary IF conditions

Let rules \( r_i, r_j \in R, i < j \), be two rules with unnecessary IF conditions. From the definition of unnecessary IF conditions, \( r_i \) and \( r_j \) each have a clause, say clause \( a \in P_i \) and clause \( b \in P_j \), respectively, such that \( a \Leftarrow b \). Furthermore,
rules \( r_i \) and \( r_j \) have the same consequent clause, and \( P_i - \{a\} = P_j - \{b\} \) (assumed here to be nonempty). It then follows from Procedure 1 that in redundancy set RS there exists 3-tuple \((i,j,x)\) for \( x \in P_i - \{a\} = P_j - \{b\} \), and either one of the following is applicable:

1. If no self-contradictory rule exists in \( \mathcal{R} \) that is self-contradictory in clause \( a \) and clause \( b \), then either \((a,0)\) or else \((b,0)\) exists in conflict set CS.

2. If there exists a self-contradictory rule \( r_k \) in \( \mathcal{R} \) with clause \( a \) as its premise and clause \( b \) as its consequent, say, then ordered pair \((a,k)\) exists in conflict set CS.

A corollary to Lemma 7 is now stated as a particular case:

**Corollary 6 (of Lemma 7).** Let rules \( r_i, r_j \in \mathcal{R}, i \neq j \), have the same consequent clause and let this consequent clause be \( c \in \mathcal{C} \). Furthermore, let \( r_i \) and \( r_j \) be the first two rules in rule-order in rule-collection \( \mathcal{R} \) with clause \( c \) as their consequent. If rules \( r_i \) and \( r_j \) have an unnecessary IF condition in clause \( a \) and clause \( b \), respectively, say, and \( P_i - \{a\} = P_j - \{b\} \neq \emptyset \), then in redundancy set RS there exists 3-tuple \((i,j,x)\) for \( x \in P_i - \{a\} = P_j - \{b\} \), in rule matrix \( N \): \( n_{ix} = j \) for \( y \in P_i - \{a\} \), \( n_{jx} = i \) for \( z \in P_j \), and the only entry off in the column corresponding to clause \( c \) is in \( n_{ix} \).

8. Another example in content verification

Let us suppose that a rule-base for a backlight chaining expert system, which explains the behaviour of a simple door latch [19], as depicted in Fig. 8(a), is under development. Such a system could later be used to programme a causal network for deep reasoning in an model based reasoning system [18]. In the operation of the door latch, as the door is pushed in from the left side of the figure, the locking arm moves up and compresses the spring. When the door is fully in, the arm falls back into the slot in the wedge of the door. To open the latch, you either press the mechanical knob or the electrical switch.

In a high level specification of the rule-base, rules may be described to encapsulate the behaviour of this device, as shown in Fig. 8(b), with the goal of establishing the truth of door_can_move. Rule 9 is added redundantly. Rule 14 may get added as the behaviour of the latch suggests that locking arm up implies door_can_move and also door_can_move implies locking_arm_up. Rule 13 may have been added erroneously and is self-referential. Rule 10 may have been added to emphasise that spring_compressed is enough for locking_arm_up and rule 12 to emphasise that both door_push and spring_compressed together also imply locking_arm_up.

Fig. 9(a) shows the interpretive structural model for this initial set of rules. The important partitions are as follows: \( \pi_1 = [L1 = \{\text{door_can_move, locking_arm_up, core_magnetised, NOT core_magnetised}\}, ..., L5] \), \( \pi_4(L4) = [L = \{\text{core_magnetised, NOT core_magnetised}\}, S = \{\text{door_can_move, locking_arm_up}\}] \), \( \pi_5(S) = \{\text{door_can_move, locking_arm_up}\} \). Clearly, there is a maximal cycle set at level 1 and the errant rules are identified to be rule 5 and rule 14 from the rule matrix. Since rule 5 captures the behaviour of the latch appropriately, we retain it and delete rule 14 from the rule-base. Since, in the rule matrix, the diagonal entry corresponding to clause knob_down is the rule number 13, rule 13 is deleted from the rule-base. With all circularity eliminated, we can now reconstruct the rule matrix and proceed further.

Fig. 10 shows the rule matrix, redundancy set, conflict set, and the various partitions induced by the reachability matrix for the rule-base with rules 1–12 of Fig. 8(b). Fig. 9(b) shows the interpretive structural model for the same. Since the goal is to establish the truth of door_can_move, block \( L_1 \) of partition \( \pi \), clearly shows that NOT core_magnetised and core_magnetised are unreachable conclusions. A missing rule is identified is: IF core_magnetised THEN
locking_arm_up. That there are missing rules is also identifiable in the separate parts partition, $\pi_1$, as there are two blocks in the partition whereas the goal for the system is only one. Checking for redundancy, we find that the redundancy set records a redundancy in rule 2 and rule 9 in clause knob_down. These rules have the conclusion knob_down and are the first two rules in the rule-base with that conclusion. In the redundancy set there is only one 3-tuple, with rule number 2 and rule number 9, (2, 9, knob_press). Furthermore, in the rule matrix there is only one entry of rule number 2 in the column of knob_down and no entry of rule number 9. Thus according to Lemma 7, rule 2 and rule 9 are redundant and rule 9 must be deleted from the rule-base.

As per the redundancy set, rules 4, 10, and 12 are redundant in clause spring_compressed. Rule 4 has no other premise besides spring_compressed. This is known as rule number 4 occurs once in the column of locking_arm_up and does not occur in the redundancy set with any other premise besides spring_compressed. In rule 10 there is a premise NOT door_push and (4, 10, NOT door_push) is not in the redundancy set. Hence rule 4 subsumes rule 10. Rule 4 also subsumes rule 12 as rule 12 has an additional premise in door_push. Similarly, rule 6 also subsumes rule 12 as rule 12 has an additional premise in spring_compressed. On examining rule 10 and rule 12, since there is an entry in the redundancy set of (10, 12, spring_compressed), we see that they satisfy the conditions for unnecessary IF conditions (see Section 7.4). In first eliminating the unnecessary IF conditions NOT door_push and door_push, we find that the two rules 10 and 12 become completely redundant with rule 4. Hence rule 10 and rule 12 are deleted from the rule-base. In doing so we eliminate all conciseness problems since we have now accounted for all entries in the redundancy set.

The presence of (core_magnetised, 0) in the conflict set and an examination of the rules that have core_magnetised and NOT core_magnetised as a premise or conclusion shows that rule 8 and rule 11 satisfy all the conditions for directly contradictory rules (see Section 7.3). Since rule 11 is in error we delete it from the rule-base.

Fig. 9(c) shows the interpretive structural model of the rule-base after the addition of the missing rule and the deletion of the rules as suggested above. The interpretive structural model clearly depicts the inference structure of the rule-base. In this example the model also shows causal connections which can be used for deep reasoning [18].

9. An informal summary of the proposed verification approach

The approach assumes the following essential characteristics of the RBS.
1. A rule in the rule-base is of the IF–THEN form, in which the rule antecedent is a conjunction of distinct, and finitely many (at least one), clauses (atomic propositions or their negations) and the consequent is a single clause, which is a restriction.

2. The use of rules of the disjunctive form is explicitly disallowed.

3. Rules are strictly deterministic in nature and do not involve any probabilities.

4. Inferencing is through backward chaining.

5. Rules are numbered, beginning with '1', which reflects an ordering among the rules as defined by their physical order in the rule-base.

Given these characteristics of the rule-based system, in the approach, content verification of a rule-base is integrated into the framework of modelling and model property identification. The basic steps are:

1. Representation of the rule-base: This step involves the construction of (i) a rule matrix which records the existence of a premise-conclusion relation (pc-relation) in a clause pair, as the rule number of the first rule in which it is observed, (ii) a redundancy set that keeps an account of rules that are redundant in any premise-conclusion pair, (iii) a conflict set which keeps a record of rules that have a conflicting premise and conclusion and also the existence of conflicting clauses in the rule-base that are not pc-related, and (iv) a clause adjacency matrix which is derived from the rule matrix and which represents the pc-relation. These representations are necessary to record all relevant information pertaining to the rule-base, for the purpose of content verification.

2. Development of the reachability matrix: This matrix, which is derived from the adjacency matrix, makes explicit all clause reachabilities that are obtained in a chaining of rules.

3. Identification of various partitions induced by the reachability matrix on the set and subsets of the set of clauses in the rule-base and then using them for content verification. These partitions are:

   (i) The level partition: This partition makes explicit the distribution of clauses across levels as defined by the pc-relation. Since circularity problems corrupt this partition, we assume, as a first step in verification, that circularity is eliminated. This can be done with the help of partitions described in (iii) below. Subsequently, the level partition can be used to eliminate dead end IF conditions, dead-end goals and unreachable conclusions.

   (ii) The separate sets partition: This partition identifies disjoint sets of clauses that are not pc-related, directly or indirectly. Disjointness of rule sets can be used to address completeness in the form of missing rules, and identify stand alone, single clause, self-referent rules.

   (iii) (a) The disjoint and the strong subsets partition of each level identified in (i), and (b) the strongly connected subsets partition as per identification in (a). These partitions help identify and eliminate circularity in the rule-base.

   (iv) The relation partition: In a content verified rule-base, this partition is used in conjunction with the level partition to hypothesise missing rules which if introduced in the rule-base do not precipitate any content problems. Inclusion of an hypothesised rule is, however, contingent to its acceptance by the expert.

4. Development of the interpretive structural model of the rule-base: A minimum edge matrix is first derived from the adjacency matrix, described in step 1 above, which is followed by the development of a visual structural model. Since the structural model is minimum edge, clause adjacency edges that are subsumed are first restored and then its vertices are interpreted as clauses to obtain the interpretive structural model. The interpretive structural model is a visual model that provides an insight look at the inference structure of the rule-base. Some of the content problems can also be detected visually, in the model.

5. Further content verification: Assuming that there are no circularity problems, the rule-base representations developed through step 1 can be used effectively to detect the problems of redundant rules, subsumed rules, and unnecessary IF conditions. Further assuming that there are no conciseness problems, the rule representations can also be used to detect conflicts in the rule-base.

10. Conclusion

This paper has analytically examined the application of a digraph-based modelling approach to the content verification of a specific type of rule-base. In the beginning basic modelling concepts, i.e. the set of clauses which make up the rules in the rule-base, and the pc-relation (premise-conclusion relation) which is defined in the set of clauses, are developed. A modelling process in then defined not only to develop a digraph model of the rule-base, but also to develop other rule-base representations to keep track of rules that have common premises and consequent, and to keep track of rules with conflicting premises and consequent. Through various partitions induced by the reachability matrix, which is derived from the pc-relation set, on the set and subset of the set of clauses, it is shown that problems in completeness can be formally detected. Furthermore, it is shown that the various rule-base model representations can be used to detect problems in conciseness and consistency. Although no exact procedures are described to detect problems, the analytic formulations provide a necessary and sufficient basis for the development of such procedures.

It had been our objective to develop a strong theorem, using the various lemmas put down in this paper, that would
lay claim to the completeness of the modelling method vis-
à-vis content verification. However, the development of such a theorem, as is obvious from the presented work, needs further research.

References