Chapter 4

4 The Removal of Redundancies and Off-line Verification of RDR KBS[20]

In the previous chapter the problem of repetition in RDR KBS was introduced. It was seen that redundant knowledge was not ideal when we seek to understand and explain the knowledge in a KBS and that removal of repetition should improve the ability of RDR to support the reflective activities. In the previous chapter there was also a description of how validation and verification (V&V) is handled in an RDR KBS. This chapter continues the discussion of repetition and V&V in more depth, describing and evaluating work that was aimed at providing a method for compacting RDR KBS and a strengthening of the RDR approaches to V&V. In Section 2.3 it was pointed out that this study was interested in two questions related to reuse. The first one being how to reuse what we already have and the other being the development of a technique that facilitates future reuse. This chapter is concerned with the first question.

Studies by Mansuri et al (1991) and Compton, Preston and Kang (1995) using real and simulated experts[21], respectively, have shown that the repetition problem is not as serious as expected and that the size of an expert built RDR KBS compares favourably in size with various inductively built KBS.

While experts appear to add very simple general rules that result in minimal repetition (Compton et al 1991a), it was the aim of this study to review the repetition and redundancy that does exist to improve the clarity of the knowledge base and its suitability for reuse. It has been claimed that the rule trace provided by an RDR KBS offers better explanation facility than conventional rule based systems (Compton et al 1991b). This is because the user can be shown the path the expert followed in creating the rule and can see not only the rules added to deal with various cases, but also the case which prompted a new rule to be added. The inference pathway reveals the history of cases from which the rules have evolved. It is expected that this explanation facility will be further improved if there are less pathways leading to the same conclusion and if superfluous conditions are removed. Further, it was the goal of this study to preserve the existing knowledge acquired from an expert while gaining the benefits of inductive

---

[20] Sections 4.1 to 4.4 follow closely after Richards, Chellen and Compton (1996). The work on Induct reported in this chapter has been primarily performed by Vijaletchmee Chellen as part of her honours thesis but is included here for completeness and clarity.

[21] The use of a simulated expert is described later under Section 4.2 Experimental Method.

learning which could include: detection and correction of inconsistencies; identification and coverage of gaps in the knowledge; removal of redundancies; and simplification of expert-derived decision rules. Section 4.7 considers the first two benefits. In Sections 4.1 to 4.5 the main focus is on the two latter points which were to be achieved by making rules more general, thereby avoiding the addition of later specific instances. It was also hoped that greater insight into the knowledge captured would be obtained so that the knowledge could be potentially used for other purposes such as teaching and modelling.

The next section gives an overview of the two machine learning (ML) algorithms which were employed. Section 4.2 describes the experimental method used in this study and Section 4.3 looks at the results of the experiments. Section 4.4 reviews the results and discusses the issues raised. Section 4.5 proposes how repetition could be removed from MCRDR KBS and Section 4.6 summarises the findings of the repetition studies. Section 4.7.1 looks at the use of rough set theory for the verification of conventional rule-based systems and 4.7.2 applies the verification techniques to MCRDR KBS.

4.1 Compacting a Manually-built RDR KB using Machine Learning[22]

Machine learning can be seen as one way to alleviate the KA *bottleneck* problem. Knowledge bases can be built much more rapidly using ML and usually have many fewer rules than those built manually. KA using manual RDR is not as quick as with a ML technique but it has been found to produce rule bases of similar size to those built using ML techniques such as ID3, C4.5 and Induct and matures more quickly, requiring a small number of cases to achieve high accuracy levels (Mansuri et al 1991). Manual RDR is a rapid method of KA, approximately 40 times faster than traditional KA techniques (Compton and Jansen 1988). The compactness of manual RDR and it's exception structure prompted Brian Gaines to modify his Induct algorithm to output RDRs. The modified algorithm is known as Induct/RDR (Gaines 1991a, Gaines and Compton 1995). Using the Garvan clinical pathology dataset of 9551 cases, with the first 8000 cases as the training set and testing on the remaining 1551 cases, Induct/RDR generated 195 rules with 2.31% errors (Gaines and Compton 1992) which is comparable

[22] Sections 4.1 to 4.4 follow closely the paper “Richards, D., Chellen, V. and Compton, C (1996) The Reuse of Ripple Down Rule Knowledge Bases: Using Machine Learning to Remove Repetition *Proceedings of Pacific Knowledge Acquisition Workshop PKAW’96*, October 23-25 1996, Coogee, Australia, 293-312.

to 550 rules with 2.8% errors and 502 rules with 2.71% errors on 9514 cases reported in Mansuri et al (1991) with human built systems.

The above results raise the question of whether manual RDR is needed. However, the successfulness of using ML to derive knowledge depends on the availability of sufficient well-classified cases. As noted above manual RDR outperforms ML techniques when only a small number of cases are available. It has also been shown that the expert enters new concepts that do not exist in the historical data (Compton et al 1991a) and the rules output by ML techniques, such as C4.5 are not as comprehensible to experts as the RDR exception structure (Catlett 1992)[23]. This is one reason why Gaines implemented Induct/RDR. One alternative proposed by Gaines and Compton (1992) is to use manual RDR to build an initial KBS and when the KB becomes large, use ML, Induct in particular, to minimise the size of the rule base. This study pursues Gaines' proposal.

The ML algorithms chosen for this study were Induct (Gaines 1989) and rough set theory (RST) (Pawlak 1991). The use of Induct/RDR was an obvious choice since it already produced RDR output. At the time of these studies RST was being considered as a way of finding the key concepts and relationships in an RDR KBS and the ability to strip out redundant attributes was attractive for the repetition studies. A key feature of RST was that it only used one of each case and did not rely on multiple instances in generating rules unlike methods based on statistics or information gain. This seemed important when the goal was to analyse rules not cases. Additionally RST was similar to RDR in its use of differences to assist classification. An overview of both Induct and RST follows.

4.1.1 An Overview of Induct

Induct uses induction to build a knowledge base in terms of classes, attributes, attribute values and rules (Gaines 1989b). Figure 4.1 below reveals the basis of the statistical tests underpinning Induct's rule generation. The aim is to find a measure, r, which determines how good is a selector S (attribute-value combination). That is, we want a selector which is least likely to predict the target class by chance.

The first step is determine "what is the probability that random selection of the same degree of generality would achieve the same accuracy or greater" (Gaines 1989b). To calculate this probability we let Q be the target class in our universe of entities, E, for

---

[23] Catlett (1992) has further suggested that because of RDR's comprehensibility that it be used as a mediating representation, where rules generated by ML are converted into RDR format for perusal by experts.

which Q(e) holds and S be the selected entities in E for which S(e) holds. We are interested in finding the probability of C(e) which is the intersection of Q(e) and S(e) and is given by:

$$C \equiv \{e: e \in E \wedge S(e) \wedge Q(e)\} \qquad (4.1)$$

The probability of getting an instance belonging to the set Q when an entity is selected at random from E is given by:

$$p = \frac{|Q|}{|E|} = \frac{q}{e} \qquad (4.2)$$

Thus, the probability of getting c (and possibly more) instances at random from Q when s items are selected from E is given by:

$$r = \sum_{i=c}^{s} {}^{s}C_i p^i (1-p)^{s-i} \qquad (4.3)$$

This is a measure of correctness of the rule produced and is an exact calculation of the probability of an event occurring unlike the entropy algorithm used in C4.5. The aim is to choose a selector S which minimises r, that is, a selector S for which the likelihood of having c entities at random from E is low. Assume the better the selector, the lower the value of r. "The advantage of selecting r as a measure of the correctness of a rule is that it is easily understood as the probability that the rule could be this good at random and that it involves no assumptions about the problem such as sampling distributions" (Gaines 1989b, p.99).

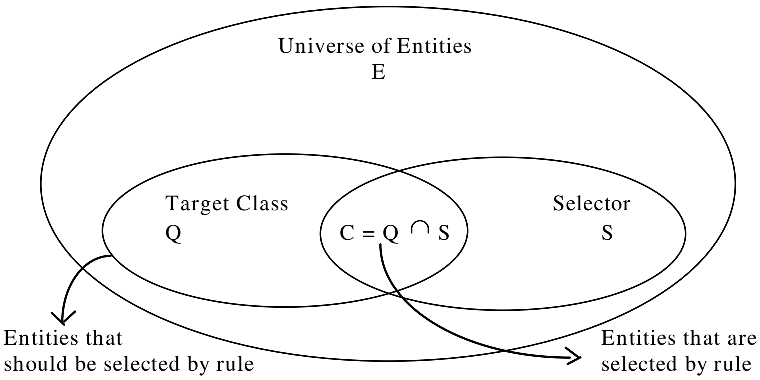


Figure 4.1 - Induct's statistical basis (from Gaines 1989b)
where

E represents the set of instances in the dataset.
Q represents the set of instances of the target class.
S represents the set of instances satisfying a predicate P.
P consists of a set of attribute value pairs.
C represents the instances of the target satisfying the predicate P.

Induct was reimplemented on the Unix platform so that we could add our own code to support the various clean up algorithms used on the cases as described in Section 4.2.

Vinduct (Vija Induct) is our Unix implementation and the algorithm employed to generate rules is presented in Figure 4.2. We have not developed a new algorithm but we also do not claim that the algorithm is identical to Gaines' Induct algorithm. The Vinduct algorithm included various heuristics to determine situations where the selector value was the same and a one step lookahead to determine the impact of adding a condition to the selector value. From personal communication between Chellen and Gaines similar heuristics have been used and the results obtained by our implementation is satisfactorily close to the results of Induct.

```
        ReadCases (Dataset)
        Find Default Conclusion
        Process (Dataset, Default)

Process(Dataset,Default)

        While (Dataset contains cases of class other than Default)
          Target Class =Second most frequent conclusion other than default
          MakeRule (Target, Default)
          SelectedCases = Cases from Dataset covered by rule
          Process (SelectedCases,Target)
          Remove SelectedCases from Dataset
        end while

MakeRule (Dataset,Default)
        Rule = { }
        lastprobability = 1

        While (Number of attributes in rule is less than the Total Number of attributes)
          Find the attribute-value pair Av1 not already in rule which in conjunction with the
                  clauses in rule produces the minimum measure of correctness r , rmin 1
          If (rmin1 > lastprobability)
                  Add Av1 to rule
                  lastProbability = rmin1
          else
                  Find the attribute-value pair Av2 not already in rule which in conjunction with
                          Av1 and the clauses in rule produces the min. measure of correctness r, rmin2
          end if
          If (rmin2 < lastProbability/10)
                  Add Av1 and Av2 to rule
                  lastProbability = rmin2
          else
                  exit
          end if
        end while
```

Figure 4.2: The Vinduct Algorithm

4.1.2 An Overview of Rough Set Theory

Pawlak (1982) first introduced rough set concepts as a means of discovering relationships in data. The idea of consistency is central to RST and hinges upon the concept of indiscernability rather than the elusive concept of truth. We are concerned that the condition attributes (P) are able to discern the decision class Q_i from another decision class Q_j, where $i \neq j$. RST has been used successfully to support the classification of uncertain or incomplete data in a range of domains (Fibak, Slowinski and Slowinski 1986, Krysinski 1990 and Slowinski, Slowinski and Stefanowski 1988). Pawlak defines a knowledge base to be a family of classifications[24] in the universe of objects being investigated. Each family member forms a subset or category. Membership of a subset is sometimes uncertain and the concepts of upper and lower approximations, $\overline{R}X$ and $\underline{R}X$ respectively, are used to approximate membership, where X is the concept and R is the knowledge held. If a concept is known, that is all objects with particular attribute value (A-V) pairs belong to the same classification, then those objects belong to the lower approximation and form the positive region. More formally we define $POS_R(X)$ as the set that can be certainly classified as X using knowledge R. Those objects that certainly don't belong form the negative region. If the information we have is not sufficient to classify an object then such objects belong to the boundary region. The upper approximation is the set theoretical union of the positive and boundary regions. Using these regions, classified sets are used to help approximate the classification of the unclassified set. This replaces the notion that set membership is a primitive relation with the view that the membership relation is based on what knowledge is held.

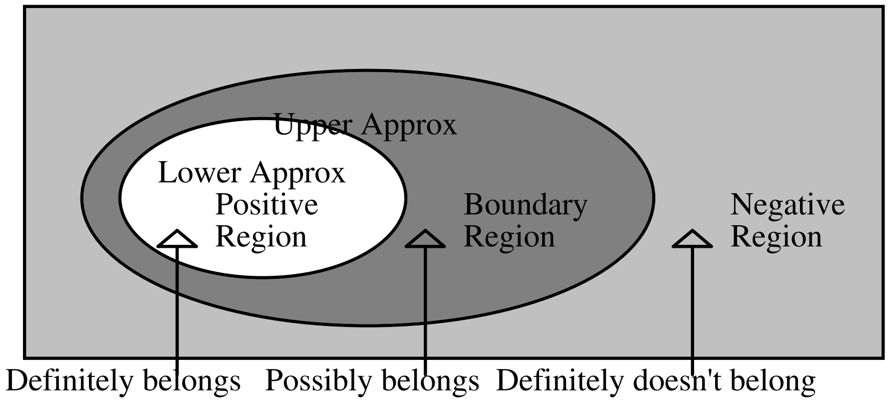


Figure 4.3: The Partitions in an RST Universe

The regions described above and shown in Figure 4.3 are used to find dependencies (rules) through the generation of the core and the reducts of a knowledge base. Reducts are collections of attribute value pairs by which a conclusion can be reached and can be seen as alternate rule pathways. The core is the intersection of the reducts. Calculation of the core and reducts can be used to simplify decision tables. This is achieved through the removal of redundant attributes (columns), rules (rows) and superfluous attribute values (Pawlak 1991, p.71).

[24]The discussion that follows is restricted to problems of classification, although the usefulness of RST for a construction problem is being investigated.

In the following example it is shown how reducts and cores are found using RST. The data used in this example are taken from an article which appeared in the Sydney Morning Herald (SMH) during the constitutional convention held in February 1998 in Canberra. There are six parties and thirteen issues which are shown in Figure 4.4 and referred to as objects and attributes, respectively. The data have been simplified into a decision table as shown in Table 4.1.

1- Breakaway Republican 1(Tim Costello)	a -President as Head of State
2- Breakaway Republican 2 (Peter Beattie)	b -Governor General as Head of State
3- Aust. Rep. Movement (ARM)(Malcolm Turnbull)	c -HOS political
4- McGarvie Model (Richard McGarvie)	d -HOS not political
5- Monarchist (Kerry Jones)	e -Popular Election
6- Non Aligned (Phil Cleary)	f- President elected by 2/3rd Parliament
	g -Self Nominating
	h -Nominees Voted
	i -Nominees voted for by a mixed council
	j - Nominees chosen by parliament
k -Nominee chosen by PM	
	l- HOS appointed
	m - HOS appointed by Queen
	n - HOS appointed by 3-person council

Figure 4.4: The parties (objects) and issues (attributes) in the Head of State Debate

U	*a*	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*
1	1	0	**1**	**0**	1	0	1	**1**	1	0	**0**	**0**	0	0
2	1	0	**1**	**0**	1	0	0	**1**	0	1	**0**	**0**	0	0
3	1	0	**1**	**0**	0	1	0	**0**	0	0	**1**	**0**	0	0
4	0	1	**0**	**1**	0	0	0	**0**	0	0	**0**	**1**	0	1
5	0	1	**0**	**1**	0	0	0	**0**	0	0	**0**	**1**	1	0
6	1	0	**1**	**0**	1	0	0	**1**	1	0	**0**	**0**	0	0

Table 4.1: The Decision Table for the Selection of a Head of State. The attributes shown in bold are later removed using RST concepts as they are dispensable.

Decision tables may be reduced through a three step process (Pawlak 1991, Chptr Six):

1. Eliminate duplicate rows.
2. Eliminate superfluous attribute values.
3. Compute the reducts of condition attributes.

Since there are no duplicate rows (each party has a different viewpoint to each other party) we can not reduce the table using step 1. We can however remove superfluous attribute values. As mentioned, discernability is a key factor in RST. From table 4.1 we can see that some issues (attributes a and c; b, d and l; e and h; f and k) have the same values for each party. Duplicate columns are shown in bold and can be removed. Discarded attributes are termed dispensable as they do not affect the decision outcome. We can also see that the table remains consistent (that is, the same condition attributes do not result in different decision attributes) by dropping either attribute a or b. After the various attributes above have been dropped we end up with Table 4.2.

U	*a*	*e*	*f*	*g*	*i*	*j*	*m*	*n*
1	1	1	0	1	1	0	0	0
2	1	1	0	0	0	1	0	0
3	1	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	1
5	0	0	0	0	0	0	1	0
6	1	1	0	0	1	0	0	0

Table 4.2:Reduced set of attributes

From Table 4.2 we compute the core values which represent the minimal characteristics of each participant. How this is performed is now given for the first object. The reducts and core values for each other object are found using the same process. We consider each attribute at a time and find the set of objects that have the same value as the first object for the attribute being evaluated. The parties are identified by their number in Table 4.1.
This can be shown as:

$\{[1]_a, [1]_e, [1]_f, [1]_g, [1]_i, [1]_j, [1]_m, [1]_n,\} = \{ \{1,2,3,6\},\{1,2,6\}, \{1,2,4,5,6\},\{1\}, \{1,3,4,5,6\}, \{1,2,3,4,6\},\{1,2,3,5,6\}\}$

We are interested in finding the intersection of these sets to determine what issues Breakaway Republican 1 shares with the other parties. $[1]_{a,e,f,g,i,j,m,n} = [1]_a, \cap [1]_e, \cap [1]_f, \cap [1]_g \cap [1]_i \cap [1]_j \cap [1]_m \cap [1]_n = \{1\}$. This set also represents the decision category since we are treating each attribute as a condition and decision attribute. To find the reducts and cores we drop one attribute at a time to see if the intersection of the remaining attributes is in the decision category. For example:

1. $[1]_a, \cap [1]_e, \cap [1]_f, \cap [1]_g \cap [1]_i \cap [1]_j \cap [1]_m = \{1\}$
2. $[1]_a, \cap [1]_e, \cap [1]_f, \cap [1]_g \cap [1]_i \cap [1]_j \cap [1]_n = \{1\}$
3. $[1]_a, \cap [1]_e, \cap [1]_f, \cap [1]_g \cap [1]_i \cap [1]_m \cap [1]_n = \{1\}$
4. $[1]_a, \cap [1]_e, \cap [1]_f, \cap [1]_g \cap [1]_j \cap [1]_m \cap [1]_n = \{1\}$
5. $[1]_a, \cap [1]_e, \cap [1]_f, \cap [1]_i \cap [1]_j \cap [1]_m \cap [1]_n = \{1,6\}$
6. $[1]_a, \cap [1]_e, \cap [1]_g, \cap [1]_i \cap [1]_j \cap [1]_m \cap [1]_n = \{1\}$
7. $[1]_a, \cap [1]_f, \cap [1]_g, \cap [1]_i \cap [1]_j \cap [1]_m \cap [1]_n = \{1\}$
8. $[1]_e, \cap [1]_f, \cap [1]_g, \cap [1]_i \cap [1]_j \cap [1]_m \cap [1]_n = \{1\}$

U	*a*	*e*	*f*	*g*	*i*	*j*	*m*	*n*
1				1				
2						1		
3			1					
4								1
5							1	
6				0	1			

Table 4.3: The core values - the core issues

We can see that the results for all equations except number 5 (the one that dropped attribute g) are within the decision category {1}. Thus, those attributes form the reducts for the first object (Breakaway Republican 1). For example in the first equation the reducts are a, e, f, g, i, j and m. Each result within the decision category offers an alternative set of reducts (or rules). The core is the intersection of the reducts which results in the core value being g(1) = 1 (meaning the value of attribute "g" for object "1" is "1"). When the cores are computed in the same manner for all parties we get the cores shown in Table 4.3. A textual description of Table 4.3 is given in Figure 4.5 and reveals the key issues for each party. It must be noted that the cores found using RST are not necessarily the core or key issues in the decision but are what differentiates one party from another.

Breakaway 1:	HOS should be self nominated.
Breakaway 2:	Parliament should select 3 nominees for HOS.
ARM :	HOS should be elected by a 2/3rds majority of Parliament.
MCGarvie :	3 person council appoints the HOS.
Monarchist :	Queen appoints the HOS
Non Aligned :	Nominees voted by mixed council.
	HOS should not be self nominated.

Figure 4.5: The key issues in the HOS Issue

In this study RDR KBS have been converted into decision tables which were used as input and RST has been used to find the reducts which form the new and reduced set of output rules. Cores were not computed as they tend to strip out too much relevant information. The various cleanup algorithms described later in Section 4.2.2 and shown in Figure 4.6 altered the content of the decision tables and thereby affected the output but the process described in this section for computing reducts was not changed.

Previous work using rough sets has been concerned with analysis of cases rather than rules[25]. However, it seemed that RST would be eminently suitable for the compaction of rule sets since RST is concerned with reducing the initial set of consistent algorithms to a smaller perhaps less consistent set through the removal of multiple instances (objects in rough set terminology) and redundant attributes. RST is able to handle missing values and the rules developed are not affected by the number of each example as are many other ML techniques, such as C4.5 or Induct. The implementation of RST used in these experiments will be referred to as RS/RDR and comprises C code which converts an RDR KBS into a flat decision table structure and back again and routines, also written in C, from the Rough Sets Library (RSL) (Gawrys and Sienkiewicz 1993) which generate reduced rules from the decision table. The interface, compaction and verification procedures and code are the same as those used by Vinduct.

4.2 The Experiments

As stated previously, it was critical that compaction be achieved without loss of the existing knowledge acquired from the expert which meant no loss in classification

[25]An exception is research by Colomb and Sienkiewicz (1995) which used RST to reduce the *inflation* that can occur when a propositional ES is converted into a decision table.

accuracy. Numerous experiments were conducted to determine the most appropriate technique for compaction. The factors affecting the design of the experiments are discussed in the next section followed by a description of the experiments.

4.2.1 Design Considerations

An RDR knowledge base consists of a set of rules together with a set of cornerstone cases, that is, the cases which resulted in the creation of a rule. It was important that we did not lose information needed to maintain the existing classificatory accuracy of our KBS. Cases and rules together provided information that was not available in cases or rules alone. The cornerstone cases provided the context in which a rule applied and the rule showed the salient features in the case. Use of the cornerstone cases alone was not appropriate for training as they only represented a small number of the total cases and may have contained attributes not used in determining a conclusion. By filtering the cases using the rules it was possible to extract the irrelevant attributes from the set of cornerstone cases. It was anticipated that the cleaned up set of cases would be a suitable training set for a machine learning algorithm, like Induct/RDR, which would produce a smaller knowledge base that performed at least as well as the original knowledge base.
It is an inherent feature of RDR that the size of the knowledge base produced is dependent on the order in which the cases are presented and will affect the amount of repetition. To ensure that the worst case is obtained in terms of repetitious knowledge, the data used in the following experiments were randomised to produce several datasets containing the same data but with different ordering of the cases.

The study was carried out on three different domains using the following datasets : Chess and Tic Tac Toe from the University of California Irvine Data Repository and the Garvan data from the Garvan Institute of Medical Research, Sydney.

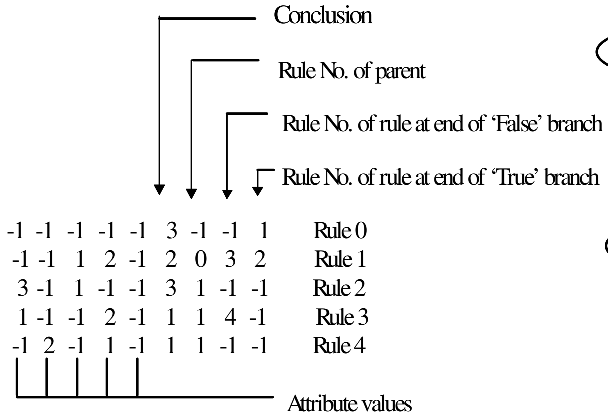
4.2.2 Experimental Method

Compton, Preston and Kang (1995) proposed the use of another KBS as a simulated expert from which knowledge can be acquired because of the unavailability of experts for controlled studies. This study attempts the compaction of a knowledge base built using a simulated expert as the source of expertise. A different synthetic expert was built for each of the domains, using machine learning techniques. The whole dataset was used as the training set so as to provide maximum expertise. The three datasets were then randomised prior to the construction of the expert systems to avoid ‘lumpiness’ (Compton, Preston and Kang 1995). The Garvan dataset was randomised to produce four different datasets. Both the Chess and TicTacToe datasets were randomised to produce nine different datasets. For each dataset 25% was kept as the test set, while the other 75% was used as the training set.

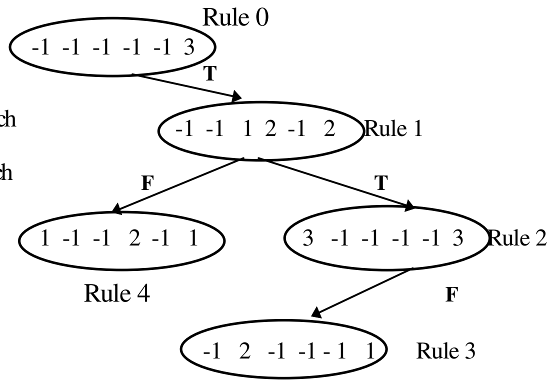
For each of the randomised datasets of each domain, fractions of the training set were used to produce knowledge bases of varying sizes using the method described in Compton, Preston and Kang (1995). New rules were incrementally added to the expert system under construction whenever the conclusion provided by the expert system for a case was found to disagree with the conclusion reached by the simulated expert. The rules were created by selecting four conditions from the intersection of the rule trace of the simulated expert, for the case and the difference list for the case. The difference list consists of conditions in the misclassified case that differed with the cornerstone case of the rule in the expert system that caused the misclassification of the case. All the conditions in the intersection were used in rule formation in the event of the intersection containing less than four conditions. An empty intersection resulted in four conditions being selected from the difference list or less if four were not available.

For each of the resulting knowledge bases, the cornerstone cases were filtered to remove all the irrelevant attributes. The clean-up process involved taking a set of cornerstone cases, a set of rules and running each case against the rules keeping for each case only those attributes that caused a rule to either fire or those that caused a rule to misfire, while replacing all other attributes by missing values. Figures 4.6.(c) - (e) shows the steps involved in removing the irrelevant attributes from the case ' 2 1 1 2 3 2' by using the ruleset in figure 4.6(a). This method of combining cornerstone cases and rules and creating a new *case* with relevant A-V pairs will be referred to as Cleanup 1. From figure 4.6, we can see that a case was initialised to missing values (-1) and then traversed the RDR tree, adding back case values for attributes used in rule conditions.

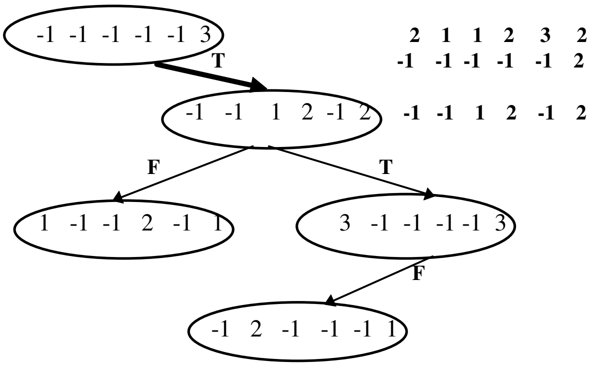
Cleanup 1 captured the attribute value pairs that determined the path. An alternative approach, known as Cleanup 2, applied to the handling of conditions on the false branch and attempted to generalise and replace a value that caused a rule to fail with the negation of the value in the rule. This meant that if the possible values were 1-4, the value in the case was 1, the value in the rule was 2, the value in the cleaned-up case was -2 and would treat the values 1,3 or 4 as matches. This was seen as a better interpretation of the knowledge. It was assumed that when the expert gives the attribute in the rule a particular value, the expectation is that any cases with all values except for the one specified will fail. Where only two values were possible such as true or false, if the value in the rule was false, the value in the case was taken. This is the same as Cleanup 1. To reduce complexity and facilitate the techniques described as Cleanup method 1 and 2, the initial KB were converted into a format where attributes only had positive values and negation was not present.



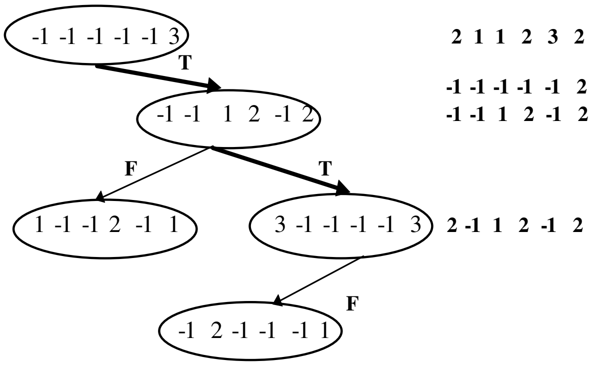
(a) A sample ruleset. The first five numbers are values assigned to the five rule condition attributes. The sixth-ninth numbers are the conclusion, the parent rule number, its leftchild, and rightchild. -1 indicates the absence of any value



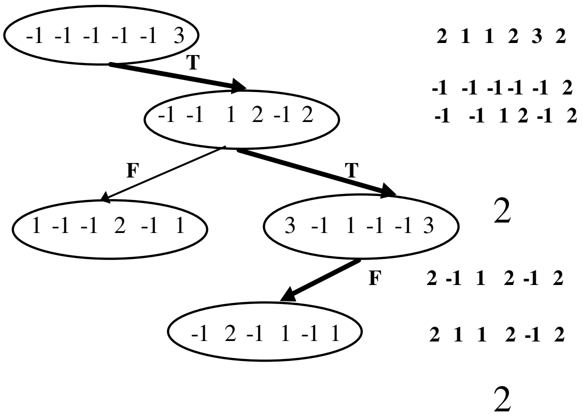
(b) The binary tree representation of the ruleset given in (a)



(c) *The case '2 1 1 2 3 2' is run through the tree to extract relevant attributes. Rule 0 fired unconditionally, so no attribute in the case is responsible for the rule firing. Rule 1 fires because of the third and fourth attribute so these are kept in the filtered case.*



(d) Rule 2 fails to fire because the original case has value 2 instead of 3 for the first attribute. So the new filtered case is ' 2 -1 1 2 -1 2'



(e) Rule 3 fails to fire because the case has value 1 instead of 2 for the second attribute.

Initial case : 2 1 1 2 3

Filtered case : 2 1 1 2 -1

Figure 4.6: The Steps in Cleanup Method 1

Filtered sets of cases, using Cleanup 1 and Cleanup 2, were then used as the training sets for Vinduct and RS/RDR. This resulted in a set of reduced knowledge bases. The percentage error of the original knowledge base and the compacted knowledge base on the test set was then determined. Plots of the original knowledge base size and compacted knowledge base size against the percentage of total cases used as the training set were obtained for each randomised dataset of each domain. The percentage error of the original knowledge base and the percentage error of the compacted knowledge base versus the percentage of total cases used as the training set were also plotted. One other method of compaction, termed Flat True rules, was used with RS/RDR which took only the true branches and ignored the false branches. The purpose of this was to gain some understanding of the impact of the false branches. The results, discussed later, are interesting.

Care was taken that the only difference between the use of Vinduct or RS/RDR was with respect to how the rules were generated. Both techniques made use of a default diagnosis to catch cases not fired and the Cleanup algorithms and method of evaluation were identical, in most cases the same C routines were used.

4.2.3 Missing Values

The handling of missing values needs special attention because typically ML techniques like Induct and RST are used on cases rather than on rules or combinations of rules and cases. When we are dealing with rules we have many more missing values than we would expect to find in a set of cases. For Induct, a selection based on missing values are allowed to contribute to false positives but not to correct positives (Gaines 1989). Thus when measure r, is being calculated for a selector S, those cases having missing values for any of the clauses in S, are considered to belong to S if and only if they have a classification other than the Target class Q. Therefore, as shown in Figure 4.7, cases with missing values can affect the value of $\underline{s}$ but not that of $\underline{c}$.

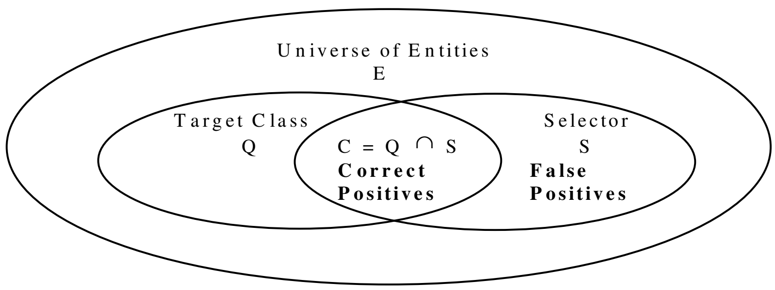


Figure 4.7- Set Representation of the Dataset (Adapted from Gaines 1989b)

When a rule has been induced and the dataset has to be divided into the 2 subsets consisting of those cases in the dataset that are covered by the rule and those that are not covered by the rule, cases with missing values for any of the conditions in the rule are considered to be in both subsets. This is good because it allows for the worst case since a case can be both covered and not covered by a rule depending on the value assigned to the missing attribute value. For the purpose of the repetition studies this approach allows us to treat rules as cases with missing values and using Induct we can reduce the rule-set (Gaines 1989b). The handling of missing values in RST is similar as they are treated as a match on all values. The main difference is that when a rule is generated a missing value is not used to discern two objects so they will not cause rules to be generated for each possible value and do not affect the generation of reducts (rules).

4.3 Results

The graphs of Figures 4.8(a) and 4.8(b) represent the growth curve and the error curve, respectively, for the knowledge bases built by the simulated expert (Initial KB) and the compacted knowledge base (Compacted KB) produced by Vinduct for the Garvan domain using the Cleaned Up (using Cleanup 1) cornerstone cases. The shaded area represents the range of values obtained from the four different randomised datasets. The x-axes of all the graphs represents the proportion of the total dataset used as the training set for the manual expert system. Similar graphs were produced for TicTacToe and Chess.

It is clearly visible from the graphs that compaction tends to decrease the size of a knowledge base by a factor of 2 or more. However, this promising result is shadowed by a considerable increase in error rate. For example, at 60 percent of the number of cases in the Garvan dataset, the knowledge base size decreases from an average of 800 rules to 250 rules. The error rate on the other hand, increases from an average of 3 percent to 20 percent. Cleanup method 1 is the one shown as the results taking the negation of the rule, Cleanup method 2, were worse. Table 4.4 shows similar figures for the Garvan data produced by RS/RDR. More description of the RS/RDR results is given below the table.

Table 4.5 provides a comparison of the results obtained from Vinduct and RS/RDR on the Garvan dataset randomised on seed 10. A synopsis of the figures accompanies the table. The results obtained on the TicTacToe and Chess datasets were similar to the Garvan results using Vinduct or RS/RDR.

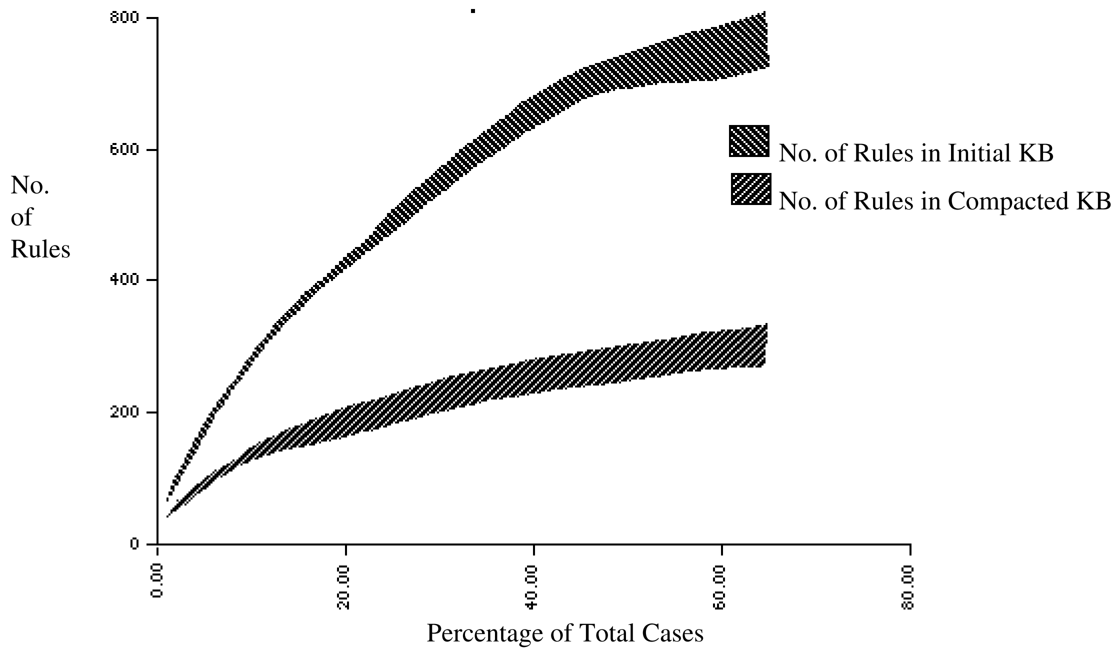


Figure 4.8(a): Growth Curve

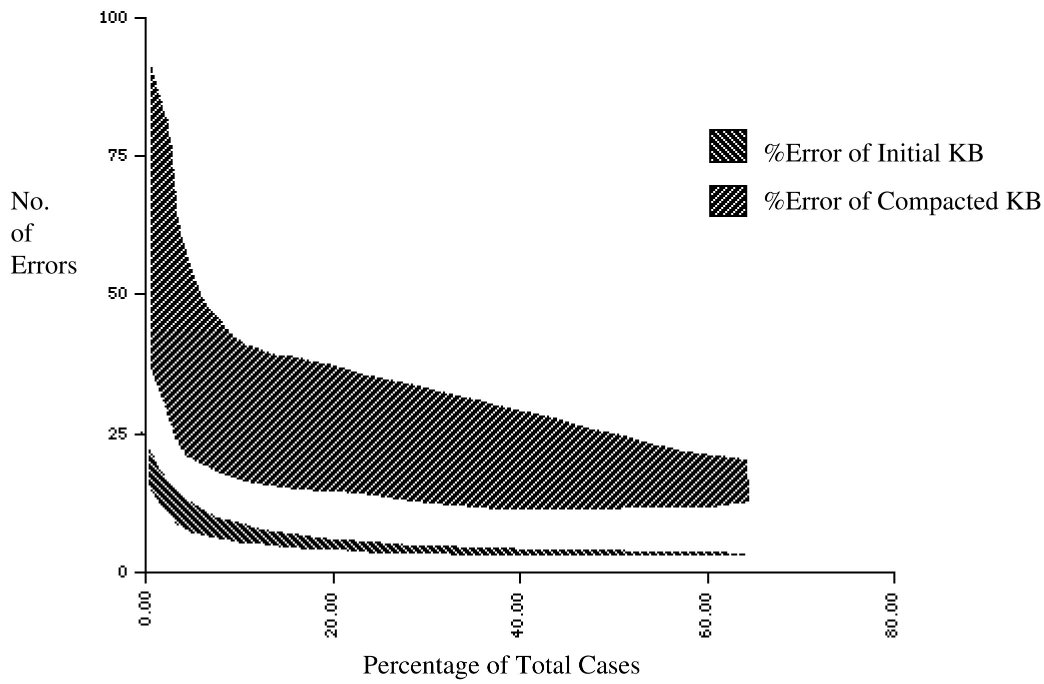


Figure 4.8(b): Error Curve

Figures 4.8(a) and (b): Comparison of Growth and Error Rates for the Initial KB and the Compacted KB (using Cleaned-up Cornerstone Cases) in the Garvan Domain using the Vinduct algorithm.

No. Rules	Randomisation Seed	%error Initial KB	True Flat rules #rules	%error self	%error testset	Cornerstone cases raw data #rules	%error	Cornerstone cases Cleanup 1 #rules	%error
100	10	9.74	63	46.94	50.47	48	33.04	61	47.88
	20	10.37	66	54.08	23.42	54	42.79	70	73.25
	30	10.09	64	44.90	26.28	59	38.83	66	41.12
	40	13.10	68	37.76	22.69	55	56.15	67	58.55
500	10	2.91	187	68.47	28.00	184	43.93	230	44.05
	20	3.01	245	61.65	23.95	170	36.27	254	40.87
	30	3.59	231	67.27	26.50	178	22.99	256	29.61
	40	3.37	249	61.65	23.15	172	42.33	263	45.34
958	10	0.86	319	71.89	26.83	172	42.34	386	32.01
	20	0.56	388	70.93	17.48	299	29.30	441	30.73
	30	0.34	319	71.54	26.83	290	29.99	457	20.27
	40	0.44	319	71.73	26.83	288	32.11	438	42.48

Table 4.4: The Garvan Results using RS/RDR on Cornerstone cases

Column 1 shows the number of rules in the Initial KB. Column 2 shows the seed used to generate random datasets. The third column gives the number of errors on the initial rules when verified against the test set, which was the last 25 percent of the 21822 Garvan cases. The fourth column shows the number of rules and percentage of errors on the test set when the rules were flattened, which meant taking only the true branch and ignoring any impact that the false branch might have. It is interesting to note that the errors substantially decreased on the test set but was very high on the cornerstone cases (self) associated with the rules that were flattened. The fifth column shows the number of rules and percentage of errors when the cornerstone cases alone, (ie. no attributes removed) are used to generate a reduced set of rules. The sixth column shows the reduced number of rules and percentage of errors when Cleanup Method 1 is used on the cases. In Cleanup 1, all attribute-values on true branches are kept. Attributes that are used by rules on the false branch and whose values differ from those in the cases are also kept. All attributes not used by the true or false rules are replaced by missing values. Where "self" is not shown the results were 100% accuracy on the associated cornerstone cases.

Cleanup 1 was the initial attempt at producing a compacted set of cases that only included the pertinent features of the data in the expectation that a smaller set of rules would be produced. However, the poor results from Cleanup 1 which used the value in the case for rules on the false branch resulted in the conjecture that the rule was not really saying take the value in the case but take the negation of the value in the rule. However, the results using both RS/RDR and Vinduct showed a further decrease in accuracy with this approach. On some datasets the number of rules generated exceeded the number in the Initial KB. This is similar to the ‘inflation’ problem described by Colomb and Sienkiewicz (1995) which results when we expand the negation to the range of possible values minus the one being negated. All results shown in tables and graphs refer to Cleanup Method 1.

All of the abovementioned results show higher error rates when Cleaned Up cornerstone cases are used, rather than straight cornerstone cases. All error rates were unacceptably high. It had been desirable to see if the approach would work on cornerstone cases as this reduced the storage and processing requirements and the cornerstone cases provide a smaller set of possible A-V pairs, thus reducing the search space for good classifiers. However, the poor results necessitated a whole new approach. Instead of using the cornerstone cases in the clean up process, the original cases from the dataset were combined with the rules generated by the simulated expert to produce a new set of Cleaned Up cases. For example, rather than compacting the 88 cornerstone cases and 88 rules generated by the simulated expert for the first 545 cases, the 545 cases were compacted using the 88 rules and the 545 Cleaned Up cases were used as input.

# Cases used to learn	Initial # rules	KB %error	Raw corners #rules Induct	Raw corners #rules RST	Raw corners %error Induct	Raw corners %error RST	Cleaned corners Cases #rules Induct	Cleaned corners Cases #rules RST	Cleaned corners Cases %error Induct	Cleaned corners Cases %error RST
545	88	11.23	36	44	52.33	31.48	43	53	39.58	41.92
1090	138	9.43	50	65	33.82	25.89	58	81	29.66	49.23
3270	295	4.89	88	118	43.77	29.19	124	150	29.88	46.52
5455	402	3.39	117	148	37.47	44.20	156	190	20.23	42.07
10910	624	1.99	165	219	32.16	26.82	221	272	39.56	39.74
16365	796	1.26	205	259	40.49	25.39	268	339	9.38	31.53

Table 4.5: Comparison of the Garvan results using Vinduct and RS/RDR using Cornerstone Cases

The first column is the number of the 21822 Garvan cases used by the simulated expert to build the Initial KB. The number of cases represent 2.5, 5.0, 15.0, 25.0, 50.0 and 75.0 percent of the total dataset. The number of rules and percentage of errors for the Initial KB are given in the second column. All testing was performed on the last 25 percent of the 21822 Garvan cases. The same rules and cornerstone cases were used by Vinduct and RS/RDR to generate a reduced set of rules. Column three shows the number of rules generated by both algorithms when the cornerstone cases are used for training the system. There is at least a two-fold reduction in the number of rules, with Vinduct producing smaller rule sets. The fourth column shows the error rates on the new set of rules generated from the cornerstone cases. Neither algorithm performs well, but RS/RDR's error rate is lower on all but one set of cases. The fifth and sixth columns give the number of rules and percentage of errors when a Cleaned Up set of cases generated from the cornerstone cases and rules are used to derive a smaller set of rules. The number of rules generated by RS/RDR and Vinduct goes up and the number of errors also increases compared to the use of cornerstone cases alone, with RS/RDR performing the worst.

Table 4.6 shows that when the algorithms are applied to the original cases, RS/RDR produces minimally more compact KB than Induct/RDR but produces more errors until 50 percent of cases are seen and then performs better than both Induct/RDR and the Initial KB. At 25% of cases RS/RDR produces less errors than Induct/RDR on cleaned up cases and has less errors than both ML algorithms when used on straight cases. Both algorithms produce larger KBs with Cleaned Up cases than with cases alone, with Vinduct producing smaller rule sets than RST. The error rates for Induct/RDR and

Vinduct are about 2% with and without Cleaned Up cases and therefore about 1% higher than the Initial KB. RS/RDR error rates drop using Cleaned Up cases as opposed to straight cases, showing a marked improvement over the Initial KB after 50 percent of cases are seen. The error rates for both algorithms are much closer to those in the Initial KB and suggests that this is a feasible approach to reducing repetition from manually built KBS that are incomplete and where a sufficiently large and complete set of cases is not yet available for machine learning.

Percent of Cases used to learn	Initial KB #rules	Initial KB %error	Cases only #rules		Cases only %error		Cleaned up Cases #rules		Cleaned up Cases %error	
			Induct	RST	Induct	RST	Induct	RST	Induct	RST
2.5	88	11.23	34	22	11.69	23.25	45	51	10.13	13.41
5.0	138	9.43	59	35	8.33	20.18	72	83	8.47	10.68
15.0	295	4.89	128	117	5.73	8.96	156	172	6.73	6.12
25.0	402	3.39	171	165	4.34	5.00	203	241	5.22	4.19
50.0	624	1.99	268	226	2.42	4.12	318	343	3.48	1.69
75.0	796	1.26	296	281	1.96	0.86	363	440	2.01	0.56
100.0	958	0.86	358	328	0.00	0.00	428	503	0.05	0.11

Table 4.6: Compaction on the Garvan Data using unmodifed Cases rather than Cornerstone Cases.
Columns four and five give the results produced from using the number of cases indicated in the first column and the straight Induct/RDR or RST algorithms. Columns six and seven show the number of rules and percentage of errors, respectively, produced by Vinduct and RS/RDR when the number of cases in column one are cleaned up using the rules in column two as input. The final row is included to show that when all cases are used for testing only the ML algorithms have no errors. In the case of cleaned up cases this is because the cases have been altered and in the case of the initial KBS manually built KBS may have errors as rules are only developed for the cornerstone cases and misclassification of other cases may have been missed.

4.4 Discussion of the Repetition Experiment Results

The results shown in Tables 4.4 and 4.5 using Vinduct and RS/RDR on the cornerstone cases show a decrease in the size of the knowledge base, which was accompanied by a significant decrease in performance of the system in terms of error rates between the Initial KB and all compacted KB, whether derived from straight cornerstone cases or cornerstone cases that had been filtered using one of the cleanup variations. More promising results are shown in Table 4.6 and Figures 4.9(a) and 4.9(b) where, for the Induct algorithm, the combined use of cases and rules to cleanup the original dataset obtains similar results to those achieved using ML alone on the same set of cases, producing a knowledge base roughly half the size of the original RDR knowledge base and the error rate for a fully matured KB went up by roughly 1%, from 1% to 2%. These results show both ML algorithms produce more compact rule sets. Induct performed similarly to the Initial KB for all percentages of cases seen. Error rates for RS/RDR were worse for both cases and cleaned up cases initially but after 25 percent of cases are used for training the error rates are lower than Vinduct on the cleaned up cases and both ML algorithms on straight cases. When 50 percent of cases are used for training RST/RDR shows a substantial improvement in the reduction of errors, which even exceeds the performance of the Initial KB.

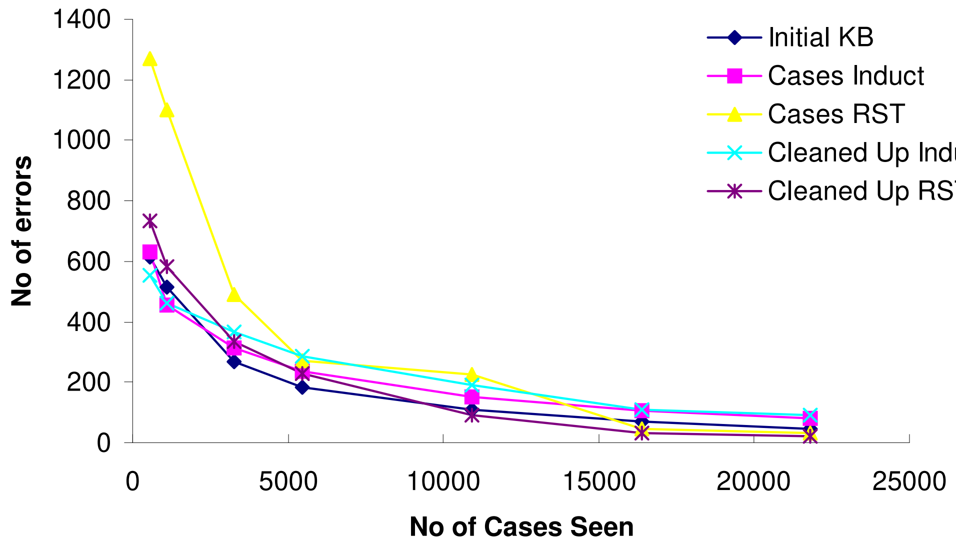


Figure 4.9(a): The Number of Errors for Garvan using Cases

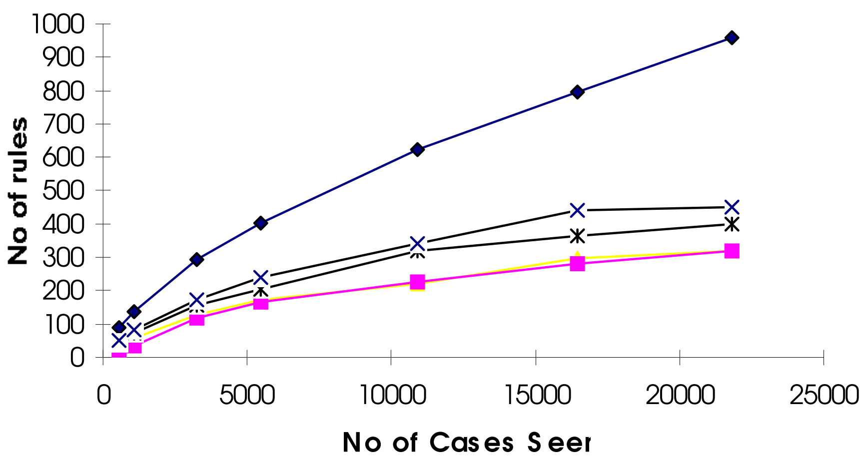


Figure 4.9(b): The Number of Rules for Garvan using Cases

The high accuracy levels and small KB size produced by ML, support the use of ML from cases for the population of knowledge bases and questions the need for the extra filtering step involved in either cleanup methods. However, as previously noted, cases are not always available and they may be unintelligible to an expert. As pointed out, RS/RDR's error rates were lower than the ML algorithms when only 25 per cent of cases have been seen. The overall improved results using all cases seen so far rather than just cornerstone cases supports Brian Gaines suggestion (personal communication) that all cases, not just cornerstones, seen by a knowledge base should be kept.

In the RS/RDR technique the original RDR KBS was being converted first into decision table format for input into the RSL routine. After rules (reducts) were produced they were being reconverted back into the RDR exception structure. It was a concern with the RS/RDR technique that the conversion from an RDR tree-structure to a flat structure and back again might alter the knowledge represented by the tree**.** Colomb (1993) has shown that conversion from a non-singulary decision tree, such as those produced by RDR, is equivalent to an unambiguous decision table. Various techniques were used to verify the conversion processes in this study. The new rules were evaluated for accuracy against the test set while flat (in decision table format) using one routine and again after conversion into RDR format. The results were within half a percent of one another, which given the large error rates using cornerstone cases, was not of great concern. Also each set of rules were evaluated against the test set as well as against the cornerstone cases in the Initial KB. Rules generated by RS/RDR and Vinduct from Cleanup Method 1 and the cornerstone cases resulted in 100% accuracy. Rules generated from the flat rule technique, only performed for RS/RDR, did not achieve high accuracy on the original cornerstone cases. This is expected since the flat rule method does not account for the knowledge in the false branches. The rules generated by Cleanup Method 2 using Vinduct covered all cornerstone cases, but this was not the case with RS/RDR and it has been found that proper handling of negation would require changes to the way reducts are generated (Sienkiewicz, personal communication), thus altering the original RST.

Another concern was that the structure of the new KB produced by RS/RDR was flat, with each new rule being added to the false branch of the previous rule. Two heuristics were applied to try to change this by adding more general rules at the top. The first heuristic was to add rules which covered the largest number of objects. If more than one rule covered the same number of objects a second heuristic was used which added the rules with the smallest number of conditions/clauses first. While these heuristics changed the order of the rules added, they did not appear to have much impact on the flat structure. An alternative not explored in these experiments is to use the Ghidora algorithm (Catlett 1992) which converts knowledge in a decision tree to RDR format.

An importance ranking (IR) is computed which gives the "expected benefit of the rule to the accuracy of the ruleset" (Catlett 1992, p.162). This IR measure is used to determine the order in which rules should be added.

Handling of the false branches has been noted as a problem in determining the most appropriate clean-up algorithm to use. A similar concern is how to handle negation. Cases usually have one or no value for each attribute. There was considerable effort required to both Vinduct and RS/RDR to deal with negation, which can be seen as the situation where all but one value can match. One attempt at avoiding making any changes to the RSL routines was to expand the negation. In the case where only two values were possible, such as true or false, if the rule condition said NOT FALSE the value in the case was taken, which would be TRUE. This is effectively the same as Cleanup method 1. Where more than one value was possible, a case was generated for each allowed value. This means duplicating each object n-2 times where n is the number of possible values. This will result in n-1 versions of the same object if only one attribute has a negated value. There are 32 condition attributes in the Garvan data. Thirteen attributes take more than two values so the inflation becomes excessive. In an Initial KB of 100 rules, 162,120 cases were generated. The memory needed for the RST tables were prohibitive and any further investigations down that path were stopped.

The large number of errors prompted investigation of the cause of these errors. It was frequently found that a small number of rules accounted for the vast majority of the errors. It seemed possible that if a few rules were removed then the error rate may drastically reduce. To test this hypothesis, a small experiment was conducted on a rule base that had one particular rule responsible for over 4000 errors. In addition, the conclusion for this rule was diagnosis 28, which was known to be a diagnosis that was difficult to accurately predict due to the large number of situations in which it could occur. Since many of the incorrectly classified cases belonged to the default class, the offending rule was removed. The result was that another rule with the same conclusion then had a similar error rate. When this rule was also removed another rule became the major cause of incorrect conclusions. This time the conclusion was diagnosis 36, another difficult classification. At this point the exercise was abandoned as the approach did not seem to solve the original high error rate problem. Another approach considered, but not tested, was the modification of the compacted rule base by an expert who could provide the correct conclusion and rule refinements. This may still result in a smaller size KB with the original or better error rates. There does not seem any easy solution. It is likely that the offending rules would require numerous patches for so many misclassified cases.

It appears that these troublesome conclusions (28, 36 and 51) are used by many rules and that the rules are quite different within each class. This means that there are many ways in which conclusion 28 can be given and this makes the task of the machine learner more difficult. To exacerbate this problem, Induct determines the most common conclusion to find rules for first. Since these conclusions are the most common these poor rules will be added to the RDR KBS first and may produce a poor structure for the KBS. RST begins to find rules in the order in which classifications are found in cases and may have faired better with cleaned up cases than without since some of the irrelevant attributes were removed beforehand. RST is primarily concerned with being able to distinguish between classifications and sometimes strips out attributes that are actually important but with the given set of cases were not necessary for differentiation.

The results produced by the two ML algorithms were similar for the three domains, however each algorithm revealed various strengths and weaknesses. Table 4.6 shows that RS/RDR produced larger but more accurate KB from the cornerstone cases than Induct after 15% of cases are seen. The algorithm used by Induct uses probabilities and generalisation to determine rules for the most common classification and tries to find the attribute value pair that identifies cases with this classification. Once this rule has been added the cases are removed and new rules do not consider whether previously classified cases would now be misclassified. This problem is exacerbated by the way missing values are handled because a case may be treated as being covered by a rule (a false positive as shown in Figure 4.7) because a value was missing when in fact if the value were known the case would not be covered and a new rule would be needed to cover it. RS/RDR produces more specific rules that are generated from comparison of each case with every other case with the same conclusion. Only after evaluation of all cases are rules generated. Also as stated in Section 4.2.3, RST does not allow assumed values to missing values to contribute to the rule developed. Therefore, as confirmed by the figures in this study, the initial error rate with Induct is lower than with RST, but as more cleaned-up cases are used for training RST produces rules with less errors. Induct produces rules of around 2.5[26] clauses, which is bigger than the 1.3 clauses for manually built RDR but closer than for RS/RDR which averaged 3.3 conditions per rule. It is also interesting to note that when similarity indices were computed between sets of rules using the nearest neighbour algorithm described in Section 3.3.2.1 it was found that the rules in the KBs produced using rough sets were closer to the Initial KB than the rules in the KBs produced using Induct.

---

[26] An average of 3.8 clauses per rule is reported in Catlett (1992) from Gaines (1989a).

One reason the approach with cornerstone cases may have failed is that the sample size needed to be larger. It has been argued (Sammut 1996, personal communication) that the training set will never be large enough because not enough cases are kept. A similar view was given by one of the reviewers of Richards, Chellen and Compton (1996) who said that the study was destined to fail because it was founded on the false assumption that the cornerstone cases held some special meaning or value. As noted in the comparison between PCP and RDR no such view of cornerstone cases is held. While the cases which prompt a new rule to be added may be representative of a class of cases it is realised that there are possibly many other cases that could have been equally representative with different attributes and that it was essential to remove irrelevant attributes that do not contribute to classification. This is why the filtering process with the various clean up methods was tried. Just as it was not appropriate to use cornerstone cases alone it was also not possible to use the rules alone for reducing the size of the KB. The problem in a single classification RDR KB or any method that uses “if-not” or false branches is that we can not reconstruct the pathways without knowing what values caused the false path to be taken. We can not take the negation of all conditions on the false branch because some of the conditions may have been true. While all conditions in a rule must be true to fire we only need one condition to fail for the whole rule to fail. The combined use of rules and cases allows us to determine the actual values of attributes because we can take the value in the case on false branches. As noted in Section 3.3 the real problem remains, however, as we can not be sure that the failure is due to an irrelevant A-V pair or whether the failure is important in reaching that conclusion.

The greatly superior accuracy rates when using all available cases supports the criticisms against using cornerstone cases and favour the use of all available cases for compaction. When the reduced rules produced from either the straight cornerstone cases or cleaned up (using Cleanup 1) cornerstone cases were evaluated on the related set of cornerstone cases the accuracy was 100%. In contrast, the error rates were high when evaluated on the test set of 5457 unseen cases. When we are dealing with rules/cornerstone cases we have a much smaller training set of cases than when the whole dataset is used (using 75% of cases for training we had 796 cases using cornerstone cases or 16365 cases when using available cases). What seems even more relevant than the sample size is the need for the distribution of cases to be statistically appropriate when statistical methods such as Induct are used. As mentioned, there were a large number of cases in the Garvan KBS that were poorly classified and it is believed that this has affected the results achieved.

Another possibility for the poor results on the cornerstone cases is that the use of a KB produced by a simulated expert is not an appropriate starting point. It may be that the simulated expert does not produce repetition of knowledge to the extent or in the manner that a human expert would and that attempting to remove repetition is a futile exercise. Use of a system built by a human expert has been considered, but is difficult due to the presence of time course data, real data or incompatible file formats of existing manually built RDR KBS. In retrospect, the problems that were encountered with the experiments possibly could have been avoided or at least miminised by using multiple classification RDR because there are no ambiguous false branches in MCRDR.

The poor results obtained when straight or cleaned up cornerstone cases were used for learning implies that the structure of an RDR KB and the storing of cornerstone cases capture knowledge which is not easily re-represented without loss of predictive power. If the knowledge is to be reused it would seem desirable to make the knowledge more perspicuous and that this would involve removal of redundant knowledge and appropriate handling of the knowledge in false branches. The varying results obtained when using case values found in the conditions of true branches (Flat True rules), case values from attributes in true branches and false branches (Cleanup 1) and case values from attributes in true branches and the negation of values in rules that don't match on the false branches (Cleanup 2) prompt the questions: what knowledge is really being represented by the false branch and can the pathways be flattened without loss of information ? Cleanup 1 yielded better results than Cleanup Method 2, where the number of errors rose and in some cases the number of rules generated was greater than with the Initial KB. The increase in errors is unexplained, however the increase in the number of rules is possible when we expand the negation by creating multiple cases to cover the rule that has negated values. It is interesting that when only the true branches were used for filtering cases the error rates were much lower on the larger test set than on the cornerstone cases associated with those rules and the results with RS/RDR are better than with either Clean Up method that tried to capture the knowledge in the false branches when run on the cornerstone cases. It appears that the impact of the false branches decreases as the size of the dataset increases.

Two important concerns not addressed in this work are the question of when the reorganisation should be performed and how to proceed after the reorganisation has been done. The issue of when will depend largely on the individual KBS and the amount of redundancy that exists in it. As the whole process described for both RST and Induct is automated it would be possible to periodically determine the extent to which the KBS could be reduced. If the gain is considered worthwhile it could be performed. It is difficult to quantify the answer to “when to do” when the value of and answer to “if do” is still an open question. The second issue of how to proceed after reorganisation is of greater significance. The use of cornerstone cases for validation and assistance with KA is an integral part of the manual development of RDR. As new rules are manually added, the new case can be stored for future use but it would be necessary to develop a modified KA technique if the misclassified rule did not have an associated case. This issue has not been explored but would need to be resolved before reorganisation of RDR KBS became acceptable practice.

The results of these experiments support the removal of repetition using a ML algorithm such as Induct or RST on all available cases but the decision of whether to clean up the cases first depends on the number of available cases and their coverage of the domain. The important question that remained from these experiments was whether the reduced rules were *useful* and in what way. An evaluation of the comprehensibility of the compacted KBSs was performed but abandoned after it appeared that most of the examples chosen for comparison in each KBS (original, straight cases - Induct, straight cases - RST, cleaned up cases - Vinduct, cleaned up cases - RST) appeared to be unusual cases and the explanation provided by the conditions on the rule pathway were difficult to understand for all KBS. Again the problem was related to the anomalous classifications given to a large number of different cases which have distorted the results. To some extent this outcome is disappointing as a strong conclusion was desirable. The main reason that further investigation has not been pursued is that many of the problems were seen to be possibly related to the single classification RDR structure and research efforts were moving into the stronger MCRDR representation. In keeping with this movement to MCRDR, which is also the direction the rest of the work in this thesis has taken, a proposal for the removal of repetition from MCRDR KBS is now offered but the actual investigation is left to future research.

4.5 A Proposal for the Removal of Repetition in MCRDR KBS.

As noted in the last section the value of removing repetition from single classification RDR to support reuse of the knowledge for explanation activities was inconclusive and has not been pursued further as research interests have moved to the MCRDR representation. In this section a brief proposal for the removal of repetition from an MCRDR KBS is given. Since at this point in time there are few manually built MCRDR KBS available for testing, particularly large KBS that would have enough repetition to consider reorganisation, this section is a proposal of what could be done rather than a description of what has been done. It also appears that the MCRDR structure may be undergoing further change in the future so that experiments to restructure an MCRDR to have less repetition would be premature until the representation has stabilised.

The removal of repetition can be performed as a three step process.

1. Expand the rule pathways
2. Apply ML algorithm to remove redundancies and restructure KBS
3. Sort list of rules into MCRDR format.

The first step is to expand the rule pathways by taking each rule with its conditions and conclusion and adding all conditions of parent rules that can be traced back to the first default rule. This is the same process that is used in the following chapter to convert an MCRDR KBS into a crosstable for use by the FCA algorithms.

The second step is to use some algorithm to remove or modify rule pathways according to the type of repetition to be removed. For example we would definitely want to remove rules that had identical conclusions and conditions but we may or may not choose to remove rules that reach the same conclusion but the set of conditions in one rule are subsumed by the set of conditions in another rule. Depending on the types of redundancies to be removed we can select the appropriate algorithm. For this purpose we could again use Induct, RST or the CUT95 algorithms (Scheffer 1996) to reorganise our KBS. Without performing any experiments using these various algorithms, apart from the small experiment with RST described in section 4.7.2, it is not possible to say whether the resultant KBS would meet the requirement that KBS accuracy in classifying cases must not diminish.

After reorganisation of the knowledge it would be possible to use the flat structure to which exception rules could be added as new cases are seen. However, it may be desirable to convert the flat structure into an MCRDR to gain the potential benefits of exception structures such as the naturalness and comprehensibility of the representation by human experts; the support they offer for incremental changes; the use of context and the maintainability of the knowledge due to the localisation of changes (Gaines 1991a, Li 1991). To achieve this the algorithm by Scheffer (personal communication) outlined below is proposed to reformat the flat rules back into MCRDR format.

An alternative and possibly safer (less interference with the existing KBS) approach to these three steps is to define the types of redundancy to be removed and to look for those anomalies and remove them. For example, we can obviously remove any rules that are complete duplicates (same set of conditions and conclusion). We can also remove any subsumed rules which have a larger set of conditions but include all the conditions and have the same conclusion as another rule. Rules with unnecessary premise conditions can be modified so that the redundant condition is dropped and then the duplicate rule can be removed. By targeting particular anomalies and fixing or removing those, rather than a complete reorganisation of the KBS using a ML algorithm, we should be able to maintain our current accuracy levels, save most of the history of changes stored in the KBS structure and keep the relationship between rules and cases that is not supported in the ML approach. We can even avoid the first and third steps of flattening and then reformatting the RDR KBS if the clean-up strategy used updates the various parent and child links to reflect the changes. For example, if a rule which is part of another rule's pathway is dropped the rule left must be modified to include the dropped rule's conditions unless they are redundant. A maintenance strategy to ensure that there is no loss in consistency, completeness and accuracy of the MCRDR KBS is not proposed here but the alternative of directly updating the KBS is worth considering especially since it should be possible to retain the links between rules and cornerstone cases that already existed.

As noted in Section 4.4, it was found in the single classification RDR repetition studies that the KBS produced from the machine learners tended to be flat structures with all rules except the default rule being a leaf node on the tree. The algorithm shown below is designed to produce KBS that take advantage of the MCRDR exception structure. It is speculated that this algorithm could be used as a way of producing MCRDR format KBS from single classification RDR. The main problem anticipated with this suggestion relates to the problem of how best to expand the rules in a single classification RDR KBS as the best interpretation of the false branches is not clear. Whether single or multiple classification RDR KBS are used as input the following algorithm could be employed for the output of MCRDR KBS.

4.5.1 Sorting Lists of Rules into MCRDRs

After steps One and Two have been performed to remove repetitious and redundant knowledge the following algorithm is designed to produce MCRDR structured KBS.
The expected input to and desired output from step Three is:

Input- A list of rules with possible overlap. That is, some rules premises may be subsets of the premises of other rules. Figure 4.10 shows the possible input.
Output - An MCRDR structure such that every rule premise which is covered by another rule is explicitly marked as an exception. Figures 4.11 and 4.12 show the intermediate and final output, respectively.

For example:

Input: $a \rightarrow +$
$ab \rightarrow -$
$b \rightarrow +$
$abc \rightarrow +$

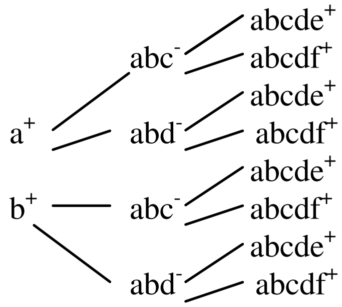
Output: a^{+} —— ab^{-} —— abc^{+}
b^{+} —— ab^{-} —— abc^{+}

As can be seen above ab is subsumed by a and by b and abc is subsumed by a, b and ab. In the worst case: if we have two rules on the first level, two rules on the next level and so on, all rules of level i cover all rules of level $i+1$, $\rightarrow$ n/2 level $\rightarrow$ $2^{n/2}$ rules in MCRDR.

For example:

Input: a^+
b^+
abc^-
abd^-
$abcde^+$
$abcdf^+$

Output:



The following sorting list algorithm is described where:

"more-than-general-graph"- see figure 4.10
V= vertices and E= edges.
input degree zero = there are no input edges.
"dangling edges" = edges that do not connect to a vertice.

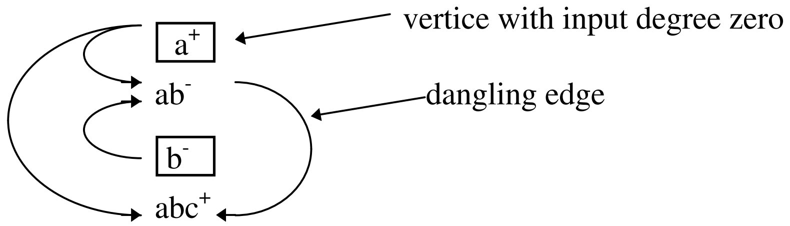


Figure 4.10: More-general-than-graph

Algorithm:
  Set up "more-general-than-graph" (V, E)
  All nodes with an input-degree of zero become rules, ie the graph vertices.
  Do while no more "dangling edges"
    look at only the subset of nodes V′ s.t. (V,V′) ∈ E
    E′ = E - "dangling edges"
    All nodes with input-degree zero are the rules (exceptions to V)
  End While

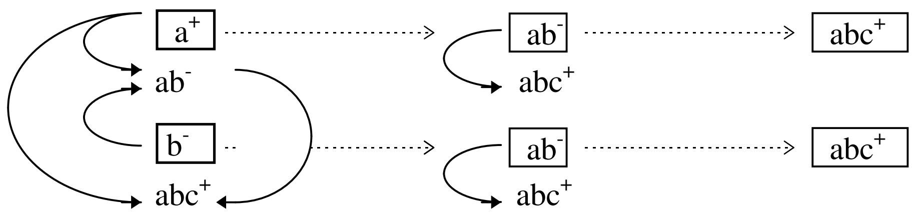


Figure 4.11: Final expanded "more-general-than-graph"

$$\boxed{a^{+}} \dashrightarrow \boxed{ab^{-}} \dashrightarrow \boxed{abc^{+}}$$

$$\boxed{b^{-}} \dashrightarrow \boxed{ab^{-}} \dashrightarrow \boxed{abc^{+}}$$

Figure 4.12: The final MCRDR Structure

The three-step process and sorting algorithm described above have not been implemented. As was found in the removal of repetition from single classification RDR KBS it is envisaged that modifications and alternative methods will need to be considered. This brief proposal has been offered as suggested future work that will best be tested when manually built MCRDR KBS of reasonable size are available and when the optimum MCRDR structure has been determined. Both of these events will be realised in the not too distant future.

4.6 Summary of the Repetition Studies

Some experiments using the Induct and Rough Set algorithms to remove repetition were described in sections 4.1 to 4.4. The results using cornerstone cases were poor but the results using all available cases (this does not mean the full dataset but all cases seen at various stages of KBS maturation) were favourable, producing KBS at least half the size with error rates the same or better than the original error rates once 50 per cent of cases were seen using RST and increased error rates of about 1 per cent using Induct. The benefits of removing repetition for explanation purposes, which was the motivation for the studies, remains an open question. The repetition experiments concerned single classification RDR and the mixed results can be attributed to some of the ambiguity of the false branches together with certain features of the algorithms used. As research interests have moved to the newer MCRDR representation further experimentation with alternative algorithms has not been pursued but a brief proposal for the handling of repetition of knowledge in MCRDR was given in Section 4.5. The proposal was not implemented as it was felt that more MCRDR KBS need to be developed first and changes to the MCRDR are likely in the near future.

This does not mean that the experiments were fruitless. The experiments raised many issues concerning the single classification structure and the ML algorithms that were used. This discussion was given in Section 4.4. The investigation of RST has also led to other benefits. As stated in the opening section of this chapter, two of the benefits of inductive learning are the removal of redundancy and inconsistency and the detection of missing knowledge. The repetition studies have been concerned with the redundancy issue. However, while exploring the potential use of RST for knowledge base compaction and reuse it was found that the routines provided in the RSL could be used to identify and remove various other verification anomalies. An algorithm was developed and applied to a subset of the rule base for AusVit, the Australian Viticulture Expert System (Richards, Gambetta and Compton 1996). This algorithm has also been applied to the Garvan, TicTacToe and Chess datasets to verify consistency and remove redundancy. In the following section we look at the use of RST for verification of rule-based systems in general and in section 4.7.2 we look at a modified approach to support the verification of MCRDR KBS.

4.7 Using Rough Set Theory for Off-line Verification[27]

In Section 4.7.1 we review work that has been done using RST for verification of traditional rule-based systems and an extension of that approach to handle MCRDR in Section 4.7.2. The process is referred to as off-line verification because it is a batch process in contrast to the existing verification method, as described in Section 3.2.3 which occurs incrementally and online as the rule base is being used.

4.7.1 A General Approach to Verification of Rule-Based Systems.

The work by Colomb and Sienkiewicz (1995) on using rough set theory to reduce the inflation problem when converting from a decision tree or propositional KBS into a decision table using rough set theory inspired the use of RST for finding the key concepts and functional dependencies between attributes in a rule-base. During these investigations it appeared that RST was eminently suitable for finding verification anomalies. In this section we consider a number of common verification anomalies: redundancy, inconsistency (ambivalence), imcompleteness (deficiency) and circularity and how they are addressed by RST.

RST defines redundancy in the same way as Preece, Shinghal and Batarekh (1992). When dealing with knowledge in decision table format, as in Table 4.4, if we remove a column, row or value without affecting the hypotheses derived then it can be removed. This concept can also be employed to calculate the significance of attributes. If the attribute is dispensable then that attribute is not significant. When dealing with a set of rules one would expect that all attributes would contribute to at least one outcome so the measure determined is one of comparative significance. If discarding one attribute changes the classifications of many objects then it is more significant than another

[27] Much of Section 4.7 is taken from the paper "Richards, D ., Gambetta, W. and Compton, C (1996) Using Rough Set Theory to Verify Production Rules and Support Reuse *Proceedings of the Verification, Validation and Refinement of KBS Workshop, PRICAI'96* 26-30 August 1996, Cairns, Australia, Griffith University, 57-66".

attribute which only alters a few objects when it is dropped. In rough set terminology, which was introduced in Section 4.1.2, the objects move from the positive to the boundary region if the attribute is dropped.

The issue of ambivalence is addressed by computation of the degree of dependency κ which is simply the number of cases in the positive region (card POSp(Q)) divided by the number of all cases (card(U)) (see formula 4.4). The dependency coefficient says that the set of attributes Q, the conclusion, depends in degree κ on the set of attributes P, the conditions, for the decision algorithm PQ. If $\kappa = 1$ the knowledge base is consistent. If the degree of dependency is greater than zero but less than one we can determine what cases are outside of the positive region and investigate why they are inconsistent.

$$\kappa = \text{card } POS_P(Q) \,/\, \text{card }(U) \qquad (4.4)$$

note: card = cardinal number of, POS = positive region, U = universe of objects.

Deficiency is seen as the situation where no hypothesis can be inferred. The quality of classification can be tested in RST with the classification coefficient (formula 4.5):

$$\gamma_R(X) = \text{card}(\underline{R}X \cup \underline{R}\text{-}X) \,/\, \text{card}(U) \qquad (4.5)$$

This formula takes the number of objects in the union of the lower approximation of a set and its complement and divides this by the number of all objects in the domain. This is checking whether all objects that are consistent can be classified. If we get a coefficient of 1 it means that all objects are consistent and can be classified.

Rough set theory was applied to the work by Richards(1994) who adapted an existing consultation-style ES so that it could be used in a "what-if" decision support system style. Richards found that determining the significance of a variable was important in allowing reuse of the knowledge. In that study, the AusVit[28] rule-base for the disease Botrytis was examined and decision tables for the final conclusion s*pray need* and the intermediate conclusion, *disease risk*, were created. For simplicity and to save space, the discussion that follows will be restricted to the *spray need* outcome, Table 4.7 shows the original 18 rules.

[28]AusVit is the Australian Viticulture ES that assists grape growers to manage various pests and diseases. The system won the Decision Support for Farmers Award at the 1996 Sydney Royal Easter Show.

Rule#	Stage	Dis Risk	Spr Active	Grape Use	Summer Wthr	Spr Need
1	6-14	not nil	not	EW/LW	---	high
2	10-14	not nil	not	ET/LT	---	high
3	16-17	---	not	not dried	not warm & dry	high
4	18-20	not nil	not	not dried	not warm & dry	high
5	18-21	high	not	LW/ET/LT	warm & dry	high
6	6-9	not nil	not	ET/LT/Dried	---	low
7	10-14	not nil	not	dried	---	low
8	16-17	---	not	---	warm & dry	low
9	16-17	---	not	dried	not warm & dry	low
10	18-20	not nil	not	dried	not warm & dry	low
11	18-20	high	not	EW/dried	warm & dry	low
12	18-20	low	not	---	warm & dry	low
13	1-5	---	---	---	---	nil
14	22-24	---	---	---	---	nil
15	6-14	nil	not	---	---	nil
16	6-21	---	active	---	---	nil
17	15-21	--	not	---	---	nil
18	18-20	nil	not	---	---	nil

Table 4.7: The rules for the final conclusion Spray Need from the Ausvit rule base. Each row in the table corresponds to a rule in the KB. "---" means "don't care" and "/" is an OR condition.

Convert KBS rules into rough set format (SYS)
While reducts (rules) of SYS show redundancies exist
    Determine why attribute, object or value was removed
    Fix objects
End-While
Calculate
While dependency coefficent $\kappa < 1$
    List all objects not belonging to the positive region and the object that they conflict with.
    Fix objects
End-While
While classification coefficient $\gamma < 1$
    Find objects not in the lower approximation or it's complement
    Fix objects
End-While
Output new rules (These may be converted back into the original format or used as tables).

Fix objects.
If coding error, fix.
Else If error due to missing values
        Initialise A-V check table
        Get set of objects that match with the object being investigated
        While there are more objects in the set
            read an object
            update A-V check table with a found flag for each non-missing value
        End-While
        Build one or more new objects from the original object using the AV check table to cover those values not covered by other objects.
Update SYS with new objects, modified objects and remove objects in error.

Figure 4.13: An algorithm of the verification process using rough sets. There are three main phases each contained within a while loop. Procedure Fix Objects is expanded to provide more detail. The word rule refers to the original and output rule sets. The word object refers to the cases being used by the RSL which represent the rules.

4.7.1.1 Preparation of the Rules

The rules in table 4.7 needed further conversion into an attribute-value (A-V) matrix so that it could be used by the rough set library (RSL). This meant removing not conditions and exploding the decision table to include one rule for each situation. For example, if the rule had a condition NOT(DiseaseRisk = nil) then two rules were created from the original rule one with the value high and another with the value low for that attribute. This resulted in the growth of the original decision table from 18 rules to an inflated table with 59 objects. This approach to negation is in keeping with Pawlak's (1991, p.87 axiom 3) where the property is considered to have all but the negated value. The RSL routines were used repeatedly at each of the three phases. An algorithm is presented in Figure 4.13, with more detail provided in the following sections.

4.7.1.2 Phase One: Removal of Redundancies

Through the generation of reducts, redundancy in the 59 objects created above was removed and resulted in 40 reduced rules. Surprisingly from a domain perpective, the reducts generated did not include the attribute *spray active.* This indicates that the attribute can be dropped without loss of classificatory power or introduction of inconsistency. It seemed reasonable that all attributes would be used for at least one rule when we are using rules for input. In fact one would expect the *spray active* attribute to be significant when the conclusion being pursued is the *spray need.* On further investigation it was found that the original structure of the knowledge contained overlapping values for *growth stages*. It was necessary to change the model of *growth stages* being used to remove the subsumption errors that were occurring. This resulted in 75 objects being created to represent the modified rule base. One assumes that one of the best outcomes of finding verification problems is to improve the KBS in other ways and the detection of a faulty *growth stage* model is an example of such an outcome.

4.7.1.3 Phase Two: Removal of Inconsistencies

The 75 objects output from phase one were processed again on the rough set routines. In the second phase 57 rules were generated and the *spray active* attribute now had the second highest significance coefficient. However, we now had a dependency coefficient of 0.42667. It was therefore apparent that at least one rule was inconsistent. The system identified Rule 17 to be in conflict with rules 3-5 and 8-12. Although there was only three objects representing the knowledge in rule 17 that were causing the problem, the dependency coefficient indicated that there were only 28 consistent objects out of the 75 and therefore 47 inconsistencies. A further indicator of the inconsistency of the knowledge was the large variation, from 46% to 96% depending on the inferencing strategy used, in the "number of good decisions" reported when the reduced rules were used to classify the test set (in this case the training set was used as the test set) .

The problem that emerged was the original KBS had depended on rule ordering to control the inference. The cause of the low dependency coefficient was the improper use of "don't care" values in rules that were meant to act as defaults. This is yet another example of Clancey's (1992) contention that separation of inference engine and knowledge base is a fallacy and that control knowledge is often embedded in the domain model. After fixing the objects that covered Rule 17 (using the Fix Objects procedure in Figure 4.13), the final A-V matrix had 74 objects which reduced to 57 rules and the "number of good decisions" was 100% for all inferencing methods.

4.7.1.4 Phase Three: Detection of Deficiencies

The final step was to find any deficiencies. The completeness of knowledge can be measured by the ability to classify objects. When this value is computed using the 57 reduced rules we get a classification coefficient of 1. The method for computing the classification coefficient is given in formula two in Section 2. For interest all possible cases (720) were generated and the 57 reduced rules were used to classify each object, again confirming that the knowledge was complete.

4.7.1.5 Further Research

A few issues are discussed below that were not covered in this study and require further investigation. Some suggestions are offered for each issue.

Finding circularity anomalies.

A more comprehensive approach would be to expand the rule base into a larger table which included a column for each attribute used in the rule base covering both the spray need and disease risk outcomes[29]. By doing this we would also address the problem of circularity, because all pathways would clarified.

Note that if the rule base being converted was an RDR KBS then it may have been possible to keep the number of cases generated to the same size as the rule base by using the cornerstone case stored with each rule as input into the RSL. However, cases are more specific than rules and the use of cases alone would probably result in an overfit of

[29] Colomb (1993) has shown that knowledge in the form of decision tables or decision trees can be viewed as propositional systems and can be converted from one representation into another. He provides algorithms for such conversions using the concept of equivalence with respect to the number of cases.

the data and higher error rates and a verification program would be needed to ensure accuracy levels were maintained. A combination of rules and cases such as described earlier in this chapter and in the work of Colomb and Chung (1995) may be appropriate.

Dependencies between attributes.

An important factor that affects the correctness of values is the existence of dependencies between attributes. The identification of functional dependencies can help us with system verification through the minimisation of redundancies, validation of new data and the inference of missing data (Keen and Rajasekar 1993). The AusVit rules did not contain dependencies between attributes. This is probably atypical of most domains. This issue is better addressed by the work using FCA discussed in Chapter Five were we are able to build a subsumption hierarchy from the rules which shows which A-V pairs tend to occur together.

4.7.1.6 Discussion of RST for Verification

It had originally been hoped that RST would be able to provide useful significance coefficients when a rule base is used as input. However, after running the rough set routines on various KBS the results were inconclusive. It is now painfully obvious that the number of times a case is seen is going to impact on the significance coefficient. Depending on how the rules are structured and the domain, some rule bases may give results that appear reasonable. It appears that if we want to compute the significance of an attribute so that the knowledge may be later understood for reuse it is necessary to keep some kind of statistics about rule usage and accuracy. Such statistics are not commonly kept. However, Edwards' (1996) *credentials* system (described in Section 3.1.3) stored statistics on the number of times a rule fires and is deemed correct or false. It is anticipated that these statistics could be used directly or in conjunction with another algorithm as a measure of importance.

While the significance coefficient was altered by the number of cases seen, the rules built were not affected by the removal of repetition since only one of each example is used in determining reducts (rules). This is not the case with most ML techniques such as C4.5 or Induct. The data were also randomised with no change in results using RST. The order in which cases are seen tends to affect ML algorithms that use some sort of covering algorithm to generate rules. The strength of RST lies in how it partitions knowledge and uses these partitions to develop rules of classification with incomplete or inconsistent data. The weaknesses of RST are its excessive memory requirements and running time.

The major contribution of this study is in demonstrating the usefulness of employing rough sets as a structured way of verifying a KB. Redundancies are identified through calculation of value reducts. Using the dependency coefficient and checking the positive region we have a means of determining when and where inconsistencies have been introduced. Using the classification coefficient we could determine the completeness or deficiency of the knowledge. The major drawback of the approach is the increase in rule base size, but size is seen as a problem due to the side-effects that are introduced when we try to maintain a large KBS. The rules that are output may be used directly in table form, which has been shown to give an order of magnitude increase in speed over propositional ES (Colomb 1990), or the rules identified as being erroneous can be modified in the original rule base. The latter approach should result in a new set of rules which is *similar* to the original set of rules and may be more comprehensible to the expert than a large decision table. A third alternative is to convert the table back into propositional form. Further work needs to be performed on larger decision tables of rules to see what problems arise with repeated conversions between representations.

4.7.1.8 Comparison with other V&V Anomaly Detection Techniques.

The approach used by RST is comparable to other verification tools. RST's Matrix A (the attribute-value table) is similar to the tables used in Expert System Checker (ESC) (Cragun and Steudel 1987) and Rule Checker Program (RCP) (Suwa, Scott and Shortliffe 1982). RCP checks for redundancy and ambivalence by comparing each rule against the others, which can be performed in n(n-1)/2 rule comparisons. Testing for deficiency using RCP requires the testing of each possible combination. The method is criticised for its high computational overheads and its limited checking features which are mainly undetected because inference chains are not examined. While RST also suffers from the high overheads associated with pair-wise checking, RST's use of the positive region in the classification coefficient is a less computationally expensive approach to determining deficiency.

The CHECK program (Perkins, Laffey and Pecora 1989) builds on the work begun by RCP and is able to detect redundant rules, conflicting rules, subsumed rules, circular rules, unreferenced attribute values, illegal attribute values/dead end goals and unreachable conclusions. CHECK creates the "if-if", "then-then" and "then-if" tables with a row and column in each table for each rule. Comparisons are made in the "if-if" and "then-then" table to see if the intersection of any rules are the "same", "superset", "subset" or "conflict". The "then-if" table in CHECK is used to detect when the antecedent of one rule is the consequent of another and in this way circular inference chains can be detected. The range of anomalies detected by CHECK is still limited and there is little empirical evidence of CHECK's usefulness. The use of "if-if" and "then-

then" tables is similar to the use of the discernability matrix, Matrix D, in RST. Matrix D performs comparisons quicker than Matrix A but the generation of Matrix D can be time consuming. For the purposes of verification of a knowledge base lengthy batch processing may not be a major problem if it is assumed that most knowledge acquisition will occur at the beginning of a system's life. Once the system was in use verification could be performed after each maintenance episode.

Search space is a problem for many verification techniques. This is particularly so if dynamic verification is performed which takes into account the inference chain, such as in COVER (Preece, Shinghal and Batarekh 1992). To reduce the search space, the generation of environments in COVER is guided by two heuristics: the first uses the existing rules to restrict the possible combinations of attributes and the second heuristic orders the environments by size so that environments that contain the ones already found are not generated. RST could employ similar techniques to reduce the large search space and processing time that is needed for pair-wise comparisons. In particular, the second heuristic bears some resemblance in RST to the use of a reduced discernability Matrix X, which subsumes all oversets and repetitions in Matrix D. Matrix X can be created for selected objects and/or attributes to find cores and reducts. The efficiency of many of the rough set algorithms have been improved but these newer versions have not yet been implemented in the RSL (Sienkiewicz 1996, personal communication).

4.7.1.7 Summary of the General Verification Technique

This section has demonstrated the utility of using RST for verification of KBS. While RST is well known in the data mining area, applying RST to KBS, rather than datasets, had not previously been performed (Sienkiewicz and Pawlak, personal communication) with the exception of Colomb and Sienkiewicz (1995). The ability of RST to remove redundant and inconsistent data and its handling of uncertain and missing values appeared to make it eminently suitable for dealing with KBS that have typically many missing values compared to cases. The case study presented using the AusVit KBS shows that the technique is applicable to KBS that can be mapped into a decision table.

While much of what RST offers is achievable by other verification tools, and some also handle inference errors without the need for conversion, RST is a technique that deserves further attention due to the simplicity of the concepts and the easy availability and customisation of the routines supplied in the Rough Set Library (RSL). Pawlak (personal communication) has expressed interest in this work and its further development. There are also many possible uses of RST that go beyond the anomalies detected in this study. For example, it is possible to pass a particular conclusion to the rough set routines and generate the rules that derive that conclusion. This could be used to test that all possible conclusions were reachable.

As noted the technique described here applies to knowledge in decision table format and the example given was a standard rule base of If-Then rules being used by Level 5 Object and included intermediate rules and conclusions. The technique was also tested on the rule bases for Chess, Tic-Tac-Toe and Garvan (which were already available in decision table format) and was found to be a reliable verification technique for the detection of redundancy, imcompleteness or inconsistencies. The approach does not particularly consider single classification RDR systems due once again to the ambiguity of the false branches and the 'inflation' problem that occurs when the false branches are expanded to handle negation. However, the approach did seem suitable for MCRDR KBS but with some refinement. In the next section some modifications to the technique described above are given to extend the approach to handle MCRDR KBS. The benefit of using this approach for MCRDR is that we can evaluate that certain groups of classifications are consistent and we can remove redundancy in RDR.

4.7.2 The Off-Line Verification of MCRDR KBS.

The previous section considered how RST could be used for the verification of conventional rule-based systems. From the lessons learnt in the compaction studies it was apparent that single classification RDR would not map easily to the decision table format required by the rough set library due to the ambiguity of the false branches. MCRDR KBS, however, do map directly to a decision table format as the KBS can be viewed as pathways of exceptions which correspond to rows in a table. It therefore seemed possible to extend the techniques described for the verification of conventional rule-bases to MCRDR KBS. An experiment on a small MCRDR knowledge base is described in this section together with a description of the modifications needed to handle multiple classifications. This extension is described in Section 4.7.2.3.

4.7.2.1 Preparation of the Rules

As already described a necessary prerequisite for applying RST was the conversion of the rule base into a decision table structure. For rulebases that use intermediate conclusions this process is achievable by exploding each rule into its a lower level form where intermediate conclusions are replaced with the primitive conditions. In an MCRDR KBS the conversion process is much simpler. Each rule represents a rule pathway which may be treated as a row in a decision table with the rule conditions along the way forming the attribute value pairs in the table. Once the rules are in the rough set format (decision table) the iterative process described below and summarized in Figure 4.13 is performed. The process is described using a small (42 rules, 11 attributes and 31 conclusions) MCRDR KBS developed for the SISYPHUS III geology domain. Thus we begin with a decision table with 42 rows and 12 columns, the twelfth column contains the conclusion which is known as the decision attribute in rough set terminology.

4.7.2.2 Phase One: Removal of Redundancies

Through the generation of reducts, redundancy in the 42 objects in our decision table was removed and resulted in 40 reduced rules. These two rules were removed either because they were originally duplicates (with conditions occurring in different sequence but giving the same conclusion) or they became duplicates after dispensable conditions had been removed. The latter situation was the reason for the rules being dropped for the SISYPHUS III KBS. While only two rules were removed we had a substantial decrease in size of the rule base in terms of the number of conditions from a mean number of 4.0 conditions per rule to a mean number of 2.2 conditions per rule. The reduced KB was then evaluated against the cornerstone cases to ensure that each case gave the same set of original conclusions (remember we are dealing with multiple classification RDR). It was found that five of the fifteen cornerstone cases gave one or two extra conclusions. By making the rules more general there were more cases covered by a rule. Assuming that our original classifications were right, it was necessary to make some rules more specific to preclude the misclassified cases by adding the A-V pair (rule condition) that had been removed. If more than one condition had been removed then by looking at the case it could be determined which condition/s needed to be added. It is initially alarming that one third of cases were overclassified, but if we take into account that the cases had an average of 6.2 conclusions each and that for a total of 93 original conclusions we obtained 99 conclusions, we achieved 94% accuracy. To achieve 100% accuracy on the cornerstone cases four rules out of the 40 needed one extra condition added.

Before going on to look at the other anomalies, it is useful to consider what benefits or drawbacks may result from this process. On the positive side, three conditions (attribute-value pairs) had been removed from the KBS and the number of concepts derived from the rules using formal concept analysis reduced from 78 to 56, signifying a 29% drop in rule-base complexity. By generalizing the rules, that is making them less specific by removing conditions, we should also develop higher concepts than we get from rules designed just to classify one example, or rock in the case of the geology domain. On the possibly negative side, this removal of conditions may result in more unseen cases being misclassified compared to the original rule-base which contained more specific rules. The strategy of keeping all cases seen, mentioned before in the removal of redundancy, and using them for evaluation before and after is seen as a way of reducing the problem of creating rules that are too general or specific.

For the purposes of making the knowledge more understandable and useful for explanation, the decision of whether to minimize the rule-base depends on our goal. If the purpose of our KBS is to provide a tutoring system that gives a complete description of a rock then removal of *dispensable* conditions is not desirable. If our goal is to perform inferencing and to understand what are the key features for classifying a rock, then minimization is appropriate. From the point of view of verification, minimization addresses the issue of redundancy but may not pursue the validation goals of the system.

The significance of an attribute.

The significance coefficient is also interesting in giving some insight into the knowledge base and can be treated as a measure of relative redundancy. Two examples are given below. The significance coefficient μ, is calculated by determining the number of objects positively classified minus the number of objects positively classified once that attribute has been dropped, divided by the number of objects in the universe of discourse. Formula 4.6 returns the significance of *attr* in the set of attributes P in the active information system.

$$\mu_{P}\{attr\} = \text{card}\ (POS_P(R) - POS_{P-\{attr\}}(R))\ /\ \text{card}\ (U) \qquad (4.6)$$

where: R={all attributes);card = cardinal number of, POS = positive region, U = universe of objects.

Significance Coefficients for rules concluding Volcanic (%VC000) or Plutonic (%PL000) are:
SignifCoef of attr 0 (GRAIN_SIZE) = 1.000000
SignifCoef of attr 3 (SILICA) = 0.600000

Sig. Coeff. for rules concluding granodiorite %GR000, granite %GR001 and microgranite %GR002:
SignifCoef of attr 0 (GRAIN_SIZE) = 0.666667
SignifCoef of attr 2 (QUARTZ) = 0.666667

Figure 4.14: Significance coefficients produced by rough set theory for two different sets of mutually exclusive conclusions.

In Figure 4.14, the first example for the conclusions %VC000 -Volcanic and %PL000 - Plutonic tells us that the attribute GRAIN_SIZE is very important in deciding on whether a rock should be classified as volcanic or plutonic, in fact all rules for these conclusions used this attribute. However, in some situations, the attribute SILICA will also affect the conclusion. This interpretation is also confirmed by the work using FCA and can be seen in the labelled line diagram in figure 5.15. The second example for the %GR family of conclusions tells us that GRAIN_SIZE and QUARTZ are equally important attributes which help us to decide if a rock should be classified as granodiorite %GR000, granite %GR001 or microgranite %GR002.

4.7.2.3 Phase Two: Removal of Inconsistencies

The 40 rules (objects) output from phase one were processed again to determine whether inconsistencies existed. Use of the dependency coefficient (formula 4.4) on all the rules in an MCRDR KBS is not appropriate because there are a number of conclusions valid for each case so that certain combinations of attribute-value pairs will result in multiple classifications, not just one, being assigned. When we determined κ for the whole SISYPHUS III KBS we got a dependency coefficient of 0 because the positive region was empty. It was therefore necessary to treat each classification individually and compute the dependency coefficient for each class to ensure that within each classification there was no inconsistency. However, it is desirable that rules which give mutually exclusive conclusions be grouped together and be verified for consistency. To achieve this in the SISYPHUS III KBS the rules for those conclusion known to be related but mutually exclusive were put into individual decision tables (files) and the dependency coefficient of each table determined. This was done for the conclusion groups: Plutonic (%PL000) and Volcanic (%VC000); Basic rock (%BS000) and acidic rock (%AC000); granodiorite %GR000, granite %GR001 and microgranite %GR002. Each group scored a dependency coefficient of 1 stating that for that type of decision the rules were consistent. A further indicator of the consistency of the knowledge is the "number of good decisions" reported after each classification. Accuracy was 100% for each individual classification and each classification group.

4.7.2.4 Phase Three: Detection of Deficiencies

When we compute the classification coefficient (formula 4.5) using the 40 SISYPHUS III reduced rules we get a classification coefficient of 1. This computation only provides a partial measure of the completeness of KBS and the discussion in 3.2.3.1 should also be taken into account.

4.7.2.5 Conclusions Regarding the Verification of MCRDR KBS using RST.

In this section we have seen that the approach to detection and correction of verification anomalies developed and described in Section 4.7 could be applied to MCRDR KBS. The only real change involved the detection of inconsistencies which required the dependency coefficient to be computed for individual or identified groups of classifications instead of the whole KBS since multiple conclusions were possible for the one case. Due to the use of cases to develop and validate rules in the RDR KA approach inconsistencies should not be possible anyway. The main benefit of the technique would be the removal of repetition and the simplification of the rules, which requires the completeness and consistency of the resultant KBS to be evaluated. It may also be desirable to sort the list of rules into an MCRDR (see Section 4.5.1).

This returns us to the main and original goal of the work reported in this chapter which was the removal of repetition. In the small example given here there was no loss of accuracy, a small reduction in the KBS size and a substantial decrease in the number of conditions per rule. The generalisation of the MCRDR knowledge and the reduction in complexity is seen to be a useful outcome. However, as mentioned, the information loss may not affect the system's ability to provide the correct classification but it is unclear whether the explanations provided by the reduced KBS will be more or less useful and in what way.

4.8 Chapter Summary

Sections 4.7.1 and 4.7.2 have described how RST can be utilised to assist the verification of conventional and MCRDR rule bases, respectively. Both sections have shown that automated attempts to perform verification generally require some sort of human intervention to determine whether the anomalies detected by the system actually require fixing and those fixes will include a mixture of automatic and manual procedures. This cooperative effort is even more noticeable in Section 7.3.1 on critiquing where an approach to computer-assisted validation is described. In the approach the computer becomes an assistant in helping the user determine where errors in the knowledge may exist.

Sections 4.1 to 4.4 were concerned specifically with the issue of removing repetition from single classification RDR KBS using RST. In Sections 4.1 to 4.4 the experiments performed by Chellen (1995) using Induct were also discussed and compared to the RST results. RST appeared better suited to the task of removing repetition but the usefulness of the compacted KBS for explanation has not been answered. The compaction process involved the reorganisation of the KBS. It is assumed that the repetition due to local patching is removed through the reorganisation process but the method is not based on the detection of repetition and the removal of identified repeated knowledge. In this way it is unclear to what extent the compaction removes repetition but since the size of a KBS is affected by the amount of repetition, a reduction in size is seen to be a reduction in repetition. Section 4.5 offered a proposal for the removal of repetition from MCRDR KBS and describes a sorting list algorithm for MCRDRs.

As stated previously, the original and primary interest was in how we can understand what knowledge we have so that it can be reused. Rough sets have been useful in analysing an existing knowledge base and improving the consistency of that knowledge which will be a better starting point when the knowledge is later put to alternative uses. It had been hoped that RST would provide the links between concepts in the knowledge that were seen as important in facilitating multiple uses of the same knowledge for different activities. It became apparent however that RST does not provide the necessary structure or abstractions. As an alternative solution, the abstraction hierarchy that FCA develops seemed a plausible answer to this problem which led to the work described in the next chapter.