The Automatic Compression of Multiple Classification Ripple Down Rule Knowledge Based Systems: Preliminary Experiments

Hendra Suryanto*, Debbie Richards** and Paul Compton*
*Artificial Intelligence Department, School of Computer Science and Engineering
University of NSW, Australia. Email: {hendras, compton}@cse.unsw.edu.au
**Department of Computing, Div.of Info. & Communications Sciences
Macquarie University, Sydney, Australia. Email: richards@ics.mq.edu.au

Abstract: Ripple Down Rules (RDR) have a longstanding (and successful) history in the field of biomedical engineering. RDR are a knowledge acquisition and representation technique that allow knowledge to be rapidly acquired and maintained by the domain expert. A key feature of RDR, and the reason why maintenance is easily managed, is that rules are never modified or deleted but they are locally patched. That is, new rules are exceptions to previous rules and the new rule is validated within the context of previously seen cases. One drawback of locally patching is that knowledge can be repeated in different locations of the knowledge base. This paper describes some work done on removing repeated knowledge. The experiments reported were performed on a pathology knowledge base but the algorithm is applicable to any multiple classification RDR knowledge based system. The results support the findings of others that exception structures are compact representations with few opportunities to reduce further. This also suggests that experts tend to provide overly general rules in the first instance which they modify by adding specialistions in the form of exception rules as new cases are seen.

1 Introduction

Ripple Down Rules (RDR) have a longstanding (and successful) history in the field of biomedical engineering. In the domain of chemical pathology, engineering techniques have been applied to the problem of adding an interpretative comment onto pathology reports (Edwards et al 1993). Such a comment adds value to the report and results in a situation where the human expert (the pathologist) is able to confer with the computer expert (the expert system) regarding the correct interpretation of a set of pathology results. From the point of view of the pathologist this reduces the amount of time spent writing comments and ensures consistency. From the point of view of the patient, they get a second medical opinion.

RDR are a knowledge acquisition and representation technique that allow knowledge to be rapidly acquired and maintained by the domain expert. The approach is based on the combined use of cases to motivate knowledge acquisition and for validation. The knowledge is acquired as production rules and structured into decision lists of exceptions. A key feature of RDR, and the reason why maintenance is easily managed, is that rules are never modified or deleted but they are locally patched (Compton, Preston and Kang 1994). That is, new rules are exceptions to previous rules and the new rule is validated in the context of previously seen cases.

One drawback of locally patching is that knowledge can be repeated in different locations of the knowledge base. Studies (Compton,Preston and Kang 1994, Mansuri et al 1991) have shown that the incidence of repetition is not high and that RDR build KBS that are comparable in size to those produced using the machine learning algorithms Induct and C4.5. Nevertheless, repetition is a verification anomaly that should be kept to a minimum so that fewer patches need to be applied and thus reduce the KA load. Also, although inferencing is not greatly impacted by the existence of repetition, repeated knowledge makes the KBS comprehensible which is important for alternative uses of the knowledge such as explanation or exploration. Earlier work (Richards, Chellen and Compton 1996) has looked at compacting single classification RDR. This paper is concerned with the more recent and robust representation known as Multiple Classification RDR (MCRDR). The results reported are preliminary and alternative algorithms are also being investigated.

The experiments reported in this paper were performed on a pathology knowledge base known as *Ultra3700* but the algorithm is applicable to any MCRDR KBS. *Ultra3700* was developed using a commercial version of MCRDR known as LabWizard by Pacific Knowledge Systems. In the next section we define some MCRDR terminology. In sections 3 and 4 the problem and method are described. In section 5 the results are given. Section 6 provides discussion of the results, future work and conclusion.

2 MCRDR Terminology

The following terms can be better understood by viewing figures 1, 2(a) and 2(b) which show examples of an MCRDR node, MCRDR tree and conversion of the tree to flat rules, respectively.

A node is one node in an MCRDR tree structure. A node contains conjunctions of conditions, a conclusion, a pointer to its parent and pointers to all children.

A child is an exception node whose conclusion replaces the conclusion of the parent.

A rule path is all conditions of ancestors and conditions of this node.

A flat rule is a rule path and one negative condition from every children condition.

A group of flat rules is all flat rules that belong to one MCRDR node. That is, conditions on a rule path and all possible combinations of children conditions. For example, if a node has five children, two children with 3 conditions, one children with 4 conditions and two children with 6 conditions, this node has 3 x 3 x 4 x 6 x 6 = 1296 flat rule.

An edge is a pair of a parent and one of its children. An edge is an atomic part of MCRDR which still carries the characteristics of MCRDR the structure.

An Empty edge is two nodes, parent and child, neither of which have a conclusion.

A Node has no conclusion, if

- this node is a stopping rule[1] OR
- all flat rules of this node are
    - subsumed by another flat rule OR
    - equal with another flat rule.

3 The Problem

There are two ways to minimise the MCRDR tree size as shown in Figure 1:

- combine 2 nodes into a single node
- delete a node

Both of them (combine and delete) can not be done at the flat rule level or the node level. Combining and deleting can only be done at the edge level. (That is, we should consider information in every edge that relates to this node before combining or deleting this node).

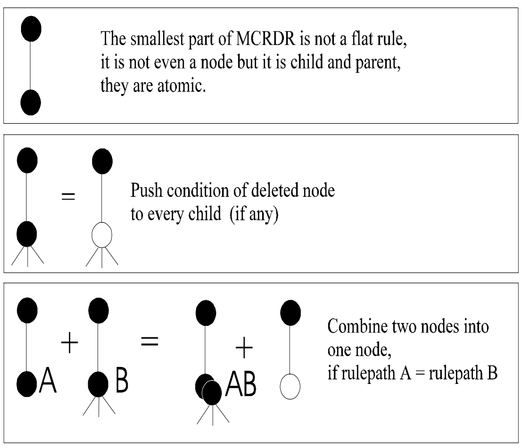

Figure 1: An example of when one node can be deleted and two nodes can be combined

A node **can be deleted** if and only if

- this node has no conclusion AND
- parent of this node also has no conclusion.

If one node is deleted, all conditions of this node should be pushed down to every child of this node.

Two nodes **can be combined** if and only if

- both of them have the same conclusion AND
- both of them have the same rule path AND
- both of them have the same child AND
    - same number of children **and**
    - same condition of every child (conclusion of every child is not necessarily the same)

OR

- one of them has no conclusion and also has no child AND
- both of them have the same rule path

4 The Method

An algorithm to minimise the number of nodes in an MCRDR is given described in this section. An example of how this algorithm works is demonstrated in Figure 2.

1. Convert all nodes to flat rules.
2. Group all flat rules by conclusion. That is, group nodes with the same conclusion in the same group.
3. Mark all flat rules that are the same as another flat rule. Choose the one in the flat rule group wit the fewest non-marked members.
4. Mark all flat rules subsumed by other flat rules.
5. Convert all flat rules back to the original MCRDR structure.
6. Delete all nodes which have no conclusion and the parent of these nodes also has no conclusion, push conditions of these nodes to every children (if any).

5 The Results

Table 1 shows the results for the *Ultra3700* KBS before and after compression. We can see a 10 percent (3710->3322) or up to 25 percent (2927->2135) reduction in size, if we allow negative conditions and thus eliminate stopping rules[2]. Both results are disappointing. It appears that there are few opportunities to combine 2 nodes in the *Ultra3700* KBS, since there are no nodes which satisfy all conditions described in section 3. Since the only available data for this experiment was *Ultra3700*, it is not clear whether this result is peculiar to this domain or typical of MCRDR KBS in general. Although combining nodes was not possible, it was possible to delete nodes. In the experiment, 21.6% of nodes could be converted to *node with no conclusion* but only 10.5% of nodes could be deleted (since the other node is still needed by its parent as a stopping rule).

	Leaf Rules	Stopping Rules	Others	Total
MCRDR before compression	2231	783	696	3710
MCRDR allowing negative conditions	2231		696	2927
Flat rules of MCRDR (w/o 16 nodes)				14589
Flat rules after compression				4452
MCRDR after compression	2070	1187	65	3322
MCRDR allowing negative conditions	2070		65	2135

Table 1: Results of compressing Ultra3700

[1] A stopping rule is a child rule with a null conclusion which stops an earlier conclusion from being asserted.

[2] Stopping rules with one condition (the majority) can be removed if we incorporate a negative condition into the parent.

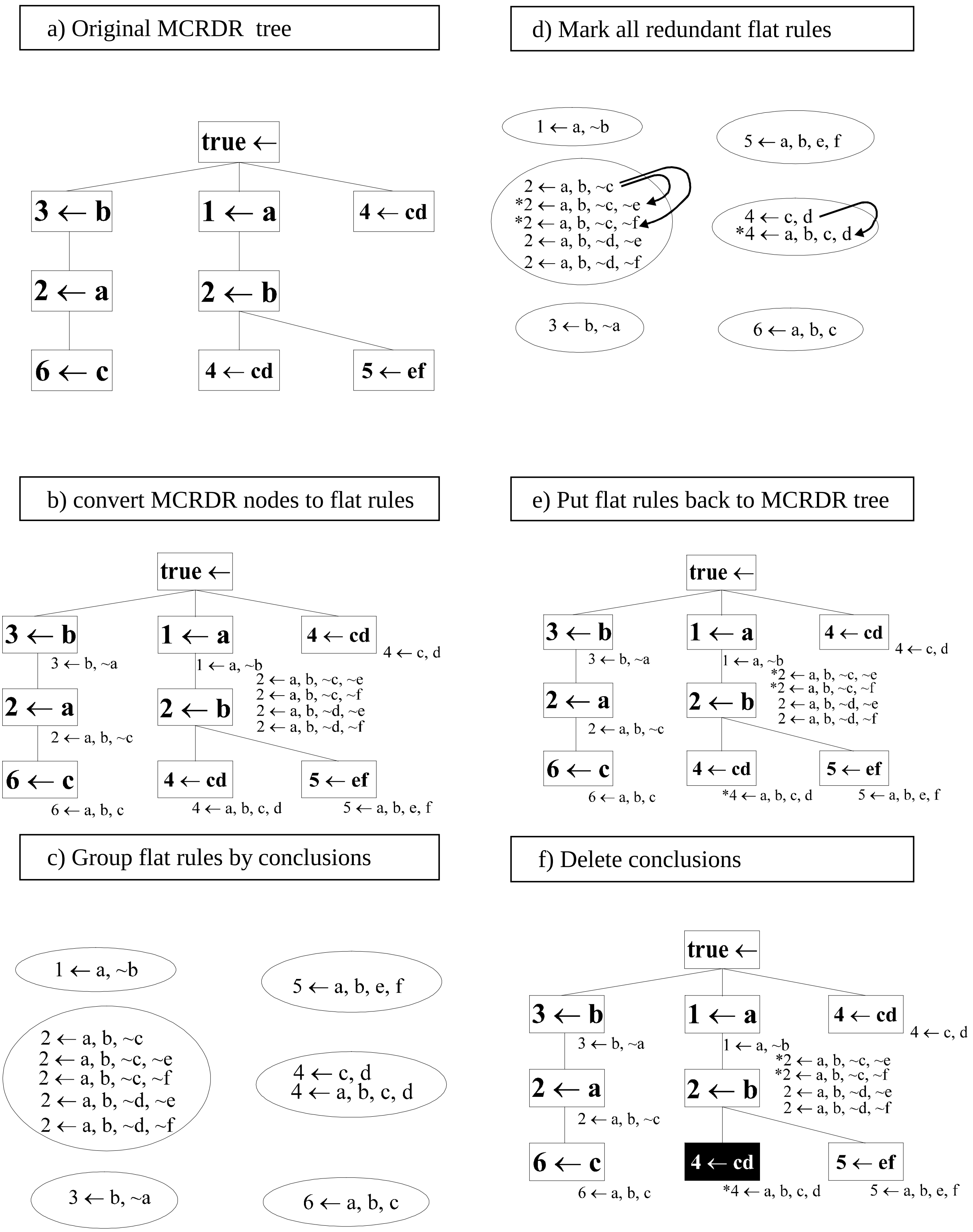

Figure 2: The steps involved in compressing an MCRDR tree

6 Discussion and Conclusion

It appears that there was little opportunity for compression in the *Ultra3700* KBS. We could have modified or extended the KBS to introduce further instances of repetition but we wanted to use a real, not simulated, KBS. Since *Ultra3700* was the only KBS made available to us, we were unable to test our algorithm on other large KBS to confirm whether our results were generally applicable or peculiar to the *Ultra3700* KBS. For commercial and confidentiality reasons, we are unable to publish our data and *Ultra3700* can not be made publicly available for experimentation by others.

Although the algorithm offered has not significantly compressed the original KBS, it has removed almost all nodes which can possibly be deleted given the definitions in Section 3. Not all possible nodes have been deleted because nodes which have too many flat rules, more than 10^{100}, (*16 nodes in Ultra3700*) were not included in the compression process. The process has eliminated all duplicate conclusions, but with a resultant increase in the number of nodes in the MCRDR structure as reported in Table 1. This increase is due to the increase in stopping rules which are necessary to ensure that a case can not be covered by more than one node with the same conclusion.

We are not only interested in whether we have reduced the size of the knowledge base, want to ensure that its accuracy level is maintained. In the process described, after compression every conclusion for every case remains the same, except conclusion 5 and conclusion 1. Conclusion 5 is *stopping rule* and conclusion 1 is *no conclusion*. Thus, the new knowledge base is equivalent.

A side-effect of this compression is that some knowledge acquisition history is lost. If such a history is important in a particular domain, compression is not recommended.

The small amount of compression that could be achieved (10 percent of the original) confirms what has already been observed by others (Catlett 1992, Gaines and Compton 1992, Kivenen, Manilla and Ukkonen 1993, Scheffer 1996 and Siromoney and Siromoney 1993) that exception structures are compact representations. It appears that although repeated knowledge is inevitable when local patching is used, the resulting tree is close to optimally compact. This also confirms that experts tend to provide overly general rules in the first instance which they modify by adding specialistions in the form of exception rules as new cases are seen.

Further work is underway using the Duce (Muggleton 1987) algorithm. Results-to-date indicate that most of the compaction that occurs is due to the truncation feature in Duce and this results in a loss of the original MCRDR structure. The main impact of this is that there is no longer any link to cases. Cases are an integral part of the RDR KA technique as they are used to maintain the systems validity and to assist the user in the selection of rule conditions.

Acknowledgements

RDR research is supported by various ARC grants. The authors would like to thank Lindsay Peters, Gerard Ellis and James Casca of Pacific Knowledge Systems for their assistance with the experiments in this work.

BIBLIOGRAPHY

Catlett, J. (1992) Ripple-Down-Rules as a Mediating Representation in Interactive Induction *Proceedings of the Second Japanese Knowledge Acquisition for Knowledge-Based Systems Workshop***,** Kobe, Japan, Nov 9-13 1992, 155-170.

Compton, P., Preston, P. and Kang, B., (1994) Local Patching Produces Compact Knowledge Bases *A Future in Knowledge Acquisition* (eds) L. Steels, G. Schreiber and W. Van de Velde, Berlin, Springer Verlag, 104-117.

Edwards, G., Compton, P., Malor, R, Srinivasan, A. and Lazarus, L. (1993) PEIRS: a Pathologist Maintained Expert System for the Interpretation of Chemical Pathology Reports *Pathology 25*: 27-34.

Gaines, B.R. and Compton, P. (1992) Induction of Ripple-Down Rules, *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence (AI'92),* Hobart, Tasmania, World Scientific, Singapore, 349-354.

Kivinen, J., Mannila, H. and Ukkonen, E. (1993) Learning Rules with Local Exceptions *Proceedings of the European Conference on Computational Learning Theory COLT'93*

Mansuri, Y., Kim, J.G., Compton, P. and Sammut, C. (1991). A comparison of a manual knowledge acquisition method and an inductive learning method *Australian Workshop on Knowledge Acquisition for Knowledge Based Systems,* Pokolbin (1991), 114-132.

Muggleton S. (1987) Duce, An Oracle-based Approach to Constructive Induction *Proceeding of the Tenth International Joint Conference on Artificial Intelligence (IJCAI'87), 287-292.*

Richards, D., Chellen, V. and Compton, C (1996) The Reuse of Ripple Down Rule Knowledge Bases: Using Machine Learning to Remove Repetition In Compton, P., Mizoguchi, R., Motoda, H. and Menzies, T. (eds) *Proceedings of Pacific Knowledge Acquisition Workshop PKAW'96*, October 23-25 1996, Coogee, Australia, 293-312.

Scheffer, T. (1996) Algebraic Foundation and Improved Methods of Induction of Ripple Down Rules Repetition In Compton, P., Mizoguchi, R., Motoda, H. and Menzies, T. (eds) *Proceedings of Pacific Knowledge Acquisition Workshop PKAW'96*, October 23-25 1996, Coogee, Australia, 279-292.

Siromoney, A. and Siromoney, R.(1993) Variations and Local Exceptions in Inductive Logic Programming, In K. Furukawa, D. Michie, S. Muggleton (eds.), *Machine Intelligence – Applied Machine Intelligence* (14), 213-234.