*Int. J. Human–Computer Studies* (1996) **44,** 127–144

# On the validation and verification of production systems: a graph reduction approach

Stephen Murrell
*Department of Computer Science and Mathematics, University of Miami, Coral Gables, FL 33124, USA*

AND

Robert Plant
*Department of Computer Information Systems, University of Miami, Coral Gables, FL 33124, USA. email: rplant@umiami.miami.edu*

This paper takes a parallel processing approach to the implementation of rule-based systems using a graph-reduction architecture, and investigates the consequences of this architecture in relation to the validation and verification of knowledge-based systems. The paper improves on the traditional sequential approaches to the development of knowledge-based systems and the limited validation and verification techniques that are applicable. This is contrasted with a graph reduction implementation of knowledge-based systems development based on an ALICE-like machine. The advantages of this style of programming in relation to systems development and program correctness are discussed. The paper shows that significant benefits could potentially be achieved through the use of graph-reduction techniques in the development of these systems. © 1996 Academic Press Limited

## 1. Introduction

This paper presents a parallel graph-reduction approach to the implementation of knowledge-based systems. This approach relies upon a specification of the system in terms of decision tables, and enables the automatic generation of programs that implement the knowledge encoded in those tables. These automatically generated programs are written in Malice (Murrell, 1989), a generic graph reduction programming language. Automatic generation of programs from decision tables eliminates the possibility of programming errors being included, and thus reduces the validation, verification, and testing overhead.

The system takes decision tables as specifications of correct behavior, and therefore relies upon a correct formulation of those tables. However, it is inevitable that in real-world applications, human errors will occur, and result in inconsistent, incomplete, or incorrect decision tables. It is accepted that the decision table could be incorrect; the paper addresses the impact of decision table errors on the system as a whole. This system has a degree of robustness uncommon in conventional implementations, and will continue to function even with a conflicting data set.

Traditionally *validation* has been defined as determining whether an appropriate product is being created; *verification* is the process of checking that product has been created correctly (Boehm, 1981). The technique we use is primarily one of verification, together with in-system run-time consistency checks. Due to the concurrent nature of the parallel implementation, there is generally no run-time overhead caused by the consistency checks.

1071-5819/96/020127 + 18$12.00/0

After a review of existing works on validation and verification, and parallel implementations, for rule-based systems, we briefly introduce (in Section 3) the ideas of graph reduction implementations of decision tables, then, in Section 4, examine the consequences of decision table errors, and show how their impact can be minimized.

## 2. Background to validation and verification for parallel knowledge based systems.

Research into validation and verification of knowledge-based systems has been progressing since the mid 1980s, when the need for techniques that considered the completeness and correctness of rule-based systems became a concern to commercial developers.

The foundations of the research into validation, verification and testing can be traced back to work on testing in relation to conventional systems and its extrapolation to the testing of early rule-based systems such as MYCIN (Suwa, Scott & Shortliffe 1982). The creation of tools to assist in the process was soon to follow, one of the first being applied to the R1/XCON System (Soloway, Bachant & Jensen 1987). From these beginnings we can break the research into five broad areas, each of which has its own extensive literature, examples of which are cited below.

- Expert System Validation (Green & Keyes, 1987; Naser, 1988; O'Leary, 1988; Rushby, 1988; Geissman & Schultz, 1988; Rushby & Whitehurst, 1989; O'Keefe & O'Leary, 1992; Coenen & Bench-Capon, 1993).
- Knowledge-base Verification (Suwa *et al.,* 1982; Nguyen, Perkins & Laffery, 1985; Ginsberg, 1987; Marcot, 1987; Cragun & Steudel, 1987; Stachowitz, Chang, Stock & Coombs, 1987; Morell, 1988; Schultz & Geissman, 1988; Botten, Kusiak & Raz, 1989; Radwan, Goul, O'Leary & Moffitt, 1989; Lehner, 1989; Miller, 1990; Ayel & Laurant, 1991*b,c*; Preece, Shinghai & Bataekh, 1992; Antoniu, 1993; Preece, 1993; Valiente, 1993; Bench-Capon, Coenen, Nwana, Paton & Shave, 1993).
- Tools (Freeman, 1985; Ginsberg & Rose, 1987; Cragun & Studel, 1987; Krishnamurthy, Padalkar, Sztipanovits & Purvis, 1987; Loiseau, 1989; Kang & Bahill, 1990; Vanthienen, 1991; Zlatarova, 1991; Ayel & Laurant, 1991*a*; Charles & Dubois, 1991; Cuda & Dolan, 1991; Preece & Shinghal, 1991; Becker, Green & Bhutinager, 1991; Steib, Small, Castells & Schofield, 1991; EPRI, 1993; SENTAR, 1995).
- Development (Chen, 1976; Guttag & Horning, 1978; Davis & Lenat, 1982; Grover, 1983; Buchanan *et al.,* 1983; Carpenter & Murine, 1984; Wielinga & Breuker, 1984; Alexander, Freiling, Shulman, Staley, Rehfuss & Messick, 1986; Ince & Hekmatpour, 1987; Breauker *et al.,* 1987; Boehm, 1988; Humphrey, 1989; Weitzel & Kershberg, 1989; Plant, 1991; ANSI, 1992; TRILLIUM, 1992; Breuker & Van de Velde, 1994; Gold & Plant, 1994; Plant & Tsoumpas, 1994; Akkermans, Schreiber & Weilinga, 1994; de Hoog, Martil, Weilinga, Taylor, Bright & Van de Velde, 1994).
- Formal Methods (Bezem, 1987; Dahl, 1990; Bolonga, Ness & Siverstsen, 1990; Breu, 1991; Fox, 1993; Herre, 1993; Meseguer, 1993; Hors & Rousset, 1993;

Rousset, 1993; Roman, Gamble & Ball, 1993; Krause, Fox, O'Neill & Glowinski, 1993; Rousset, 1994; Gold & Plant, 1994; Ourston & Mooney, 1994; Vermasan & Wergeland, 1994*a,b*; Bouali, Loiseau & Rousset, 1994; Krause, Byers & Hajnal, 1994; Murrell & Plant, 1995*a*).

The application of Parallel Processing to Artificial Intelligence has been primarily in the areas of vision processing, image analysis and robotic systems, areas with high computational demands. The utilization of parallelism in the development of knowledge-based systems has as a general rule been limited to prototype systems in the areas of medical diagnosis (Plant, Murrell & Moreno, 1994; Murrell & Plant, 1995*b*; Todd, Stamper & Macpherson, 1995). The extension of parallel processing into the area of knowledge-based systems development with a focus on the validation and verification issues currently has only a small literature (Murrell & Plant, 1995*b*).

The authors of this paper wish to extend the research into the application of parallel processing for knowledge-based systems as it is our belief that there is a fundamental problem with validating rule-based systems that have been implemented in traditional programming styles such as LISP, CLIPS or OPS5. It is our premise that these environments inhibit testing due to the complexity of the implementations' syntactic structures, and that the validation of the system should be performed automatically where possible, and at run-time by the system, thus, relieving the programmer of this overhead. Further, the system should be specifiable. In order to achieve these two goals the authors advocate the specification of the systems' rules in a simple decision table form that can be automatically translated into an ALICE† graph-reduction-machine program that is executable in a multi-processing environment.

## 3. Parallel processing and graph reduction

With the movement of rule-based systems from the research laboratory into an industrial setting there has been a significant increase in the size of the rule-bases and a demand for faster processing. Any increase in processing speed has to be derived in one of two ways: either by adapting the representation [e.g. ordering the rules through techniques such as clustering (Mehotra, 1993)], or by new implementation platforms, such as parallel processing. There are many approaches to parallel processing that could be taken [e.g. the Hypercube architecture (Seitz, 1985), Parlog (Clocksin & Mellish, 1984; Clark & Gregory, 1986), the Connection Machine (Hillis, 1985), Occam and the Transputer (Hoare, 1985; Jones, 1986), Neural Nets (Minsky & Papert, 1969; McLelland & Rumelhart, 1986)) however few have been applied (Plant *et al.*, 1994; Todd *et al.*, 1995) to the implementation of knowledge-based systems. In this work, we combine both directions, making a parallel implementation based on an improved representation. The graph reduction architecture is based upon the representation of programs and data in an efficiently interconnected form, which allows the elimination of any searching, and gives a very natural representation of the decision structure.

† We do not use ALICE itself, but a local implementation (MALICE) which follows the original very closely.

## 3.1. GRAPH REDUCTION

Graph Reduction systems (Darlington & Reeve, 1981; Townsend, 1987; Reeve & Zenith, 1989) provide a form of automatic concurrency in the execution of programs. Programs and data are encoded as graphs in which the nodes represent items of data and computational operations, and the arcs represent the structural relationships between items of data, the interdependencies of computational operations, and the application of operations to data. Eligible computational nodes are selected, by nondeterministic means, for execution; if multiple processors are available, multiple nodes will be executed concurrently. Any algorithm translated into a graph reduction implementation can be expected to run with a degree of concurrency, but for optimal concurrency, some deliberate design effort is, of course, required. Graph reduction provides a high level conceptual base for program design; the low level concerns of more conventional parallel platforms (such as the interprocess communications overhead, and protection of shared data) are abstracted away. As the concurrency is virtually automatic and transparent, it has no impact of its own on the validation and verification process.

The general principle upon which this technique is built, is that a rule-based system, originally provided in the form of a decision table, may be directly and automatically translated into a graph. The graph itself may be understood as a program to be executed by a graph-reduction computer. A very simple example is shown below in Figures 1 and 2, which are from Plant *et al.* (1994) where a detailed explanation may be found; the technique is covered fully in Murrell (1989) and Murrell & Plant (1995*b*).

Each node in the graph represents an executable "packet". A graph-reduction machine in the style of ALICE (Darlington & Reeve, 1981) performs its computation by repeatedly selecting at random such a packet, and replacing it be an equivalent (possibly empty) sub-graph of packets according to a set of programmed rules.

Initially, only the "program" packet is eligible for selection; as it is dependent upon two "conclude" packets, those two will become eligible. Eligibility of packets for selection is propagated through out the graph, according to the programmed rules, until some non-dependent packets become selectable.

When non-dependent packets (e.g. "condition" packets) are executed they are replaced, according to the programmed rules, by what may be considered results;

| | 1 | 2 | 3 |
|---|---|---|---|
| q1: | Y | Y | N |
| q2: | N | Y | N |
| q3: | Y | - | N |
| c1: | X | X | |
| c2: | | X | X |

FIGURE 1. A trivial decision table.

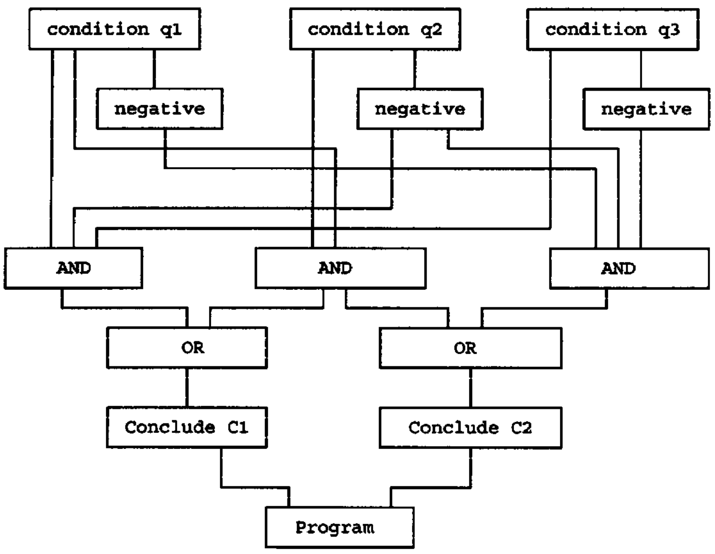

FIGURE 2. The graph created from Figure 1.

packets dependent upon these results thus become executable, until eventually the "conclude" packets are able to provide their answers. An example of a large scale application in the domain of psychiatry is described by Plant *et al.* (1994) and a detailed case study in graph-reduction development is provided by Murrell & Plant (1955*b*).

## 4. Verification and graph-reduction

In this section we consider the aspects of verification that are of key concern to the area of rule-based systems. We take the union of the areas identified by Culbert (1990), Preece (1993) and O'Leary (1994) which enumerate types of possible defects in the correctness of rule bases: redundancy, conflict, circularity, and errors introduced by incorrect knowledge acquisition. These aspects of the validation of rule-based systems have been considered by other researchers in relation to conventional implementations (Nguyen *et al.,* 1985; Rushby, 1988; O'Leary, 1994), and are given a full treatment (with respect to the validation and verification of decision tables) in Murrell & Plant (1995*c*).

In the following sections we follow the organization of Murrell & Plant (1995*c*) showing how the four major decision table error types: *Redundant rules* (including: *identity, subsumption, indirect, unfireable, reducible*), *Conflicting rules, Circular rules,* and *Errors of omission* (*unused inputs, missing rules, impossible combinations, dead end rules*) affect and are affected by a graph reduction implementation. Accepting the assumption that decision tables used as specifications for graph-reduction implementations may not be totally correct, it is necessary to be aware of both the semantic and syntactic errors that can occur, and work towards methods for their detection and solution. In general, semanic errors can not be detected and we show what effects their presence can have on the behavior of the system. Syntactic errors are easier to detect, and for these we also discuss the appropriate detection methods.

## 4.1. REDUNDANT RULES

A redundant rule is simply one which makes no contribution to the system. Redundancy may be decomposed into five sub-categories: *identity, subsumption, indirect redundancy, unfireability,* and *reducibility.* None of these cause any practical problems for the graph-reduction implementation.

### *4.1.1. Identity*

The first type of redundancy to be considered is that of identical rules, which can be broken down into two sub-categories: *syntactic* and *semantic* redundancy.

The case of syntactic redundancy is illustrated thus:

RULE 26: IF X AND Y THEN Z
RULE 93: IF Y AND X THEN Z

where both rules will be applicable if X and Y have been substantiated. This can cause several problems in traditional implementations in that the rule may be fired twice, as the conflict resolution strategy is often ineffective in removing or coping with redundancy. However, in the graph reduction implementation these problems can not arise, as once a rule fires it ceases to be computable and therefore can never be fired again. In many cases second and subsequent rules leading to the same conclusion would never even be tested once the conclusion has fired. This also illustrates the automatic conflict resolution strategy of this implementation.

The implementation of redundant rules, therefore, is not problematic for a graph-reduction implementation. However, the developer may wish to detect these redundancies prior to implementation. This can be done when the decision tables are constructed. Syntactically redundant rules can be identified as identical columns in a decision table. For example, the rules given above would appear in the form of Figure 3. This would produce the graph shown in Figure 4.

Syntactic redundancy presents no practical difficulties, it may be taken as a sign of an error in the rule-base and is efficiently detectable as shown in Murrell & Plant (1995*c*).

| | |
|---|---|
| | . . . 26 . . . 93 . . . |
| c1: X | . . . Y . . . Y . . . |
| c2: Y | . . . Y . . . Y . . . |
| a1: Z | . . . X . . . X . . . |

FIGURE 3. Redundant rules.

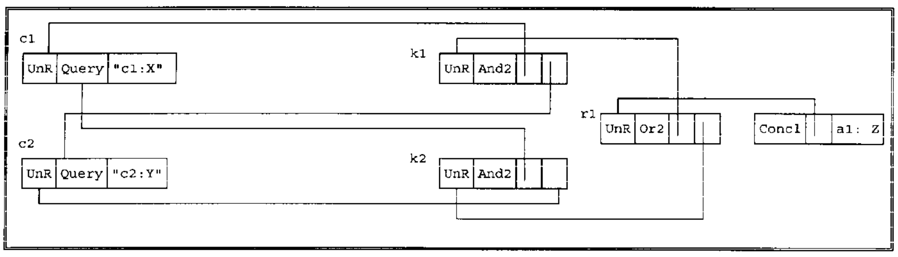

FIGURE 4. Graph reduction of syntactically redundant.

A less tractable, but strongly related problem is **semantic redundancy**. This covers cases when two (or more) rules have the same meaning, but are formulated in different ways.

RULE 45: IF X AND Y THEN "weight > 2240lbs"
RULE 83: IF Y AND X THEN "weight > 1 ton"
(i.e. the conclusions are semantically equivalent).

| | 45 | 83 |
|---|---|---|
| c1: X | Y | Y |
| c2: Y | Y | Y |
| a1: Z1 | X | |
| a1: Z2 | | X |

FIGURE 5. Semantic equivalence.

These would be represented by the decision table shown in Figure 5 and the graph reduction implementation is shown in Figure 6. The problem becomes more acute when the conditions are semantically but not syntactically equivalent. For example:

RULE 63: IF hot AND humid THEN thunderstorms
RULE 99: IF sultry THEN electricalstorms.

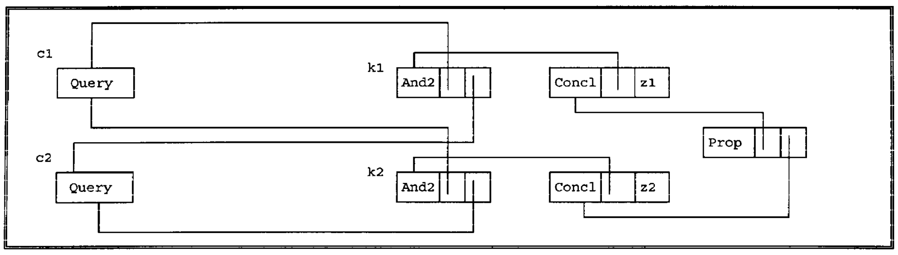

FIGURE 6. Graph reduction of semantic redundant rules.

| | 7 | 8 |
|---|---|---|
| c1: W | Y | Y |
| c2: X | Y | Y |
| c3: Y | Y | - |
| a1: Z | X | X |

FIGURE 7. Subsumption.

There is no possibility for a solution to this problem being brought about by graph reduction or any other implementation method; the problem can not be identified without some knowledge that is outside the system. (i.e. hot and humid means sultry). Problems of this type (occasionally referred to as *deep inconsistencies*) are in general not open to solution without the application of intelligence, and arise in many different forms.

### *4.1.2. Subsumed rules*

One rule is said to be subsumed by another, when it specifies that the same (or fewer) actions are to be applied under the same (or stricter) conditions. Subsumption is a generalization of the problem of identity, and has both syntactic and semantic variants, of which only the former is practically detectable. As an example, in the following, rule 7 is subsumed by rule 8.

RULE 7: IF W AND X AND Y THEN Z
RULE 8: IF W AND X THEN Z,

which may be represented as a decision table, such as Figure 7. These would then produce the graph shown in Figure 8.

Subsumed rules do not create any significant problem for the graph-reduction approach to the implementation of production systems because once a conclusion has been fired it ceases to be computable and therefore can not be fired again (as in

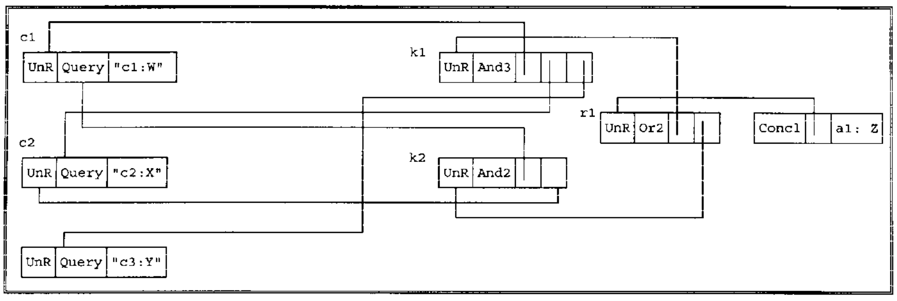

FIGURE 8. Graph reduction of subsumed rules.

the case of syntactic redundancy). The only slight problem is that the number of packets created may be increased unnecessarily.

### *4.1.3. Indirect redundancy*

Indirect redundancy of the form:

RULE 11: IF p THEN q
RULE 22: IF q THEN r
RULE 33: IF p THEN r

can only be reliably detected by a brute force search over all possible sets of inputs. Clearly such a search which would require exponential time is not a *practical* proposition for any real system (although some systems do attempt this). Indirect redundancy is again a generalization of Identity, and also has an intractable semantic variant.

In our graph reduction approach the search would not be necessary as the reduction process would fire based upon the quickest reduction. Thus, a significant advantage is achieved through this approach for the usual reason that conclusions can not be fired twice.

### *4.1.4. Unfireable rules*

A rule may be unfireable for one of three reasons:

- its condition is a logical impossibility (e.g. rule 23 below),
- its condition is logically possible but no combination of other rules firing can satisfy it (e.g. rule 97 below, under the assumption that both m and n can be true, but not at the same time),
- the condition is semantically impossible (e.g. rule 16 below).

RULE 23: IF p AND NOT p THEN r
RULE 97: IF m AND n THEN x
RULE 16: IF vital AND unimportant THEN action

The first form cannot occur in standard forms of decision table, and is therefore not a problem. The second form can be detected in the decision table after a search over all possible input values. The third form, as with all semantic errors, can not be detected by practical means.

The presence of an unfireable rule may simply result from incomplete knowledge on the part of the original human expert, and is not *per se* wrong; nor does it cause any run-time problems. Future additions to the knowledge base may reverse the situation and render the rule fireable.

### *4.1.5. Reducible rules*

When two rules have conditions that are identical but for one variable, and that one variable appears in a positive form in one rule, and negated in the other, and the actions associated with the two rules are identical, then those two rules may be reduced to one, by simply ignoring the differentiating variable. For example:

RULE 9: IF X AND Y AND Z THEN A
RULE 12: IF X AND NOT Y AND Z THEN A

| | 9 | 12 |
|---|---|---|
| c1: W | Y | Y |
| c2: Y | Y | N |
| c3: Z | Y | Y |
| a1: A | X | X |

FIGURE 9. Rule reduction.

may be reduced to:

RULE 912: IF X AND Z THEN A

The unreduced form appears in a decision table as shown in Figure 9. This would be transformed into a graph with the form shown in Figure 10 which executes correctly. Reducible rules may be detected and reduced after a search of the decision table, but do not need to be removed. The only potential disadvantages to leaving them unreduced are that more packets are created than are strictly necessary, and some irrelevant questions may be asked of the user. The correct operation of the system is not compromised.

## 4.2. CONFLICTING RULES

Rules are in conflict when one allows a particular conclusion to be deduced, another allows the inverse of that conclusion to be deduced, and both are able to fire. For example:

RULE 1: IF P THEN Q
RULE 42: IF P THEN NOT Q

or

RULE 3: IF very_cold THEN nice_day
RULE 40: IF frigid THEN NOT nice_day

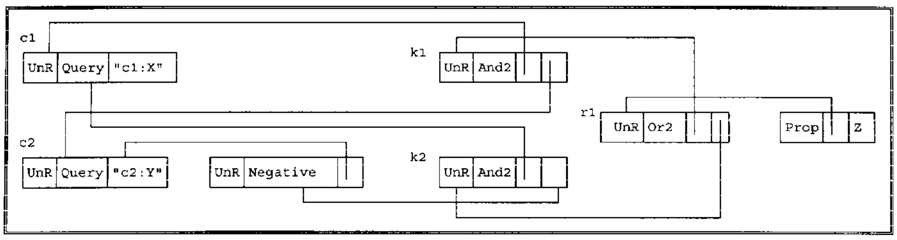

FIGURE 10. Graph reduction of unnecessary rules.

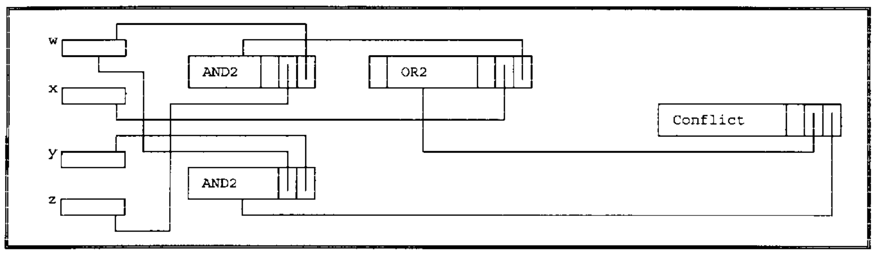

FIGURE 11. Run-time conflict detection.

Syntactic conflict may be detected by a search of the decision table; semantic conflict is not practically detectable. In any case of conflict, when the conclusions are syntactic inverses (as in the last example above), the graph-reduction implementation will always accept whichever conclusion is deduced first and not change if a conflict arises later (once a conclusion has fired, it can not fire again, so can not change its logical state), so a user may never become aware of the error. If there is a risk of such conflicts, it is possible to add run-time consistency checking in the form of an extra packet that monitors the results of conditions that could lead to conflicts, ensuring that improper combinations never occur.

RULE 17: IF X THEN A
RULE 18: IF W AND Z THEN NOT A
RULE 19: IF Y AND W THEN A

This would produce the graph of Figure 11 (conclusions have been omitted for clarity).

The "conflict" packet, combining the trees for [X OR (Y AND W)] and [W AND Z] is activated only if both reduce to true or both reduce to false, and produces an error warning.

This solution may easily be generalized to cover systems which have sets of complementary solutions such as negative/zero/positive (i.e. sets of conditions which are mutually exclusive), by extending the actions of the "conflict" packet to signal an error if more than one of its argument packets reduces to true, see Figure 12.

## 4.3. CIRCULAR RULES

Circularity is present when there is a sequence of rules, each of which "calls" the next, and the last of which "calls" the first. This would appear in one of two forms shown in Figures 13 and 14:

In many existent systems, either of these would be likely to cause an infinite loop. In a graph reduction implementation, this can not happen. With the first representation this is due to the independent nature of packets (Figure 15). With the second representation, a circular structure would be created (Figure 16). Once either of X

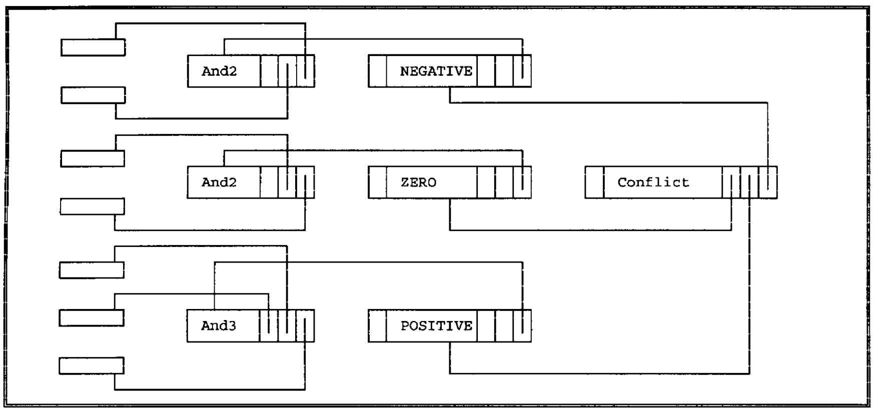

FIGURE 12. Conflicting rules flags.

| | 9 | 10 |
|---|---|---|
| c1: W | Y | |
| c2: Z | | Y |
| a1: Z | X | |
| a2: X | | X |

FIGURE 13. Circular Table 1.

| | 9 | 10 |
|---|---|---|
| X | Y | CY |
| Z | CY | Y |

FIGURE 14. Circular Table 2.

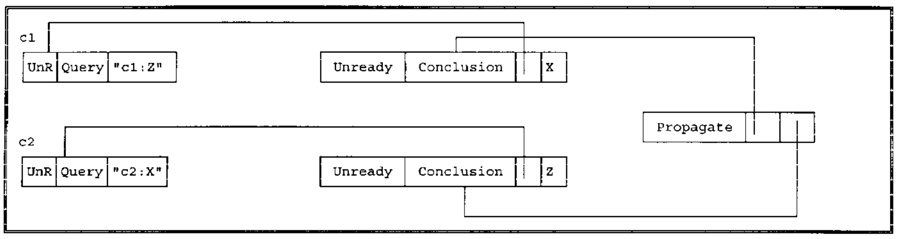

FIGURE 15. Circularity in the rules.

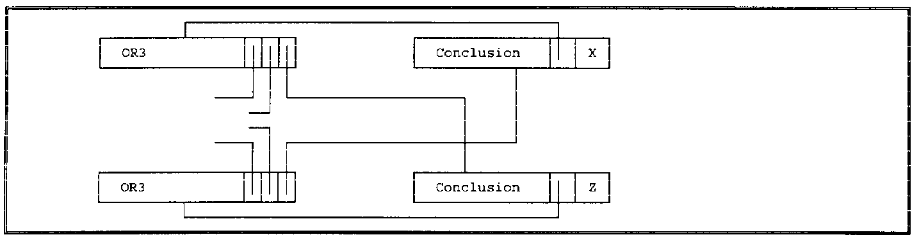

FIGURE 16. Circular rules.

or Z become true the other will also become true, but because packets may only be reduced once no infinite loop occurs. It should be noted that while circularity is often an undesired condition in a rule set, it does not *necessarily* signify an error.

## 5. Comments and conclusions

In this paper we have presented an alternative implementation of rule-based systems, in an ALICE-like graph-reduction architecture. The graph-reduction style of implementation when applied to a simple decision table specification of the knowledge base has shown several advantages over the traditional styles of implementation. The first benefit of this implementation style originates in the automatic generation of ALICE programs from the decision table by a transformation program. This relieves the programmer of the code generation overhead, placing the development emphasis upon the specification of the knowledge. Further to this, the decision tables can easily be subjected to several validation tests to identify errors or highlight possible conflicts (Murrell & Plant, 1995*c*).

Thus, there are two paths to reliability: if the decision tables are accepted as an unarguable specification or reality, and the transformation into a graph reduction program is error free, the resultant implementation of a rule-based system is guaranteed correct. Alternatively, if the decision tables may be imperfect, those errors that can be detected, will be detected during the transformation process, and those that can not be detected are usually the result of incomplete knowledge on the part of the expert, and could not be avoided by any means.

The paper followed the research of earlier workers (Nguyen *et al.,* 1985; Rushby, 1988; Murrell, 1989; Culbert, 1990; Preece, 1993; O'Leary, 1994; Plant *et al.,* 1994; Murrell & Plant, 1995*b,c*) in categorizing the kinds of error that may occur; redundancy, conflict, circularity, and acquisition defects, and examined the consequences of each of these validation error types in relation to the graph-reduction implementation. This examination revealed that graph reduction is of course subject to the same validation problems as other techniques in terms of semantic errors, but was able to offer several advantages over traditional implementations for other error types. In terms of syntactic identity, subsumption, direct redundancy, conflicting rules, and circularity, it was shown that the problems associated with traditional implementations, such as multiple firings of the same rule or infinite loops, would not occur in a graph-reduction implementation due in part to the system's inability to reduce packets which compute a rule more than once. Other categories of error such as syntactic redundancy with either identical conditions or identical conclusions, unused inputs or outputs, and missing rules were shown to be capable of identification within the transformation process from the decision tables.

Thus, we have shown a new approach to the construction of knowledge-based systems that has moved the onus of validation and verification away from testing to the specification stage, whilst accommodating a parallel processing capability in a graph reduction form that automatically raises the validity of the rule base processing through the packet-based nature of the computations, and produces a significant speed-up in processing.

## References

Akkermans, J. M., Schreiber, A. T. & Wielinga, B. J. (1994). Steps in constructing problem solving methods. *Shareable and Reusable Problem Solving Methods. Proceedings of the 8th Banff Knowledge Acquisition for KBS Workshop,* pp. 29-1–29-21. Alberta, Canada.

Alexander, J. H., Freiling, M. J., Shulman, S. J., Staley, J. L., Rehfuss, S. & Messick, S. L. (1986). Knowledge level engineering: ontological analysis. *AAAI5,* pp. 963–968. Philadelphia, PN.

ANSI (1992). Life Cycle Development of Knowledge Based Systems Using DoD-Std 2167A. ANSI/AIAA G-031-1992.

Antoniu, G. (1993). Modular design & verification of logical knowledge-bases. *AAAI Workshop on Validation & Verification of Knowledge-based Systems.* Washington, DC.

Ayel, M. & Laurant, J. P. (1991*a*). SACCO-SYCOJET: two different ways of verifying KBS. In M. Ayel & J. P. Laurant, Eds. *Validation, Verification and Test of Knowledge-Based Systems,* pp. 63–76. Chichester: Wiley & Sons.

Ayel, M., Laurrent, J. P. (1991*b*). Two different ways of verifying knowledge-based systems. In M. Ayel, J. P. Laurrent, Eds. *Validation & Verification of Knowledge-based Systems.* Chichester: Wiley & Sons.

Ayel, M. & Laurrent, J. P. (1991*c*). *Validation & Verification of Knowledge-based Systems.* Chichester: Wiley & Sons.

Becker, L. A. Green, P. G. & Bhutinager, J. (1989). Evidence flow graphs for V&V of expert systems. *NASA Contractor Report 181810,* Langley Research Center, Hampton, VA, USA.

Bench-Capon, T., Coenen, F., Nwana, H., Paton, R. & Shave, M. (1993). Two aspects of the validation and verification of knowledge based systems. *IEEE Expert,* **8,** 76–81.

Bezem, M. (1987). Consistency of rule-based expert systems. *Lecture Notes in Computer Science, 310*. Berlin: Springer-Verlag.

Boehm, B. W. (1981). *Software Engineering Economics.* Englewood Cliffs, NJ: Prentice Hall.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer,* **21,** 61–72.

Botten, N., Kusiak, A. & Raz, T. (1989). Knowledge-bases: integration, verification and partitioning. *European Journal of Operational Research,* **42,** 658–662.

Bouali, F., Loiseau, S. & Rousset, M. C. (1994). KBS correction: a proposal based on diagnostic theory. *Proceedings of ECAI-94 Workshop on Validation & Verification of Knowledge-based Systems.* Amsterdam, The Netherlands.

Breu, R. (1991). *Algebraic Specification Techniques in Object Orientated Programming Environments* Berlin: Springer-Verlag.

Breuker, J. & Van de Velde, W. (1994) (Eds). *Expertise model document part II: the commonKADS library.* ESPRIT Project P5248 KADS-II/I/VUB/TR/054/3.0/June.

Breauker, J. A., Wielinga, B. J., Van Someren, M., de Hoog, R., Schreiber. A. T., de Greef, P., Bredweg., B., Wielmaker, J., Billault, J. P., Davoodo, M. & Hayward, S. A. (1987). *Model driven knowledge acquisition: interpretation models.*

ESPRIT Project P1098 Deliverable D1 (task A1), University of Amsterdam and STL Ltd, Amsterdam, The Netherlands.

Buchanan, B. G., Barstow, D., Bechtal, R., Bennett. J., Clancy, C., Kulikowski. C., Mitchell, T. & Waterman. (1983). Constructing an Expert System. In F. Hayes-Roth, D. A. Waterman, & D. G. Lenart, Eds. *Building Expert Systems.* Reading, MA: Addison-Wesley.

Carpenter, C. L. & Murine, G. E. (1983). Measuring software product quality. Applying software quality metrics. *ASQC Quality Congress Transactions,* pp. 373–377.

Chang, C. L., Coombs, J. B. & Stachowitz, R. A. (1990). A report on the expert systems validation associate (EVA). *Expert Systems with Applications,* **1,** 217–231.

Charles, E. & Dubois, O. (1991). MELODIA: logical methods for checking K-bases. In N. Ayel & J. P. Laurant, Eds. *Validation, Verification and Test of Knowledge-Based Systems,* pp. 95–105. Chichester: Wiley & Sons.

Chen, P. (1976). The entity relationship model—towards a unified view of data. *ACM Transactions of Database Systems,* **1,** 9–36.

Clark, K. L. & Gregory, S. (1986). PARLOG: parallel programming in logic. *ACM TOPLAS,* **8,** 1–49.

Clocksin, W. F. & Mellish, C. S. (1984). *Programming in Prolog.* Berlin: Springer Verlag.

Coenen, F. & Bench-Capon, T. (1993). *Maintenance of Knowledge-Based Systems* London: Academic Press.

Cragun, B. J. & Steudel, H. J. (1987). A decision-table processor for checking completeness and consistency in rule-based expert systems. *International Journal of Man–Machine Studies,* **26,** 633–648.

Cuda, T. & Dolan, C. P. (1991). Tool aided non formal knowledge verification. *AAAI Workshop on V&V,* Anaheim.

Culbert, C. (1990) (Ed). Verification and validation of knowledge-based systems. *Expert Systems with Applications,* **1,** 197–328.

Dahl, O. J. (1990) *Object-orientation and formal techniques.* Department of Informatics, Research Report No. 138. University of Oslo, Norway.

Darlington, J. & Reeve, M. (1981). ALICE, a multiprocessor reduction machine. *ACM/MIT Conference on Functional Programming Languages and Computer Architecture,* New Hampshire.

Davis, R. & Lenat, D, (1982). *Knowledge-based Systems in AI.* New York, NY: McGraw-Hill.

de Hoog, R., Martil, R., Wielinga, Taylor, R., Bright, C. & Van de Velde, W. (1994). The common KADS model set. ESPRIT Project P5248 KADS-II/DM1.1b/UvA/018/6.0/FINAL.

ESPRI '93 (1993). *Survey and assessment of conventional software verification & validation techniques.* SPRI TR-102106, Project 3093-01, Final Report, February.

Fox, J. (1993). On the soundness and safety of expert systems. *AI in Medicine,* **5,** 159–179.

Geissman, J. R. & Schultz, R. D. (1988). Verification and validation of expert systems. *AI Expert,* **February,** 26–33.

Ginsberg, A. (1987). A new approach to checking knowledge bases for inconsistency and redundancy. *3rd Annual Conference on Expert Systems in Government,* pp. 102–111. Washington, DC, USA.

Ginsberg, A. & Rose, L. (1987). *KB-reducer: a system that checks for inconsistency and redundancy in knowledge-bases.* Technical Report, AT&T Laboratories, Holmdel, NJ, USA.

Gold, D. I. & Plant, R. T. (1994). Towards the formal specification of an expert system. *International Journal of Intelligent Systems,* **9,** 739–768.

Green, C. J. R. & Keyes, M. M. (1987). Verification and validation of expert systems. Western conference on expert systems. In U. Gupta, Ed. *Validating and Verifying Knowledge-Based Systems,* pp. 20–29. Los Alamitos, CA: IEEE Press.

Grover, M. D. (1983). A pragmatic knowledge acquisition methodology. *Proceedings of the International Joint Conference on Artificial Intelligence,* **8,** pp. 436–438. Washington, DC, USA.

Guttag, J. V. & Horning, J. J. (1978). The algebraic specification of data types. *Acta Informatica,* **10,** 27–52.

Herre, H. (1993). Semantical completeness of model based diagnosis. *Proceedings of EUROVAV'93.* Palma de Mallorca, Spain.

Hillis, W. D. (1985). *The Connection Machine.* Cambridge, MA: MIT Press.

Hoare, C. A. R. (1985). *Communicating Sequential Processes.* Englewood Cliffs, NJ: Prentice Hall.

Hollnagel, E. (1989). *The Reliability of Expert Systems.* Hemel Hempstead: Ellis Horwood.

Hors, P. & Rousset, M. C. (1993). Consistency of structured knowledge: a formal framework based on description logics. *Proceedings EUROVAV'93,* Palma de Mallorca, Spain.

Humphrey, W. (1989). *Managing the Software Process.* Reading MA: Addison-Wesley.

Ince, D. C. & Hekmatpour, S. (1987). Software prototyping—progress and prospects. *Information and Software Technology,* **29,** 8–14.

Jones, G. (1986). *Programming in Occam.* Englewood Cliffs, NJ: Prentice Hall.

Kang, Y. & Bahill, T. (1990). A tool for detecting expert system errors. *AI Expert,* **February** 42–51.

Krause, P., Fox, J., O'Neil, M. & Glowinski, A. (1993). Can we formally specify a medical decision support system? *IEEE Expert,* **8,** 56–62.

Krause, P., Byers, P. & Hajnal, S. (1994). Formal specification and decision support. *Decision Support Systems,* **12,** 189.

Krishnamurthy, C. Padalkar, S. Sztipanovits, T. & Purvis, B. R. (1987). Methodology for testing and validating knowledge bases. *Proceedings of the 3rd Conference on AI For Space Applications,* NASA JSC, Houston, TX, USA.

Lehner, P. E. (1989). Towards an empirical approach to evaluating the knowledge-base of an expert system. *IEEE Transactions on Systems, Man and Cybernetics,* **19,** 658–662.

Loiseau, S. (1989). La description et la detection des incoherences dans les bases de regles. *Proceedings of the International Conference on Expert Systems and their Applications,* Avignon, France.

Loiseau, S. & Rousset, M. C. (1993). Formal verification of knowledge bases focused on consistency: two experiments based on ATMS techniques. *International Journal of Expert Systems: Research & Applications,* **6,** 273–280.

Marcot, B. (1987). Testing your knowledge-base. *AI Expert,* **2,** 42–47.

McLelland, J. L. & Rumelhart, D. E. (1986). *Parallel Distributed Processing.* New York, NY: MIT Press.

Mehotra, M. (1993). Multi-viewpoint clustering analysis. *Workshop Notes, AAAI Workshop on Validation & Verification.* Washington, DC.

Meseguer, P. (1993). Expert system verification through knowledge base refinement. *Proceedings of the IJCAI-93,* Chamberly, France.

Miller, L. A. (1990). Dynamic testing of knowledge bases using the heuristic testing approach. *Expert Systems with Applications,* **1,** 271–281.

Minsky, M. L. & Papert, S. A. (1969). *Perceptrons.* New York, NY: MIT Press.

Morell, L. J. (1988). Use of metaknowledge in the verification of knowledge-based systems. *Proceedings of the IEA-AIE,* June, pp. 847–857.

Murrell, S. (1989). *Guide to malice.* University of Miami, Computer Science Technical Report No 1. Department of Math & Computer Science, University of Miami, FL.

Murrell, S. & Plant, R. T. (1995*a*). Formal semantics for rule-based systems. *Journal of Systems & Software* (in press).

Murrell, S. & Plant, R. T. (1995*b*). A graph reduction implementation of a production system. *Knowledge-Based Systems,* **8,** 155–160.

Murrell, S. & Plant, R. T., (1995*c*). Decision tables: formalization, validation and verification. *Journal of Software Testing, Reliability and Validation,* **5,** (9).

NASA Conference Publication 2491 (1987). *First Annual Workshop on Space Operations Automation and Robotics* (*SOAR'87*). Johnson Space Centre, Houston TX, August 5–7.

Naser, J. (1988). Nuclear power plant expert system verification & validation. *AAAI Workshop Notes on Verification & Validation of Knowledge-based Systems,* pp. 1–18, St. Paul, MN: AAAI Press.

NGUYEN, T. A., PERKINS, W. A. & LAFFERY, T. J. (1985). Checking an expert systems knowledge base for consistency and completeness. *Proceedings of the Ninth International Joint Conference on AI,* 18–23, Los Angeles, CA. pp. 375–378. August.

O'KEEFE, R. M. & O'LEARY, D. E. (1992). Expert system verification and validation: a survey and tutorial. *Artificial Intglligence Review,* **16,** 25–60.

O'LEARY, D. E., (1988). Methods of validating expert systems. *Interfaces,* **18,** 72–79.

O'LEARY, D. E., Ed. (1994). *Collected Papers of AAAI Workshops on Validation and Verification 1988–92.* Reading, MA: Wiley & Sons.

OURSTON, D. & MOONEY, R. J. (1994). Theory refinement combining analytical and empirical methods. *Artificial Intelligence,* **66,** 273–309.

PLANT, R. T. (1990). Validation and verification and testing of knowledge-based systems. *Heuristics: The Journal of Knowledge-based Systems,* **3,** 59–67.

PLANT, R. T. (1991). Utilising formal specifications in the development of knowledge-based systems. In D. PARTRIDGE, Ed. *Artificial Intelligence & Software Engineering.* Norwood, NJ: Ablex Press.

PLANT, R. T. & TSOUMPAS, P. (1994). An integrated methodology for knowledge-based system development. *Expert Systems with Applications,* **7,** 259–271.

PLANT, R. T., MURRELL, S. & MORENO, H. R. (1994). Prototype decision support system for a differential diagnosis of psychotic, mood, and organic mental disorders: Part II. *Medical Decision Making,* **14,** 273–289.

PREECE, A. (1993). A new approach to detecting missing knowledge in expert system rule bases. *International Journal of Man–Machine Studies,* **38,** 661–688.

PREECE, A. D., & SHINGHAL, R. (1991). COVER: a practical tool for verifying rule-based systems. *AAAI Workshop on Validation & Verification Notes.* Anaheim, CA.

PREECE, A. D., SHINGHAL, R., & BATAREKH, A. (1992). Verifying expert systems: a logical framework and a practical tool. *Expert Systems with Applications,* **5,** 421–436.

RADWAN, A. E., GOUL, M., O'LEARY, T. J. & MOFFITT, K. E., (1989). A verification approach for knowledge-based systems. *Transportation Research-A,* **23A,** 287–300.

REEVE, M. & ZENITH, S. E., Eds (1989). *Parallel Processing and Artificial Intelligence.* Chichester: Wiley.

ROMAN, G., GAMBLE, R. F. & BALL. W. E. (1993). *Formal derivation of rule-based programs. IEEE Transactions on Software Engineering,* **19,** 277–296.

ROUSSET, M. C. (1994). Knowledge formal specifications for formal verification: a proposal based on the integration of different logical formalisms. *Proceedings of the ECAI94,* Amsterdam, The Netherlands.

RUSHBY, J. (1988). *Quality measures and assurance for AI software.* NASA Contact Report NASI-17067, Langley Research Centre, Hampton, VA, USA.

RUSHBY, J & WHITEHURST, R. A. (1989). *Formal verification of AI software.* NASA Contract Report 18226 (Task 5), February, Langley Research Centre, Hampton, VA, USA.

SCHULTZ, R. & GEISSMAN, J. R. (1988). Bridging the gap between static & dynamic verification. In U. GUPTA, Ed. *Validating & Verifying Knowledge-Based Systems,* pp. 86–92. Los Alamitos, CA: IEEE Computer Society Press.

SEITZ, C. L. (1985). The cosmic cube. *Communications of ACM,* **28,** 22–33.

SENTAR '95 (1995). *Distributed hybrid systems V&V database annex C.* Technical Report, Sentar, Inc., Huntsville, AL.

SOLOWAY, E., BACHANT, J. & JENSEN, K. (1987). Assessing the maintainability of XCON-in-RIME: coping with the problem of a very large rule-base. *Proceedings of the 6th IJCAI,* pp. 824–829, Seattle, WA, USA.

STACHOWITZ, R. A., CHANG, C. L., STOCK. T. S. & COOMBS, J. B. (1987). Building Validation Tools for Knowledge-Based Systems. In *NASA Conference Publication 2491, First Annual Workshop on Space Operations Automation and Robotics* (*SOAR'87*), pp. 209–216. Johnson Space Centre, Houston, TX. August 5–7.

STEIB, M., SMALL, R., CASTELLS, C., & SCHOFIELD, J. (1991). Tailoring VASTT for expert system verification, validation and testing. *Workshop Notes: AAAI Workshop on V&V.* Anaheim, AL.

SUWA, M., SCOTT, A. C., & SHORTLIFFE, E. H. (1982). An approach to verifying completeness & consistency in a rule-based system. *AI Magazine,* **3,** 16–21.

Todd, B. S., Stamper, R. & Macpherson, P. (1995). A probabilistic rule-based expert system. *International Journal of Bio-Medical Computing* (in press).

Townsend, P. (1987). Flagship hardware and implementation. *ICL Technical Journal,* **5,** 575–594.

TRILLIUM (1992). TRILLIUM: telecom software product development capability assessment model. Bell Canada Quality. Technical Report Draft 2.2., Bell Canada, July.

Valiente, G. (1993). Verification of knowledge-based redundancy and subsumption using graph transformations. *International Journal of Expert Systems: Research and Applications,* **6**, 341–355.

Vanthienen, J. (1991). Knowledge acquisition and validation using a decision table engineering workbench. *World Congress of Expert Systems,* pp. 1861–1868, Orlando, FL, USA.

Vermasan, A. I. & Wergeland, T. H. (1994*a*). A formally based methodology for deriving verifiable expert systems from specifications. *Workshop Notes, AAAI Workshop on Validation & Verification.* Seattle, WA.

Vermesan, A. I. & Wergeland, T. (1994*b*). *Expert system verification and validation: issues and approaches.* Working Paper: 82/1994, Centre for Research in Economics and Business Administration, University of Oslo. Norway.

Weitzel, J. R. & Kershberg, L. (1989). Developing knowledge-based systems: reorganising the system development life cycle. *Communications of the ACM,* **32,** 482–490.

Wielinga, J. B. & Breuker, J. A. (1984). Analysis techniques for knowledge-based systems: part 1. Report 1.1 Esprit Project 12.

Zlatarova, N. (1991). VVR: a uniform framework for expert system knowledge bases verification, validation and refinement. *Workshop Notes: AAAI Workshop on V&V,* Anaheim, AL.