Rule Groupings: An Approach towards Verification of Expert Systems

Mala Mehrotra

Vigyan Inc.

30, Research Drive

Hampton, Va 23666.

mm@air12.larc.nasa.gov

Knowledge-based expert systems are playing an increasingly important role in NASA space and aircraft systems. However, many of NASA's software applications are life- or mission-critical and knowledge-based systems do not lend themselves to the traditional verification and validation (V&V) techniques for highly reliable software. Rule-based systems lack the control abstractions found in procedural languages. Hence, it is difficult to verify or maintain such systems. Our goal is to automatically structure a rule-based system into a set of rule-groups having a well-defined interface to other rule-groups. Once a rule base is decomposed into such "firewalled" units, studying the interactions between rules would become more tractable. Verification-aid tools can then be developed to test the behavior of each such rule-group. Furthermore, the interactions between rule-groups can be studied in a manner similar to integration testing. Such efforts will go a long way towards increasing our confidence in the expert-system software.

There are two main reasons why expert systems defy verification and validation efforts. First, rapid prototyping and iterative development form key features of any expert system development activity. This has led to the development of ad-hoc techniques for expert-system design without any software engineering guidelines. Second, due to the data-driven nature of expert systems, as the number of rules of an expert system increase, the number of possible interactions between the rules increases exponentially. The complexity of each pattern in a rule compounds the problem of testing even further. This makes exhaustive, or even systematic, testing of large knowledge bases infeasible. As a result, large expert systems tend to be incomprehensible, difficult to debug or modify, and almost impossible to verify or validate.

This situation is not wholly unlike that which faced traditional software development before the introduction of structured software engineering.Conventional software yields more easily to verification efforts because control is explicitly represented as procedures which can be structured to encapsulate run-time abstractions. Modules can be designed in con-

ventional software, each consisting of a manageable unit with a well-defined interface. Furthermore, procedures can be grouped into packages or objects which share an internal data structure. These units can then be subjected to unit/integration testing techniques. In expert systems, rules play a role analogous to paths through procedures. However, each rule in an expert system is data-driven, since the presence or absence of data controls the flow of execution. Hence, V&V techniques for expert systems have to view interactions between all pairs of rules. For large expert systems, this is quite difficult and can be prohibitively expensive.

Our research efforts address the feasibility of automating the identification of rule groups, in order to decompose the rule base into a number of meaningful units. Each such group can then be viewed as a procedure. Identification of the intra-group and inter-group items for a group of rules would be analogous to defining local variables and parameters for procedures in conventional software. A verification-aid tool could then test the behavior of each such unit under all possible values of inputs [1].

The grouping of rule bases can play an important role in verification and validation of flight-critical systems at NASA. In such systems one needs to identify critical regions, assert various criticality levels [2] for them, and test such regions both analytically and exhaustively. If one is able to isolate the group of rules that deal with the critical features of the problem domain, certain safety properties of the system can be verified. Knowledge of the function of a group of rules would allow us to choose the distribution of inputs in such a way that typical situations where functioning of the system is critical could be studied in isolation [7]. Moreover, if support existed for specifying what rules should not get fired under certain circumstances, backward flow analysis techniques [3] could be used to locate critical paths. An additional advantage of modularization would be the identification of modules and data items that are necessary in a degraded (fail-soft) processing mode. Validating such modules is clearly critical to confidence in the reliability of the software.

Validation of conventional software systems relies on systematic testing, since exhaustive testing of such systems is generally infeasible. Such systematic testing is based on abstractions such as procedures, functions and other control structures. Along with such explicit abstractions, other testing techniques, such as data flow and path testing, take into account implicit dependencies between different parts of the program. Rule-based systems lack the control abstractions found in procedural languages. Although the non-procedural nature of rule-based systems is sometimes cited as an advantage, the subsequent lack of control abstractions makes it difficult to verify and maintain these systems.

Towards this goal, we would like to structure a rule base into a set of groups consisting of related rules. We have taken a pattern-matching approach for this grouping of rules. In this approach, the commonality of items in the rules determines the distance between them. Our rule grouping process consists of three stages. First, the distance between each pair of rules is computed and stored in a distance matrix. In the second stage, the computed distances are modified so that all distances satisfy the triangle inequality. That is, we replace the distance between two rules by a shorter distance, if there exists an intermediate rule through which a shorter path can be created. This procedure thus extracts transitive dependencies between rules. Finally, we apply a clustering algorithm to form our groups.

In [4, 5, 6], we have studied three types of distance metrics and two approaches to clustering. The first one is an automatic clustering algorithm based on graph-partitioning approaches. The second requires the user to designate certain rules as "primary rules" or "seed rules" around which the clustering algorithm will form groups. These primary rules typically reflect key concepts from the domain; thus, the resulting clusters correspond closely to the user's conceptual model of the problem domain. Two independent evaluation criteria were developed to measure the effectiveness of the grouping strategies.

Based on our results so far, we believe a comprehensive set of distance metrics can be designed which would be effective in grouping all types of rule bases. By applying our grouping tool to more complicated and larger rule bases, we hope to obtain more insight into parameters that play a critical role in grouping. A significant outcome from our study will be the formulation of software engineering guidelines for the design of rules, which would promote the grouping process without sacrificing programming flexibility. In fact these guidelines may actually aid in programming by providing guidance and discipline, similar to the aid that structured programming gives in maintaining intellectual control of traditional software. This is also analogous to work in language design which is driven by the ease of proof rules for verification. A good style of rule-based programming will make the underlying relationships between key concepts more transparent and easier to understand and verify.

In future, we intend to study the interplay of distance metrics, clustering criteria, objective functions to be optimized and software engineering guidelines on grouping. We would also like to formulate rigorous evaluation criteria to judge the quality of groups formed. A handle on the definition of what constitutes a "good grouping" will feed back into the criteria to be used for formation of good groups. A set of software engineering requirements could then be laid out that are necessary for producing useful partitions in a rule base from the point of view of testing them. The "quality" of grouping is very much dependent on the motivation behind grouping. For maintainability and ease of comprehension of the rule base one would want the groupings to "match" human expectations. However, for testing and verification, this aspect is less important. In fact, one could argue that "unnatural" groupings would force the software engineer to see the rule base in a new light which could give insight into its behavior - *particularly* its unexpected behavior.

Our research plans are designed to give us insight into the factors that lead to meaningful rule groupings after which we intend to develop automatic verification aid tools for unit/integration testing of well-partitioned rule bases. Each partition or group obtained through our grouping process can be viewed as a procedure. Our software tool would be extended to allow identification of the common intra-group and inter-group items for a group of rules, which would be analogous to local variables and parameters for procedures in conventional software. Such groups can be formed into software modules with "firewalls," having well-defined inputs and outputs. Formation of such modules can be seen as the first step towards managing their complexity for verification and validation. The interface of the modules can then help in the automatic generation of test suites required for verification and validation. Validating such modules is clearly critical to confidence in the reliability of the knowledge-based system software.

References

[1] C. Culbert and R. T. Savely. Expert system verification and validation. In *Proceedings, Validation and Testing Knowledge-Based Systems Workshop*, August 1988.

[2] S. C. Johnson. Validation of highly reliable, real-time knowledge-based systems. In *SOAR 88 Workshop on Automation and Robotics*, July 1988.

[3] N. G. Leveson. Safety-critical software development. In T. Anderson, editor, *Safe & Secure Computing Systems*, chapter 9, pages 155–162. Blackwell Scientific Publications, 1989.

[4] M. Mehrotra. Rule groupings: A software engineering approach towards verification of expert systems. Technical Report NASA CR-4372, NASA Langley Research Center, Hampton, VA., May 1991.

[5] M. Mehrotra and S. C. Johnson. Importance of rule groupings in verification of expert systems. In *Notes for the AAAI-90 Workshop on Verification, Validation and Testing of Knowledge-Based Systems*, July 1990.

[6] M. Mehrotra and S. C. Johnson. Rule groupings in expert systems. In *Proceedings, First CLIPS Users Group Conference*, Aug 1990.

[7] D. L. Parnas, J. van Schouwen, and S. Po Kwan. Evaluation of safety-critical software. *Communications of the ACM*, 33(6):636–648, June 1990.