Applications of Rule-Base Coverage Measures to Expert System Evaluation

Valerie Barr
Department of Computer Science
Hofstra University
Hempstead, NY 11550
vbarr@magic.hofstra.edu

Abstract

Often a rule-based system is tested by checking its performance on a number of test cases with known solutions, modifying the system until it gives the correct results for all or a sufficiently high proportion of the test cases. This method cannot guarantee that the rule-base has been adequately or completely covered during the testing process. We introduce an approach to testing of rule-based systems which uses coverage measures to guide and evaluate the testing process. In addition, the coverage measures can be used to assist rule-base pruning and identification of class dependencies, and serve as the foundation for a set of test data selection heuristics. We also introduce a complexity metric for rule-bases.

Introduction

In this work we examine ways to overcome limitations of common methods for rule-based expert systems evaluation. In particular, we would like the rule-base testing process to exercise every inference chain or provide information about the failure of the testing process to do so. Usual approaches to verification and validation (V&V) of rule-based systems typically employ a static structural analysis (verification) method to detect internal inconsistencies, followed by a dynamic, functional, validation in which system behavior is compared with expected behavior. The weakness of a strictly functional approach to validation is that the test data available may not adequately cover the rule-base, and, at best, limited information about the coverage will be obtained. System performance statistics are usually presented as if they apply to the entire rule-base, rather than just to the tested sections, which can lead to false estimates of system performance.

A rule-base coverage analysis method can enhance functional evaluation of rule-bases by providing information that can lead to identification of both incompleteness in the test data and potential errors in the rule-base. Our approach carries out structural analysis of the rule-base using five rule-base coverage measures to identify sections not exercised by the test data. This makes it possible to improve completeness of the test suite, thereby increasing the kinds of cases on which the rule-base has been tested. The coverage information can be used to both assess the testing which has taken place and guide further testing. In addition, the coverage information can be used to assist rule-base pruning and identification of class (goal) dependencies.

Coverage analysis, obtained during execution of test cases, can be used to highlight problems in both the test suite and the rule-base, thereby pointing to areas in which we cannot guarantee or predict the system's performance. In typical rule-base testing a set of test cases with known results is run through the rule-base, and actual results are compared to expected results. However, we must also consider completeness of the test set and coverage of the rule-base by the test data, as indicated in Figure 1. *Completeness* of the test set refers to the degree to which the data represents all types of cases which could be presented to the system under intended conditions of use. *Coverage* of the rule-base refers to how extensively possible combinations of inference relations are exercised during test data evaluation. In the trivial case, with a correct rule-base and a complete test suite, the test data would completely cover the rule-base, all actual results would agree with expected results, and we could predict completely correct performance of the rule-base in actual use.

The system performance indicated by the comparison of actual and expected results is relevant only for the tested sections, while performance in the untested sections can not be predicted. Finally, if the test set is made up of cases which were selected from a larger population, coverage information can be used to direct re-sampling of the population in order to choose additional cases for the test suite, as shown in Figure 2.

More commonly there may be errors and incompleteness in the rule-base, as well as inadequacies in the test data. If the system is judged based only on a comparison of actual and expected results, the rule-base could perform well on the test data, but actually contain errors which are not identified due to incompleteness of

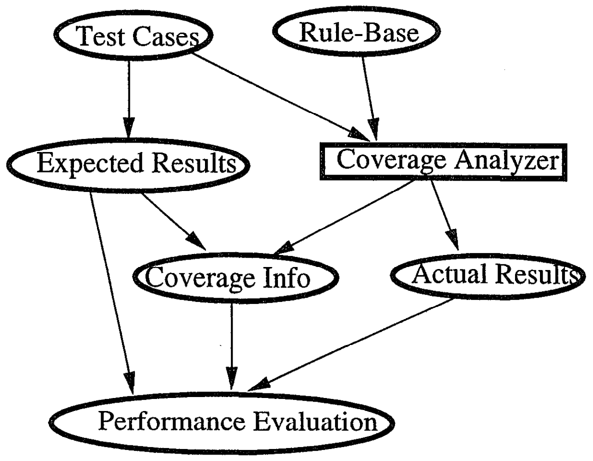

Figure 1: Rule-Base Evaluation with Coverage Analysis.

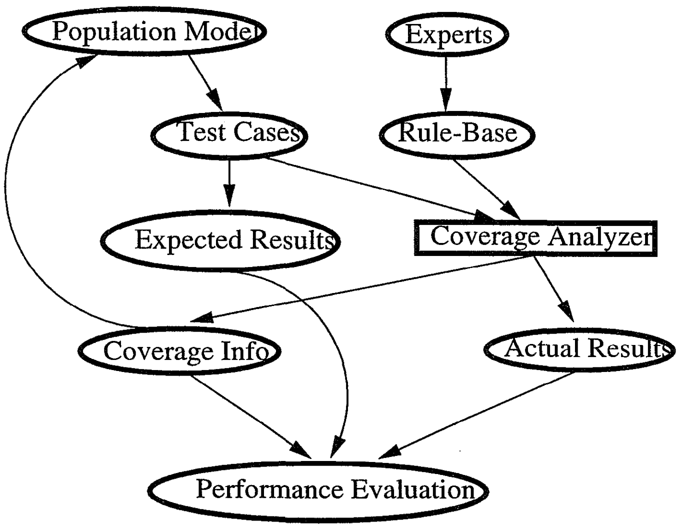

Figure 2: Evaluation with Coverage Analysis and Data Re-sampling.

the test data. This could lead to a false prediction of correct performance on all cases, when in fact we cannot make any accurate prediction about performance of the rule-base in those areas for which there is an absence of test data.

Analysis of the coverage resulting from execution of the test data allows for several types of rule-base analyses and refinements. Lack of coverage results from either incompleteness of the test data or errors in the rule-base. Using coverage measures and heuristics for test case selection allows us to improve the completeness of the test suite and increase coverage of the rule-base, thereby increasing the kinds of cases for which the rule-base has been executed during testing.

This work builds on both coverage-based testing methods for procedural software (see (Adrion, Branstad, & Cherniavsky 1982) for a review of methods and (Frankl & Weyuker 1985) for a data-flow approach to testing) and earlier work on rule-base analysis. Early approaches for rule-base analysis carried out only verification or validation. A number of systems, such as ONCOCIN (Suwa, Scott, & Shortliffe 1982) and others, carry out only verification. A number of dynamic analysis tools were also developed, such as TEIRESIAS (Davis 1984), which aids in debugging and knowledge acquisition, and SEEK2 (Ginsberg, Weiss, & Politakis 1985), a rule-base refinement tool. As the field of verification and validation matured, criticisms of the existing approaches (Andert 1993; Nazareth 1989; Andert 1992; Rushby 1988) have led to the development of a number of new approaches, of which the work described here is one. COVER (COmpleteness VERifier) (Preece 1989; Preece & Shinghal 1992) is another approach which combines both a functional and structural analysis of the rule-base.

In addition to the hierarchy of rule-base coverage measures, we describe a rule-base representation which facilitates application of the coverage measures; a set of heuristics for re-sampling the population of available test cases, based on coverage information; strategies for rule-base pruning and identification of class-dependencies; a rule-base complexity metric. In other work (Barr 1996) the utility of the above is illustrated extensively using rule-bases which were prototypes for the AI/RHEUM system (Kingsland 1985).

Testing with Rule-Base Coverage Measures

The first step in rule-base testing with coverage measures is to build a graph representation of the rule-base. Our method uses a directed acyclic graph (DAG) representation. We assume a generic propositional rule-base language (Barr 1996) into which other rule-base languages can be translated. During construction of the DAG, pairwise redundant rules, pairwise simple contradictory rules and potential contradictions (ambiguities) are identified. After DAG construction is complete, static analysis (verification) of the rule-base reports dangling conditions (an antecedent component that is not defined as a finding and is not found as the consequent of another rule), useless conclusions, and cycles in the rule-base. At this point the rule-base could be modified to correct any static problems.

Next dynamic analysis is carried out using test cases. As test cases are processed, one or more of several rule-base coverage measures (RBCMs) can be reviewed in order to determine the quality of the test data supplied thus far. Additional information about the rule-base and its testing can also be used to guide the selection of future test data. The tester would start by providing sufficient test data to satisfy the simplest functional measure (conclude each class of the system) and proceed to the more difficult structural measures. Finally, if the user is unable to provide sufficient data to attain the desired degree of rule-base coverage, the user can use the DAG representation to synthesize data, which can then be reviewed by an expert to determine if the data represents a valid case in the problem domain.

This testing approach, described more fully below, has been implemented in the TRUBAC tool (Testing with RUle-BAse Coverage) (Barr 1996; 1995).

Rule-base Representation

The DAG representation, which allows identification of sections which have and have not been covered, or exercised, by the test data, is based on the AND/OR graph implicit in the rule base (Meseguer 1991). The DAG has a source node, corresponding to working memory, and a sink node, corresponding to success in reaching one of the classes (diagnoses or goals) of the system. The two types of interior nodes are sub-class nodes (SUBs) and operator nodes. Sub-class nodes represent non-class rule consequents (intermediate hypotheses of the system). Operator nodes are used to represent the allowable operators AND, OR, and NOFM[1]. These operator nodes represent the fact that the conjunction and/or disjunction of multiple components of an antecedent must be true in order for the conclusion of a rule to be entered into working memory. There are edges from the source to each finding and from each class to the sink. The antecedent of each rule is represented by a subgraph which connects findings and sub-class nodes to operators as indicated by the antecedent. Each antecedent-consequent connection represented by a rule is also represented by an edge from the subgraph for the antecedent to the node for the consequent.

For example, the representation of the two rules

If P1 *&* P2 *then* R1
If R1 *&* R2 *then* Q

is shown in Figure 3. **P1**, **P2** and **R2** could be findings

[1]NOFM nodes represent the construction "if N of the following M things are true, then...", which is a feature of EXPERT (Weiss *et al.* 1987) rule-bases.

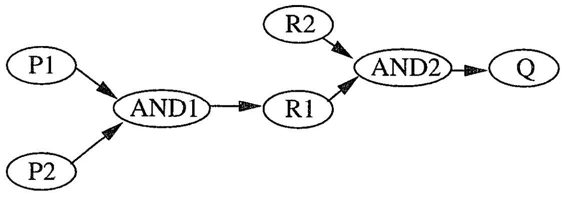

Figure 3: DAG representation of two related rules.

or sub-class nodes, while **R1** and **Q** are either class or sub-class nodes. The complete graph of a rule-base is constructed by linking together the individual structure for successive rules. Using this framework we can easily represent rule-bases, including those in which the certainty factors are hard coded within the consequent definitions.

Rule-base coverage measures

Each reasoning chain through the rule-base corresponds to a sub-DAG of the DAG representation. That is, if we were to mark all nodes and edges in the DAG that correspond to the rules fired during a particular chain of inferences, we would include: the source node; the sink node; all nodes corresponding to the findings involved; the class concluded; all nodes corresponding to antecedent components, operators, and rule consequents; and all edges involved in antecedent-consequent links formed by the rule connections used in the reasoning chain. A rule firing involves all edges and nodes in the graph that corresponds to the rule. An *execution path* is the sub-DAG that corresponds to all the rules fired along a particular reasoning chain executed due to a specific set of findings.

Ideally, in testing a rule-base, we would like to provide sufficient test data to cause every possible execution path of the DAG to be traversed, which corresponds to the firing of all rules in every combination possible. Since this is usually not reasonable in the testing environment, we propose the following hierarchy of rule-base coverage measures (RBCMs) for rule-based systems in order to guide the selection of test data and give an objective measure of how well a test suite has covered the rule-base.

1. **Each-class** Satisfied if the test data causes traversal of one execution path to each class of the system.
2. **Each-hypoth** Satisfied if the test data causes traversal of execution paths such that each sub-class is reached, as well as each class.
3. **Each-class-every-sub** There may be many execution paths that connect each sub-class to each class (and no execution paths for some sub-class to class combinations). This coverage measure is satisfied if, for each sub-class to class combination connected by some execution path, at least one execution path which includes the combination is executed.

4. **Each-class-every-finding** There may be many execution paths that connect each finding to each class (and no execution paths for some finding-class combinations). This coverage measure is satisfied if, for each finding-class combination connected by some execution path, at least one execution path which includes the combination is executed.
5. **All-edges** Satisfied if the data causes traversal of a set of execution paths such that every inference relationship (every edge in the graph) is utilized along some inference path. While this will not guarantee that all rules will be used in every combination possible, it will guarantee that every rule is used in all possible ways along some execution path.

In a very complex rule-base, or one for which there is very little test data, using these criteria will help the tester determine the ways in which the data has been deficient in testing portions of the rule-base. The coverage measures provide information that will lead to the acquisition or development of additional test cases, or will lead to the discovery of errors in the rule-base.

Additional Applications of Coverage Analysis

In addition to providing information about the testing process itself, the coverage analysis can be used to enhance testing and facilitate other kinds of rule-base analysis, as described below.

Heuristics for Test Data Selection

It is often the case that there is a large pool of available test data, from which a subset of cases must be selected for the test suite due to the lengthy time necessary to run all the available cases. However, a poor selection of cases will lead to an incomplete test set. We can use the coverage information as the foundation of a set of heuristics for test data selection, in order to construct a test set that will maximize rule-base coverage.

To date we have restricted our work to classification systems, so each goal of the system is a class and we consider each intermediate hypothesis to be a sub-class. Assuming that we have some degree of meta-knowledge about the make-up of individual test cases (e.g., what facts or intermediate hypotheses are involved in each case), a set of heuristics for data selection is:

1. For each class, select a test case which concludes only that class.
2. For each class not yet tested, select a test case which will conclude it (and additional classes).
3. Select test cases which will conclude unused sub-classes.
4. Select test cases which will cover sub-class to class relations, and direct finding to class relations for findings which do not lead to an intermediate sub-class.

	PM	PSS	SLE	MCTD	RA
PM		0	0	22.22	22.22
PSS	0		16.67	16.67	16.67
SLE	0	9.09		0	0
MCTD	18.18	9.09	0		**36.36**
RA	28.57	14.29	0	**57.14**	

Table 1: Overlap of sub-classes for class pairs.

5. Select test cases which will cover finding to class relations for findings which represent alternative ways to conclude sub-classes. That is, while a sub-class to class relation may have been covered, there may be multiple ways to conclude the sub-class.

Other work in this area (Barr 1996) shows that the use of the coverage information, along with meta-knowledge about the pool of available cases, is critical to selecting the test cases in a way that contributes to a useful test of the rule-base. This in turn gives greater assurance that the system has been tested and works correctly for the situations that we expect to encounter most often, as well as clearly identifying those parts of the system that still require more examination, testing, or refinement.

Class Dependence

Another aspect of rule-base analysis or evaluation is the identification of class dependencies, in which the rules that lead to the conclusion of one class are also highly involved in the conclusion of another class. If two classes, C1 and C2, are dependent, and we have a large number of test cases that we know will be classified as C1, it may be that some of those cases will also be classified as C2. In this way, testing for one class will also help us achieve coverage for the other. Furthermore, if C1 and C2 are dependent classes and we change rules for C1 then we should both rerun test cases which we know conclude C1, and rerun test cases which we know conclude C2 in order to verify that changing rules for C1 did not inadvertently affect the system's ability to properly classify C2 as well.

While shared findings does not imply rule sharing for class pairs, to determine if two classes do have rules in common we look at sharing or overlap among the sets of sub-classes that can lead to a class. If there is a high degree of overlap among sub-classes then it does imply overlap among the rules and a degree of dependency between those classes.

This information can be obtained quite immediately from data which is collected while the DAG representation is built. We accumulate for each class a list of all the sub-classes which can help to conclude the class, and then compare these lists for pairs of classes. Table 1 shows the overlap figures for a small prototype of the AI/RHEUM system for rheumatology diagnosis (Kingsland 1985). The figures of interest are those for RA and MCTD (rheumatoid arthritis and mixed connective tissue disease). These figures indicate that 57% of the sub-classes which lead to RA also lead to MCTD, while 36% of the sub-classes which can lead to MCTD also lead to RA. The asymmetry in these figures results from the fact that the absolute number of sub-classes which can lead to the classes is different. These figures would indicate that modifications made to the rules for RA would likely also affect the performance of the system on cases that should be classified as MCTD, while there would be a lesser affect on the system's classification of cases as RA if the rules for MCTD were modified.

These results are consistent with those found in (Indurkhya 1991), which uses Monte-Carlo simulation-based techniques to carry out rule-base evaluation. However, in our approach we obtain the information without the overhead of executing the rule-base.

Rule-Base Pruning

Once a rule-base has been constructed, it may be possible that not all the rules are necessary for the rule-base to perform correctly. For example, during incremental development some early rules may be supplanted by rules added later. If the system can be pruned by removing rules or components within rules, and the performance on the test cases is not affected, then it is possible that there were unnecessary rules or that the test cases are not adequate to evaluate the entire rule-base (Indurkhya 1991).

We use coverage information to focus the pruning steps on sections of the rule-base which have not been executed by the test data. A section of the rule-base that is never used or executed during evaluation of the test cases contains unnecessary rules or the test suite is not sufficiently rich and additional test cases are necessary to cause execution of inference paths through the unexecuted section of the rule-base. It will be up to the developer and/or the expert to decide whether the proper approach is to prune or to add test cases. If we believe that the test suite is truly representative of cases that will be found in the application environment, and the rule-base performs correctly on the test set, then it is more likely that the uncovered portion of the rule-base is in fact unnecessary and can be pruned. If the test set is complete and the rule-base performs incorrectly then the uncovered sections of the rule-base are candidates for rule refinement. The coverage measures will provide information about why the rules were never used, based on findings and sub-classes which appear in rule antecedents but are not present in any test cases. Removal of these antecedent components from the rules may generalize them sufficiently that they will correctly handle some of the test cases and will be covered by some existing portion of the test suite.

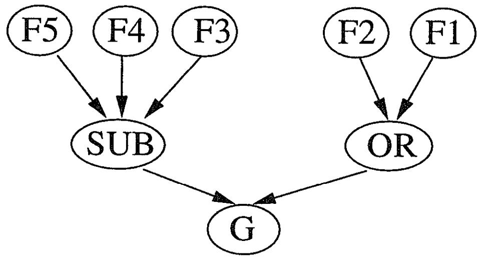

Figure 4: Graph of rule-base with OR and SUB.

A Metric for Rule-Based Systems

The graph representation proposed here serves as a suitable foundation for a complexity metric to predict or measure the complexity of the system and of the testing process. The actual number of execution paths is based on the logical relationships in the rule-base, using the following mechanism:

- For each class node, count one path for each edge into the class node. In Figure 4 there are 2 paths based on in-edges at class node G.
- For each OR node, count one additional path. In Figure 4 we count one more path to G at the OR node.
- For each SUB node, consider the number of parent edges. As with an OR node, each parent edge represents a possible path which concludes the sub-class. One of these paths will be counted when the classes are considered, and we count additional paths for the additional parent edges. In Figure 4 there are a total of 5 paths to G: 2 at G, 1 more at the OR node, and 2 more at the SUB node.
- for AND nodes no additional paths are added.

This execution path metric can serve a number of purposes in rule-base development and analysis. The total number of execution paths represents the maximum number of test cases needed for complete coverage of the rule-base according to the strongest rule-base coverage measure (**All-edges**). However, usually the actual number of data sets needed will be less than the number of execution paths, since often, particularly in diagnosis systems, one test set may cover a number of execution paths to different diagnoses.

Conclusion and Future Work

In this work we show that rule-base performance evaluation can be misleading unless care is taken to identify problems with both the test data and the rule-base. Both the test data and the rule-base can be improved by using information about the extent to which the test data has covered the rule-base under test. This work can be extended in a number of directions. A quantitative performance prediction can be developed based on performance of the system on test cases, a measure of how well the test data covers the rule-base, and a measure of the degree to which the test set is representative of the population for which the system is intended. A second area of extension is for systems which have dynamic computation of certainty factors, which requires modification of the rule-base coverage measures (Barr 1996) as well as changes to the implementation and the data selection heuristics. Finally, it may also be possible to extend this approach to analysis of Bayesian belief networks. The probabilistic relationships between nodes of the network, with information flow which is bi-directional along the arcs, and nodes which may be dependent in some contexts and independent in others, significantly complicates this task.

References

Adrion, W.; Branstad, M.; and Cherniavsky, J. 1982. Validation, verification, and testing of computer software. *ACM Computing Surveys* 14(2):159–192.

Andert, Jr., E. 1992. Automated knowledge-base validation. In *Workshop on Validation and Verification of Knowledge-Based Systems.* San Jose, CA: Tenth National Conference on Artificial Intelligence.

Andert, Jr., E. 1993. Integrated design and V&V of knowledge-based systems. In *Working Notes from Workshop on Validation and Verification of Knowledge-Based Systems.* Washington, D.C.: Eleventh National Conference on Artificial Intelligence.

Barr, V. 1995. TRUBAC: A tool for testing expert systems with rule-base coverage measures. In *Proceedings of the Thirteenth Annual Pacific Northwest Software Quality Conference.*

Barr, V. 1996. *Applications of Rule-Base Coverage Measures to Expert System Evaluation.* Ph.D. Dissertation, Rutgers University.

Davis, R. 1984. Interactive transfer of expertise. In Buchanan, B. G., and Shortliffe, E. H., eds., *Rule-Based Expert Systems.* Reading, MA: Addison-Wesley. 171–205.

Frankl, P., and Weyuker, E. 1985. A data flow testing tool. In *Proceedings of IEEE Softfair II.*

Ginsberg, A.; Weiss, S.; and Politakis, P. 1985. SEEK2: A generalized approach to automatic knowledge base refinement. In *Proceedings of IJCAI-85.*

Indurkhya, N. 1991. Monte-carlo simulation-based evaluation and refinement of rule-based systems. Technical Report DCS-TR-277, Department of Computer Science, Rutgers University.

Kingsland, L. 1985. The evaluation of medical expert systems: Experiences with the AI/RHEUM knowledge-based consultant system in rheumatology. In *Proceedings of the Ninth Annual Symposium on Computer Applications in Medical Care*, 292–295.

Meseguer, P. 1991. Structural and performance metrics for rule-based expert systems. In *Proceedings of the European Workshop on the Verification and Validation of Knowledge Based Systems*, 165–178.

Nazareth, D. 1989. Issues in the verification of knowledge in rule-based systems. *Int. J. of Man-Machine Studies* 30:255–271.

Preece, A., and Shinghal, R. 1992. Analysis of verification methods for expert systems. In *Workshop on Validation and Verification of Knowledge-Based Systems.* San Jose, CA: Tenth National Conference on Artificial Intelligence.

Preece, A. 1989. Verification of rule-based expert systems in wide domains. In *Research and Development in Expert Systems VI, Proceedings of Expert Systems '89*, 66–77. London: British Computer Society Specialist Group on Expert Systems.

Rushby, J. 1988. Quality measures and assurance for AI software. Technical Report SRI-CSL-88-7R, SRI International, Menlo Park, CA.

Suwa, M.; Scott, S.; and Shortliffe, E. 1982. An approach to verifying completeness and consistency in rule-based expert system. *AI Magazine* 3(4):16–21.

Weiss, S.; Kern, K.; Kulikowski, C.; and Uschold, M. 1987. A guide to the EXPERT consultation system. Technical Report CBM-TR-94, Department of Computer Science, Laboratory for Computer Science Research, Rutgers University.