Issues in Designing Rule-based Systems for Integrated Evaluation

P. G. Chander*
Bell Laboratories,
Business Communications Systems R&D,
Denver, Colorado 80234,
gokul3@bell-labs.com

T. Radhakrishnan, and R. Shinghal
Department of Computer Science,
Concordia University,
Montreal, Canada H3G 1M8
{krishnan, shinghal}@cs.concordia.ca

Abstract

The design of rule bases is plagued by various anomalies due to lack of structured development, and lack of a formal reference for the acquired knowledge. Often, the lack of a clear link connecting the functional requirements, the design, and the implementation levels of a rule-based system makes it difficult to analyze the rule base in terms of how well it represents the acquired knowledge. This also makes the existing verification and validation (V&V) tools to stand alone and to be isolated from development. Integration of V&V in a rule-based system design and development life cycle is being recommended by contemporary researchers for quality and reliability improvement. However, V&V processes for rule-based systems have been accepted to be non-trivial: methods that are general enough for comprehensive anomaly detection require impractical amounts of computation, and special methods (reduced computation for V&V) lack in their scope and applicability. In this work, we outline the use of a knowledge acquisition strategy called goal specification and its role as a link that connects the functional, design, and implementation stages of a system. We then identify various issues that affect integrating evaluation into development and how goal specification can facilitate handling these issues.

Key words: Knowledge acquisition & representation, Rule base design, Integrated evaluation.

*Author for correspondence. The work reported in this paper was done while the author was at Concordia University, Montreal.

Introduction and Motivation

Conceptually, the design and development of a rule-based system is a combination of comprehension, mapping, and encoding of the knowledge from the domain expert into a set of rules (Yost 1993; Chander, Radhakrishnan, & Shinghal 1997a).However, a variety of anomalies plague rule bases, and detecting such anomalies requires a variety of evaluation procedures (O'Keefe & Lee 1990; O'Keefe & O'Leary 1993; Kiper 1992; Guida & Mauri 1993).

The evaluation processes for rule-based systems have been traditionally classified as verification, validation, and performance analysis (or measurement) (Zlatareva & Preece 1994; Guida & Mauri 1993). Verification detects the anomalies due to redundancy, deficiency, circularity and ambivalence; functional validation tests the conformance of a system with its functional requirements; structural validation checks whether the observed functional performance is the result of the correct structure (rule interactions) and tries to capture the system structure in a well defined way; and performance assessment measures the adequacy, optimality, and test case coverage of the system. Not only are these processes non-trivial, but system developers are often unclear where they fit in a system life-cycle (Andert Jr 1993; Hamilton, Kelley, & Culbert 1991).

Our motivation arises partly from our experience in the design and evaluation of rule-based systems, and from the increasing emphasis on integrating evaluation and on reducing evaluation costs (Meseguer 1992; Lee & O'Keefe 1994; Chander 1996). Contemporary researchers believe that quality and reliability improvement for rule-based systems can be obtained through formal approaches to system construction and integrated evaluation (Plant 1992; Krause *et al.* 1993; Lee & O'Keefe 1994). Integrating evaluation in a system's life-cycle, however, is non-trivial as costs can be prohibitive if all tests are automatically repeated for every modification to the system. It has also been pointed out that a development methodology that does not encourage integrating V&V (or emphasizes only on function-based evaluation) is unsatisfactory as functional or random testing approaches alone do not test a system sufficiently (Plant 1992). Based upon these observations, and from our experience, we believe that integrating evaluation in a system's life-cycle should emphasize incremental evaluation procedures and should take into account the the role of a system's structure for its evaluation (Chander, Shinghal, & Radhakrishnan 1997).

The work reported in this paper is part of a larger frame-work for rule-based systems that is aimed towards providing "testable designs." More specifically, our objective is to provide models for integrating the evaluation processes into a rule-based system life-cycle. We argue that such integration of design and evaluation can occur only by providing a link between the knowledge acquisition, the design, and the implementation stages of a rule-based system (Chander, Shinghal, & Radhakrishnan 1997). Unless there is a formal link between these stages, it is difficult to analyze how effectively the rule base represents the acquired knowledge (Chander, Radhakrishnan, & Shinghal 1997a). In this paper, we outline how such a link encourages integrated evaluation.

The paper is organized as follows. In section 2, we describe our approach to knowledge acquisition called **goal specification**, and report out experince on its use at both the design and evaluation level. In section 3, we outline the various issues that confront a developer in trying to integrate V&V processes into a system's life-cycle, and argue how goal specification can play a useful role in tackling such issues. Section 4 provides concluding remarks.

Rule Base Design

A rule base will consist of a set of rules and hypotheses. The hypotheses are atoms of first order logic that capture a concept, or inference associated with the domain. For example, the designation of "professor in a university" is typically captured as $PROFESSOR(x, y)$, where x refers to the professors name, and y to the university. However, this alone does not portray the importance of this concept (hypothesis) in relation to solving problems in its domain. Thus, capturing knowledge in terms of atoms for rule encoding is necessary, but not sufficient. The insufficiency arises because in diagnostic domains where rule-based systems are typically employed, the required knowledge should capture the *progress* of problem solving in the domain through a set of concepts/hypotheses so that the rule base can be designed to reflect this progress through well defined rule sequences.

In a goal specification approach towards knowledge acquisition, the domain expert, in conjunction with a knowledge engineer, specifies a set of states that are needed to solve problems from the domain. Typically, the domain expert specifies concepts associated with the domain that serve as mile posts of problem solving, and the knowledge engineer translates these concepts into a set of first order logic atoms that capture the intent of the domain expert. Such states are called *goals*. In addition, the domain expert also specifies constraints associated with the domain called *inviolables*; an inviolable is a conjunction of hypotheses such that all of them cannot be true at the same time. An example of an inviolable is $MALE(x) \wedge FEMALE(x)$; it is obvious that no goal or part of a goal should be an inviolable.

Goal specification is a rigorous process that involves the mapping of the acquired concepts into goal descriptions, and is in conformance with the contemporary frame works for knowledge acquisition (Yost 1993; Krause *et al.* 1993). Every goal in a goal specification, when translated into a first order logic formula, consists of a conjunction of hypotheses. The hypotheses that are used as goal compositions are called goal atoms, in order to contrast them with the other hypotheses in the system that may be needed (for rule base coding) called non-goal atoms. Solutions to problems in the domain are also clearly demarcated at the time of goal specification. Thus, it is possible to partition the goal specification into two goal classes: *intermediate* goals and *final* goals. Typically, the intermediate goals are those that are achieved in order to infer a final goal.

A design issue arises in mapping the intermediate and final goals arrived at during specification to the intermediate and final hypotheses in a rule base. A *design scheme* is a satisfiable restriction imposed in mapping intermediate and final goals to intermediate and final hypotheses in the rule base.

Definition 1 (Design Scheme) *A design scheme $\mathcal{D}$ is an ordered pair of partial mappings $< \mu_1, \mu_2 >$ such that*

$$\mu_1 : \mathcal{F} \mapsto H_i \cup H_f$$
$$\mu_2 : \mathcal{I} \mapsto H_i \cup H_f$$

where $\mathcal{F}$ is the set of final goals, $\mathcal{I}$ is the set of intermediate goals, H_i is the set of intermediate hypotheses, and H_f is the set of final hypotheses. The mapping is partial because not all hypotheses need be goal constituents.

Though several design schemes can be identified from the above mapping (see Figure 1), only a subset of these schemes whose mapping constraints make sense are allowed (for example, schemes with constraint F4 in Figure 1 where final goals are composed only of intermediate hypotheses is counterintuitive to the notion of a final goal, and are not allowed.)[1] For more details on the properties of the design schemes, and their impact of choosing a design scheme on system evaluation, see (Chander 1996; Chander, Radhakrishnan, & Shinghal 1997a).

[1]This does not mean final goals cannot contain intermediate hypotheses. If some final goal representation requires use of intermediate hypotheses (typically in synthesis domains), use mapping constraints F2 or F3.

Hypotheses in a final goal f	Hypotheses in an intermediate goal i
(F1) All are final hypotheses.	(I1) All are final hypotheses.
(F2) At least one hypothesis is final.	(I2) At least one hypothesis is final.
(F3) At least one hypothesis is intermediate.	(I3) At least one hypothesis is intermediate.
(F4) All are intermediate hypotheses.	(I4) All are intermediate hypotheses.
(F5) No constraints.	(I5) No constraints.

Figure 1: The choices for constituent hypotheses in a final goal f and an intermediate goal i, provided neither f nor i contains an inviolable.

A rule base constructed based on a given goal specification of a domain implements the problem solving by rule sequences (called *paths*) that progress from goal(s) to goal (Chander, Radhakrishnan, & Shinghal 1996). The complete knowledge of the domain is represented in a system via the goals and the paths inferring these goals causing a progression in problem solving. This progression can be conceptually portrayed as an AND/OR graph called the *goal graph of the domain*, or, simply, the goal graph.The extent to which a given rule base realizes the acquired knowledge of goal inference is reflected by the paths in the rule base; they are collectively said to portray the **structure** of the rule base.

Definition 2 (Rule Base Structure) *The structure of a rule base (or, simply structure) is defined as* $< \mathcal{G}, \Sigma >$ *where* $\mathcal{G}$ *is the goal specification of the domain, and* Σ *is a set of paths in the rule base such that,*

$$(\forall \sigma \in \Sigma)(\exists G, g)(G \subset \mathcal{G})(g \in \mathcal{G})\ G \wedge \sigma \vdash g$$

A sample goal graph and a rule base path are shown in Figure 2.

Goal Specification: Applications

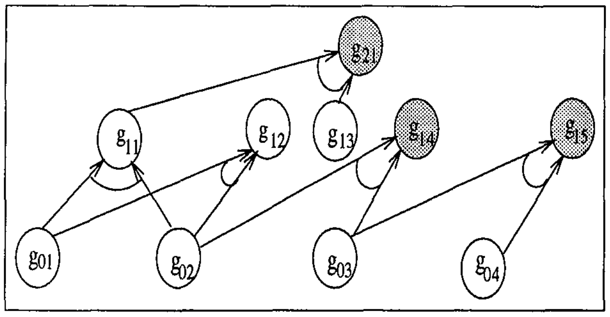
Goal specification is a pragmatic approach and has been used successfully in two applications. The first one is an expert system to solve an ill-structured problem called the "black box puzzle." Here, goal specification was used to extract the paths from the existing rule base (Grossner *et al.* 1996; Simon 1973). It proved to be a powerful tool by controlling the combinatorial explosion that arose due to the multitude of rule dependencies while enumerating rule base paths. The second applicaiton is an expert system that mimics a "reference librarian." Here, goal specification was used to structure the encoded knowledge to optimize the rule base and make rule base analysis easier. The earlier version of the rule base, due to lack of a structured design, performed poorly when the input size was large (Chander *et al.* 1997). A brief description of how the goal-based approach facilitates both system re-structuring and evaluation appears below.

Experience Report 1: The Blackbox System The Blackbox expert is a large system consisting of 435 rules developed using the CLIPS expert system shell (Giarratano & Riley 1993), and has been designed to solve the Blackbox puzzle (Grossner *et al.* 1996).[2] An important problem that is associated with the structural validation of such large rule-based systems in the extraction of paths is the combinatorial explosion that occurs while trying to enumerate all rule sequence combinations (Grossner *et al.* 1996). A brief description of how goals were useful in cutting down this combinatorial explosion appears below. For more details on the structural validation of the system using paths and goals see (Preece *et al.* 1993b; Grossner *et al.* 1996).

In the Blackbox expert, the rule base was developed without any prior knowledge of the goals, and goals were identified by the importance of an atom. The goals were first reverse engineered using hypotheses from the rule base by the rule base designer, Since the paths in our case are rule sequences inferring goals, whenever a combinatorial explosion arose because an inferred atom A by a rule r is used in the antecedent of many rules, the number of such rules was used as a heuristic to determine whether the atom A should be included as part of an existing goal (goal incorporation), or qualifies as a new goal (incremental goal identification) to cut the rule dependencies arising due to that atom. This goal-based restructuring process helped cut the combinatorial explosion and facilitated the extraction of rule base paths and to perform a succesful structural validation of the system.

Experience Report 2: The Reference Librarian System The second application of goal specification was an expert system to perform library search. The expert is given a set of input search fields associated with a document such as `title, author, subject` ..., up to a total of thirteen fields. The system checks for the location of the document using a data base system, and having obtained the location eventually retrieves the document. For more details on the expert system, see (Chander *et al.* 1997).

[2]The Blackbox puzzle consists of an opaque square grid (box) with a number of balls hidden in the grid squares. The puzzle solver can fire beams into the box. These beams interact with the balls, allowing the puzzle solver to analyze the contents of the box based on the entry and exit points of the beams. The objective of the Blackbox puzzle-solver is to determine the contents of as many of the grid squares as possible, while minimizing the number of beams fired (Grossner *et al.* 1996).



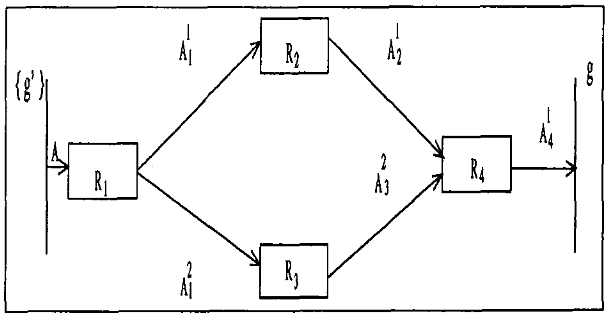


Figure 2: A sample goal graph and a rule base path.

```
;; A typical rule simplified from its original form
(defrule test-rule  (phase ?x)
  (phase ?y&~?x)
  (phase ?z&~?y&~?x)
=>
  (find-match t ?x ?y ?z) ; some action
)
```

Figure 3: A simplified version of a rule in the rule base of the library expert system. The rule takes three input fields and tries to find a set of documents matching the input.

```
(defrule test-rule-restructured
  (FIELD-INPUT 3) ; cut subsumption
  (FIELD-VALUES ?x ?y ?z) ; cut enumeration
  (phase ?x)
  (phase ?y)
  (phase ?z)
=>
  (find-match t ?x ?y ?z) ; some action
)
```

Figure 4: Rule in Figure 3 restructured using two goals (FIELD-INPUT and FIELD-VALUES).

In this application, however, the knowledge obtained from the human reference librarians (the domain experts) was translated literally to handle the various input fields while searching for a document. One such rule is shown in Figure 3. However, the direct translation as shown to check whether each input field is different from the other using the "&" and "~" operators of the CLIPS shell in the rules caused an enormous computational overhead as the inference engine tries to enumerate all possible instantiation for the variables in the rule antecedent (in fact, this pattern matching computation is of factorial complexity (Chander *et al.* 1997)). For example, the rule in Figure 3 would be activated by the CLIPS inference engine six times (= 3!) for the sample input `(phase author)`, `(phase title)`, `(phase subject)`. The primary reason for this inefficiency is the pattern matching computation that is forced to enumerate the variable instantiation in the rule antecedent. This, augmented by severe subsumption between rule groups in the system (rules that handle *i* input fields subsume rules that handle *i+1* input fields entered, but, the handling of *i* input fields versus *i+1* input fields are supposed to be exclusive), caused additional performance degradation. This subsumption arises because there are no goals identified as discriminators and encoded into the rule groups. For goal-based re-structuring, we identified predicates and goals to reflect the extent of input handling. In addition, we also need to make use of the fact that rules that handle i-fields can be treated exclusively from rules handling a different number of input fields. Subsumption was controlled and rule base performance improved considerably upon augmenting the antecedent of the rules of the form shown in Figure 3 with the such goals used as discriminators (see Figure 4 that shows a re-structured rule using goals as discriminators) (Chander *et al.* 1997).

Design Issues for Integrated Evaluation

Broadly speaking, integrated evaluation is the applicability of appropriate evaluation procedures to assess the evolution of every stage in a life-cycle that otherwise does not include an explicit (that is, post-development) evaluation phase. In other words, the system development and evaluation processes go hand in hand and are not treated separately in a system's life-cycle.

A design that allows integrated verification and validation should emphasize rule sequences rather than individual rules. The intent is that the design captures the rule interactions in the form of rule sequences rather than individual (or pairwise) rules so that the role of every rule in problem solving can be explicitly mapped to the acquired knowledge. An ideal model would capture every possible rule sequence without losing computational tractabiliy, but owing to the exponential complexity associated with rule sequence enumeration, one can only hope to be closer towards the ideal (Preece *et al.* 1993a; Hamilton, Kelley, & Culbert 1991).

Goal specification can be used to extract paths and can also be used to control the computation for path extraction. This allows one to speculate on its role for integrated evaluation. Below, we outline issues that confront integrating evaluation in a system's life-cycle and explore how goal specification based on rule base designs can *facilitate* integrating evaluation into a system's life-cyle. We do not claim that goal specification and path-based evaluation methods (Chander 1996) are panacea to the problem of integrating evaluation in a system life-cycle efficiently, but merely point out many of its features that support this objective.

Structure extraction for V&V As defined earlier, the structure of the system in our case is a static part of the system which is a set of rule sequences that move from goal-to-goal resulting in a progression of problem solving. The structure of a rule base forms the basis for structural validation (Kiper 1992; Chang, Combs, & Stachowitz 1990; Rushby & Crow 1990; Grossner *et al.* 1996). The extraction of paths, however, is a non trivial issue due to the combinatorial explosion that arises while trying to enumerate the paths in a system, and goal specification can be used as "meta knowledge" of the domain to cut down the computation required to extract the paths involved in goal-to-goal progressions in the system. They also facilitate path extraction, even if goals are to be reverse engineered from an existing rule base (see Experience Report 1 above). A software tool, called Path Hunter, has also been developed to extract the paths in a rule base given the goal specification (Grossner *et al.* 1996). The choice of a design scheme has an important bearing in extracting the paths from a rule base (Chander, Radhakrishnan, & Shinghal 1996).

Behavior understandability The behavior is a dynamic part that is actually observed as the various rules are fired, and is a complimentary aspect of the rule-based system structure. Behavior understandability, informally, can be stated as "how well the system concurs with the problem solving view/method of a domain expert?" It is difficult to understand the problem solving that occured by examining only the rules fired because a rule can fire under different situations, and it is difficult to determine the exact problem solving context under which a given rule fired for each of its firing observed in a run trace. The understandability of behavior thus implies the ability to map a run trace unambiguously to a set of paths thereby knowing the actual goal-to-goal progression that occured while solving a given problem. Our design scheme helps one to understand the role of a fired rule by mapping this rule to a path.

Performance evaluation A well-defined way for performance evaluation is provided by the scheme using the acquired knowledge as the basis. One of the problems that is currently faced by researchers in performance evaluation is that of defining a "good" criteria to assess performance (Guida & Mauri 1993). In our case, the goal specification approach incorporates the notion of a goal, captured during knowledge acquisition, as a unit of work done by the system in solving a problem: indeed by building the goal graph from the paths, a procedure has been developed to assess a system's optimality (a measure of the extent of redundant work done by the system) and its adequacy (a measure of the system's ability to solve problems in a domain) (Chander *et al.* 1997).

Incremental Evaluation An important aspect of integrated evaluation is its inherent ability to facilitate incremental testing to reduce evaluation costs (Meseguer 1992). Evaluation based upon goals and paths provide support to this objective in the sense that incremental path extraction is possible subject to some limitations (based on the type of modifications effected on a goal specification and its associated rule base). Incremental path extraction for unconstrained modifications to goal specification and rule base, however, is an open problem (Chander 1997).

Test case generation The ease with which test cases can be generated to test a given rule base is important to control evaluation costs. In our case, sequences of paths enumerated from permissible initial evidence to final goals can help identify initial evidence that can be input selectively to the system (Chander, Shinghal, & Radhakrishnan 1994). Automated test-case generating tools, similar in spirit to those described in (Ayel & Vignollet 1993), can easily integrate with our approach. Indeed, a simple modification to the goal graph extraction algorithm presented in (Chander, Shinghal, & Radhakrishnan 1994) is all that is required.

Ease of analysis of test case coverage This can be re-stated as follows: "given a test case and the run trace of the system as a set of rule firings, how can we measure the coverage of the system for this test?" The paths extracted from a goal specification based design can help measure given test case coverage of the rule base by means of a set of criteria that allows how many paths are exercised and the extent to which a given rule sequence is exercised for a test run; a software tool, called Path Tracer, has also been developed by us to analyze the run trace of a system to measure rule sequence coverage (Preece *et al.* 1993b).

Comprehensive verification of rule base anomalies Inference chains are widely used as a basis for comprehensive verification schemes (Ginsberg 1988; Ginsberg & Williamson 1993; Rousset 1988; Loiseau & Rousset 1993). Paths are generalized inference chains compared to the linear chains used for computing labels in the above schemes. A set of criteria has also been developed for comprehensive rule base verification by spotting certain path combinations called "rule aberrations" (Chander, Shinghal, & Radhakrishnan 1995) to detect rule base anomalies.

Quality Assurance Goal-based design scheme provide several metrics to assess the "goodness" of knowledge representation, and implementation (Chander, Radhakrishnan, & Shinghal 1997a). It is also relatively easy to track these metrics as a rule base evolves. In addition, paths can provide several implementation-specific metrics to assess the various qualities of a rule base such as its complexity, verifiability, etc (Preece *et al.* 1993a).

Reverse engineering support for existing rule bases Several existing rule bases have been constructed with incomplete, ad hoc methods of knowledge acquisition and representation. Evaluation methods for such rule bases should provide support for reverse engineering of specifications in a reasonable manner. Goal specification allows for reverse engineering simply by identifying and specifying the goals appropriate to the existing rule base to extract the paths (see also Experience Report 1). Goal specification based rule base construction provides several different design schemes and a systematic set of guidlines have also been developed to facilitate rule base developers in choosing a design scheme from several alternatives for easier reverse engineering and evaluation (Chander, Radhakrishnan, & Shinghal 1997b).

Summary & Conclusion

Our aim to integrate the evaluation processes in a system life-cycle is based on providing a link that connects the conceptual, design, and implementation levels of a system. Knowledge is acquired using goal specification to capture the desired problem solving states. Based on the specified goals, the structure of the system at the implementation level is defined by a set of rule sequences inferring goals. Goals also play an important role in providing design alternatives based on the choice of goal composition versus hypotheses in a rule base (Chander, Radhakrishnan, & Shinghal 1997a).

We believe that a goal specification based approach for system construction holds much promise for continous improvement of a system's quality and reliability by helping to integrate system evaluation as part of its development cycle. However, integrating the verification and validation processes is non-trivial and requires consideration of several research issues. We outlined those issues and argued how a goal specification based design can help a developer in tackling these issues.

References

Andert Jr, E. P. 1993. Integrated Design and V&V of Knowledge-Based Systems. In *Notes of the Workshop on Validation and Verification of Knowledge-Based Systems (Eleventh National Conference on Artificial Intelligence)*, 127–128.

Ayel, M., and Vignollet, L. 1993. SYCOJET and SACCO: Two Tools for Verifying Expert Systems. *International Journal of Expert Systems* 6(3):273–298.

Chander, P. G.; Shinghal, R.; Desai, B.; and Radhakrishnan, T. 1997. An Expert System for Cataloging and Searching Digital Libraries. *Expert Systems with Applications* 12(4):405–416.

Chander, P. G.; Radhakrishnan, T.; and Shinghal, R. 1996. Quality Issues in Designing and Evaluating Rule-based Systems. In *Notes of the Workshop on Verification & Validation of Knowledge-Based Systems (Thirteenth National Conference on Artificial Intelligence)*, 33–42.

Chander, P. G.; Radhakrishnan, T.; and Shinghal, R. 1997a. Design Schemes for Rule-based Systems. *International Journal of Expert Systems: Research and Applications* 10(1):1–36.

Chander, P. G.; Radhakrishnan, T.; and Shinghal, R. 1997b. A Study of Qualitative and Empiricial Selection Factors for Rule-based System Design. . In *Proceedings of the International Conference on Artificial Intelligence and Soft Computing (IASTED'97)*, 419–422.

Chander, P. G.; Shinghal, R.; and Radhakrishnan, T. 1994. Static Determination of Dynamic Functional Attributes in Rule-based Systems. In *Proceedings of the 1994 International Conference on Systems Research, Informatics and Cybernetics, AI Symposium (ICSRIC 94)*, 79–84.

Chander, P. G.; Shinghal, R.; and Radhakrishnan, T. 1995. Using Goals to Design and Verify Rule Bases. *Decision Support Systems.* In press.

Chander, P. G.; Shinghal, R.; and Radhakrishnan, T. 1997. Performance Assesment and Incremental Evaluation of Rule-based Systems. In *Notes of the Workshop on Verification & Validation of Knowledge-Based Systems (Fourteenth National Conference on Artificial Intelligence AAAI-97)*, 40–46. In press.

Chander, P. G. 1996. *On the Design and Evaluation of Rule-based Systems.* Ph.D. Dissertation, Department of Computer Science, Concordia University, Montreal.

Chander, P. G. 1997. Design Issues in Communication and Cooperation for Office Information Management. Computer Science Technical Report (Jan'97), Concordia University, Montreal, Canada.

Chang, C. L.; Combs, J. B.; and Stachowitz, R. A. 1990. A Report on the Expert Systems Validation Associate (EVA). *Expert Systems with Applications* 1(3):217–230.

Giarratano, J., and Riley, G. 1993. *Expert Systems: Principles & Programming* (2nd edition). Boston, MA: PWS Publishing Company.

Ginsberg, A., and Williamson, K. 1993. Checking for Quasi First-Order-Logic Knowledge Bases. *Expert Systems with Applications* 6(3):321–340.

Ginsberg, A. 1988. Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI 88)*, volume 2, 585–589.

Grossner, C.; Gokulchander, P.; Preece, A.; and Radhakrishnan, T. 1996. Revealing the Structure of Rule-Based Systems. *International Journal of Expert Systems: Research and Applications* 9(2):255–278.

Guida, G., and Mauri, G. 1993. Evaluating Performance and Quality of Knowledge-Based Systems: Foundation and Methodology. *IEEE transactions in Knowledge and Data engineering* 5(2):204–224.

Hamilton, D.; Kelley, K.; and Culbert, C. 1991. State-of-the-Practice in Knowledge-based System Verification and Validation. *Expert Systems with Applications* 3(3):403–410.

Kiper, J. D. 1992. Structural Testing of Rule-Based Expert Systems. *ACM Transactions on Software Engineering and Methodology* 1(2):168–187.

Krause, P.; Fox, J.; Neil, M. O.; and Glowinski, A. 1993. Can We Formally Specify a Medical Decision Support System? *IEEE Expert* 8(3):56–61.

Lee, S., and O'Keefe, R. M. 1994. Developing a Strategy for Expert System Verification and Validation. *IEEE Transactions on Systems, Man, and Cybernetics* 24(4):643–655.

Loiseau, S., and Rousset, M.-C. 1993. Formal Verification of Knowledge Bases Focused on Consistency: Two Experiments Based on ATMS Techniques. *International Journal of Expert Systems* 6(3):273–298.

Meseguer, P. 1992. Incremental Verification of Rule-based Expert Systems. In Neumann, B., ed., *10th European Conference on Artificial Intelligence*, 829–834.

O'Keefe, R. M., and Lee, S. 1990. An Integrative Model of Expert System Verification and Validation. *Expert Systems With Applications* 1(3):231–236.

O'Keefe, R. M., and O'Leary, D. E. 1993. Expert System Verification and Validation: A Survey and Tutorial. *Artificial Intelligence Review* 7(1):3–42.

Plant, R. T. 1992. Expert System Development and Testing: A Knowledge Engineer's Perspective. *Journal of Systems Software* 19(2):141–146.

Preece, A.; Gokulchander, P.; Grossner, C.; and Radhakrishnan, T. 1993a. Modeling Rule Base Structure for Expert System Quality Assurance. In *Notes of the Workshop on Validation of Knowledge-Based Systems (Thirteenth International Joint Conference on Artificial Intelligence)*, 37–50.

Preece, A.; Grossner, C.; Gokulchander, P.; and Radhakrishnan, T. 1993b. Structural Validation of Expert Systems: Experience Using a Formal Model. In *Notes of the Workshop on Validation and Verification of Knowledge-Based Systems (Eleventh National Conference on Artificial Intelligence)*, 19–26.

Rousset, M.-C. 1988. On the Consistency of Knowledge Bases: The COVADIS System. *Computational Intelligence* 4(2):166–170. Also in *ECAI 88, Proc. European Conference on AI* (Munich, August 1–5, 1988), pages 79–84.

Rushby, J., and Crow, J. 1990. Evaluation of an Expert System for Fault Detection, Isolation, and Recovery in the Manned Maneuvering Unit. NASA Contractor Report CR-187466, SRI International, Menlo Park CA.

Simon, H. A. 1973. The Structure of Ill-Structured Problems. *Artificial Intelligence* 4:181–201.

Yost, G. R. 1993. Acquiring Knowledge in SOAR. *IEEE Expert* 8(3):26–34.

Zlatareva, N., and Preece, A. D. 1994. State of the Art in Automated Validation of Knowledge-Based Systems. *Expert Systems with Applications* 7(2):151–167.