Integrated Design And V&V Of Knowledge-Based Systems

Ed P. Andert Jr.
Conceptual Software Systems, Inc.
P.O. Box 727, Yorba Linda, CA 92686, U.S.A.
andert@orion.oac.uci.edu

Introduction

The state-of-the-practice in Knowledge-Based System (KBS) verification and validation (V&V) has three major problems: developers are unaware of the need for and basic issues of V&V, tools are generally unavailable to aid developers, and research has concentrated primarily on static anomaly detection which alone is inadequate for validation of real-world KBSs. This abstract proposes an approach to improving the situation that concentrates on two key areas. The first is exploring a common knowledge representation and KBS execution environment to be augmented with integrated automated V&V tools. The goal is to develop the integrated environment as an open system for wide dissemination to developers. The second improvement is to develop innovative automated dynamic analysis techniques to augment current static anomaly detection capability.

The State-of-the-Practice in KBS V&V

The state-of-the-practice in KBS V&V can be characterized as not very good. This is despite a significant amount of research that has been performed in the area. KBS developers are unaware of the need for and the basic issues of V&V. In addition, even basic tools are virtually unavailable to aid developers. Finally, KBS V&V research has concentrated primarily on applying static anomaly detection on the KB which is not adequate for the validation of most real-world KBSs.

A solution is being pursued to the problem of KBS developers being unaware of the needs for and basic issues of V&V. A workshop has been developed by IBM for NASA/JSC to motivate the need for applying more systematic V&V approaches to KBS development and to provide hands-on experience in using V&V techniques. This is augmented by researcher efforts to document guidelines (funded by EPRI/NRC) and teach KBS V&V tutorials (primarily at AI conferences).

A major deficiency that has been noted by the IBM workshop effort and various V&V researchers, is that tools to aid developers in KBS V&V are virtually unavailable. These tools are needed to streamline the V&V process, reduce the cost of performing V&V, and aid in integrating V&V into the KBS development process. One commercial KBS development environment, EXSYS 3.0 from EXSYS, Inc. is known to have an automated V&V tool capability. Unfortunately, this tool performs only a single technique of basic static anomaly detection as a utility with little emphasis on basic issues and systematic V&V. Probably one of the most common mistakes in V&V is relying on a single post-development technique.

In addition, several tools have been developed by researchers. The most note-worthy of these are EVA and COVER. EVA built on results from seminal anomaly detection research (RCP and CHECK) to yield a comprehensive static anomaly detection tool. Unfortunately it was developed for the then-popular KEE environment on Symbolics platforms which has been abandoned by KBS developers. Thus, EVA is inaccessible and unused by KBS developers. COVER has similar functionality to EVA, but can be used on a variety of KBS development languages. The process for using the tool is to convert a KBS into COVER's rule language and then run the anomaly detection tool. Besides requiring non-automated KB conversion, COVER is only available from the academic developer which makes it essentially unavailable to KBS developers. R&D tools have made essential contributions to improving the state-of-the practice in V&V. The have shown the possibility of user-accessible tools, theoretical foundations for techniques, and the practicality of such tools despite theoretical complexity limitations.

KBS V&V research has concentrated on static KB anomaly detection techniques which are inadequate for assuring the reliability of nearly all real-world KBS applications. This is motivated by the ease of defining V&V techniques for a KB with a strong theoretical foundation assuming the validity of requirements and procedural (control) aspects of the KB/ inference engine. Unfortunately, the requirements and their transfer to the KB design are difficult to validate for most ill structured problems tackled with KBSs. Most commercial KBS development tools contain proprietary inference engine control specifics, making their validation a daunting task for developers. Therefore, it is difficult to assume the validity of requirements and inference engines. Finally, knowledge-base representation methodologies available in commercial tools have expanded to non-rule-based representations, such as case-based and model-based reasoning for which anomaly detection techniques for rule and frame representations do not readily apply. Thus, the well-

researched anomaly detection techniques alone are not adequate to validate KBSs.

Improving the State-of-the-Practice by Advancing the State-of-the-Art

Clearly the state-of-the-practice in KBS V&V needs improvement. Developers need help in becoming more aware of V&V issues. Tools that allow and encourage systematic V&V need to be made available to developers. Alternative techniques to static anomaly detection techniques need to be developed and explored for KBSs.

This approach to improving the situation is to advance the state-of-the-art in two key ways. The first is to explore a common knowledge representation and KBS execution environment to be augmented with integrated automated V&V tools. The goal is to develop the integrated environment as an open system for eventual transition to a commercial product. The second improvement is to develop dynamic analysis techniques to augment current static anomaly detection V&V capability.

The development of a common knowledge representation and execution environment (CKE) has several advantages. Knowledge-bases from various commercial/ popular knowledge languages can be automatically translated into the CKE and validated. This is more useful than trying to target a single knowledge language that suddenly becomes unpopular/unused due to unrelated issues. It is more advisable to select a couple of popular knowledge languages that are more closely associated with the CKE. This allows automated V&V integrated with development while avoiding lack of acceptance of the CKE because of the sudden downfall of a single knowledge language. An open system approach to inferencing and reasoning facilitates validation of non-proprietary control logic. In addition, the CKE should support open system style definition of various control and inference engine logics. This will allow both experimentation with alternative and more rapid validation of KBSs developed with less common knowledge development tools. The validation suite associated with the CKE can then readily address important issues such as real-time performance. All of this increases the usage of the V&V tool which improves developer awareness of issues and encourages systematic V&V.

A general improvement in the state-of-the-practice will result from even relatively small-scale acceptance of such a V&V tool (such as 10% of developers interested in V&V). The tool should be distributed with integrated tutorial information on how V&V should be systematically applied to KBS development.

The advantages of developing dynamic analysis V&V techniques include more comprehensive coverage, the ability to validate KBS that utilize proprietary inference engines and the ability to validate non-rule-based KBs. Dynamic analysis has long been utilized in general software V&V to yield more extensive software coverage to approximate exhaustive testing (since exhaustive testing is usually computationally intractable). Although dynamic testing has been characterized as inefficient in detecting and locating errors, it should be at least as effective as current static techniques. Current static analysis techniques can only detect anomalies that are indicators of errors and assume a validated inference engine/ control structure. In addition, dynamic analysis can aid in the validation of KBSs that utilize proprietary inference engines. Test sets can be generated from the KB and/or requirements and then applied to the KBS in its execution environment, including the proprietary inference engine. This will provide some level of confidence in the reliability of a KBS implementation. Finally, the current state-of-the art in static KB anomaly detection techniques and their theoretical foundation has primarily considered rule and frame-based systems. Dynamic analysis can be utilized to address model and case-based knowledge representation providing some degree of automated tool coverage. Theoretical underpinnings of various dynamic analysis testing techniques applicable to general software are just as applicable to KBSs.

There currently exists a unique opportunity to improve and impact the practice of KBS software development. KBS V&V is a young area of study with a small set of researchers and body of published research. The majority of KBS developers utilize a countable number of KBS development tools. A growing number of developers and potential developers are searching for answers on how to perform V&V on their software. Many KBS developers can be reached via publication and advertisement through a few magazines and advertisements. The opportunity is to develop and release a seminal KBS development environment that both implements and encourages comprehensive, integrated V&V. This can yield an improvement over the state-of-affairs in general software V&V where the process is essentially unautomated due to a large, complex body of research, techniques, and software developers. It is viewed as too costly to utilize automated methods because each project would require development of a unique tool that meets its needs and utilizes a selection of techniques from the myriad of technical literature.

Incorporating Uncertainty in a DAG-Based Approach to Static and Dynamic Verification of Rule-Based Systems: Extended Abstract

Valerie Barr
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
E-mail: vbarr@cs.rutgers.edu

In [Barr, 1992] we presented preliminary work on an approach to static and dynamic analysis of rule-based expert systems that uses a common DAG representation framework for both. The DAG is made up of predicate nodes and AND nodes. The predicate nodes represent individual components of antecedents and consequents, while the AND nodes represent the fact that multiple components of an antecedent must be true in order for the conclusion of a rule to be made.

The static analysis uses a look-up table (with entries for each component of an antecedent and for each complete antecedent) and depth-first traversal of the DAG representation in order to identify problems in the rule-base such as circularity, redundancy, conflict and ambiguity, dangling conditions, useless conclusions, and isolated rules. An approach for dynamic analysis was outlined, based on data-flow analysis of procedural programs, which executes over the same DAG representation used for static analysis. The dynamic analysis can determine if test data provided satisfies a chosen test criterion selected from a hierarchy of testing criteria. Of the five testing criteria proposed, the simplest is traversal of one sub-DAG to each goal of the system. This is equivalent to providing, for each goal, one test case that concludes that goal, clearly a very minimal criterion. The most difficult criterion is traversal of all sub-DAGs. (See [Barr, 1992] for the complete hierarchy).

In the initial work we considered only rule-bases that encoded reasoning processes without any explicit representation of uncertainty factors.

Ideally we would like the verification technique to be applicable to more general classes of rule-based systems, or base the method on a format to which systems can be easily converted regardless of their initial format. This task is made more difficult when we consider rule-based systems that include uncertainty information. The interpretation of uncertainty can vary from system to system, being viewed as probabilities or confidence factors. The particular way in which uncertainty information is viewed in turn opens up a number of possibilities for how to combine and manipulate that information during execution of the expert system. For example, if we consider the uncertainty information to be probabilities then we can take either a Bayesian or Dempster-Shafer approach during execution of the system. One aspect of our continuing work using the DAG representation is the selection of one view of uncertainty which will then be incorporated into the dynamic testing scheme.

The inclusion of uncertainty raises a number of issues for testing. For instance, if the system reaches a (final or intermediate) conclusion but only with very low certainty, should that be sufficient to satisfy a criterion which states that we must have test cases which cause every conclusion to be reached? Or must the system make a conclusion with high confidence in order for the tester to be satisfied that the system has worked correctly in that case. One approach to addressing these questions involves a fairly simple method for handling uncertainty in the DAG representation.

Commonly, the uncertainty associated with a consequent in a rule is expressed as a value within the interval $[0, 1]$. However, working with uncertainty over such a continuum would make it extremely difficult to incorporate into the DAG representation. Therefore we propose dividing the uncertainty range into three subranges:

- a "low" range, which is values in the interval $[0, .4)$
- a "medium" range, which is values in the interval $[.4, .7)$
- a "high" range, which is values in the interval $[.7, 1]$

We then represent each (intermediate or final) conclusion of the system three times, once for each subrange of uncertainty values. The testing criteria must be modified as well, to address sub-DAGs which include uncertainty.

There are related issues that arise once we start to include uncertainty in the DAG. Certainly, the num-

ber of test cases necessary to satisfy any of the test criteria will be greater than in a DAG without certainty factors. This raises the issue for the tester of how to choose samples to test the three different levels of certainty factors. Furthermore, there is the possibility that one input set might cause the same conclusion to be made multiple times, but with different certainty factors in each case. We must determine what this means relative to the overall correctness and integrity of the knowledge base.

In addition to the work involving uncertainty, there are additional aspects of this continuing work on the DAG approach to testing expert systems. One aspect is the inconsistency problem of identifying subsumption in the rule-base. It seems that potential instances of subsumption can be identified if the look-up table entries for consequents also point back to the antecedents that concluded them. This will make it possible to compare a pair of antecedents which both result in the same consequent, to see if one subsumes the other. However, this still leaves the problem that subsumption is a computationally difficult problem. In an actual expert system, where the number of components of each antecedent can be relatively low, the problem would be less costly to solve. Clearly some work remains to fit the solution into the model outlined in [Barr, 1992] and provide a bound on the complexity, given an assumption about antecedent size. This also raises a further question of overall complexity tradeoffs, since limiting the number of components in each antecedent will cause the overall depth of the DAG to increase (concomitant with an increase in the length of reasoning chains in the underlying knowledge base).

Finally, there is the issue of possible restrictions on the DAG. Are there reasonable restrictions that we can make on the DAG that will not reduce the power of the analysis but will reduce the complexity of the analysis, particularly regarding the number of sub-DAGs? For example, if the DAG is singly connected [Pearl, 1988], where no more than one path exists between any two nodes, how does this affect the power and complexity of the analysis over the DAG, and what sorts of changes does such a restriction imply for the underlying knowledge base?

These represent the key issues that must be resolved in order for the DAG representation presented in [Barr, 1992] to be useful for the static and dynamic evaluation of rule-based expert systems that include uncertainty.

References

Barr, V.B. 1992. A dag-based approach to static and dynamic verification of rule-based systems. In *Workshop Notes of the AAAI-92 Workshop on Verification and Validation of Expert Systems*, San Jose, CA.

Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA.