From: AAAI Technical Report WS-93-05. Compilation copyright © 1993, AAAI (www.aaai.org). All rights reserved.

Validation & Verification of Knowledge-Based Systems

Musa J. Jafar
Department of Management & Systems
Washington State University
Pullman, WA 99164-4726
musa@wsuaix.csc.wsu.edu

Historically, knowledge-base validation and verification (V&V) has been carried out manually. It is the process where the knowledge engineers and their experts review the knowledge-base and look for syntactic and semantic errors, build decision trees, check for the logical connectivity of the rules, run as many test cases as possible, and watch for mismatches between the system's and experts' output. This is a time-consuming, error-prone process; it does not guarantee finding all errors. Manually built decision trees are tedious and need to be redrawn after each change in the knowledge-base. Finally, running test cases is a hit-or-miss activity, it only tells if the system has errors, it does not tell where an error occurred or what caused it.

Interactive V&V on the other hand, is an automated process that insures the soundness and completeness of a knowledge-base. It guarantees that the addition, deletion or modification of any set of rules does not leave the system in a state of chaos. It allows the testing of predicates for their values, and rules for their validity and proper connectedness. It also ensures that every element in the knowledge-base is logically accessible and contributes to the overall soundness and completeness of the system.

To solve the V&V problem we treat a knowledge-base as a proof system where enough evidence accumulates to successfully fire every rule and prove the truthness of it's propositions. The truthness of every premise of every rule is thought of as a theorem to be proved through inferencing procedures, from facts in the underlying database or through an external interface. The rules and their premises are well formed formulas that represent the strategic governing rules of operations and skills in a domain which is critical to the success of the organization. Rules and skills accumulate over time. They are hard to replace, "Corporate short-term memory problem, where experts are not around for a long time". The meta-facts and meta-knowledge represent the integrity constraints of the system. The underlying database is a set of well defined constructs and procedures that serve as a repository for the corporate data and is used at the staffing and management level of the organization. A knowledge-based system provides all features provided by a database system, plus support for complex operations that involve decision making (rules and policies). Within this framework, the integrity problem of a database system is a sub-problem of the integrity problem of a knowledge-based system; a "View" in a relational database is equivalent to a set of production rules; a table is equivalent to an object where the table name is equivalent to the object name; the table attributes are equivalent to the object attributes and the table records are instances of the object. In a rule, the instance of an object is usually called a premise, a predicate or a proposition. Within this framework the following definitions are valid.

Interpretation: An Interpretation I is a mapping from the states of the knowledge-base to the underlying conceptual model that represents the target system. The domain of I is the set of all possible snapshots of the knowledge-base and the range of I is the set of all possible realizations of these snapshots. An interpretation maps the elements of the language used to build the knowledge-base to their conceptualizations. It assigns variables used by production rules to the corresponding elements in the universe of the conceptualization.

Valid Rule: A production rule is valid if it is true under the variable assignment rules of every interpretation of the knowledge-base. In other words, the conceptualization of the rule is indeed true in the real world. Validity of a rule does not guarantee the truthness of its conclusions. It is also possible for a rule to be valid and to have false condition premises.

Provable Rule: A rule is provable if there exits a non-empty interpretation whose variable assignment (restricted to the rule) enables the firing of the rule. A data-driven rule is provable if there exists an interpretation under which the condition premises of the rule are evaluated to true. A goal-driven rule is provable if there exits an interpretation under which the conclusion premises are in the search path of a goal.

Consistency: A knowledge-base is consistent if there is no interpretation that yields both a statement and its negation. In logical terms, it is impossible to infer both P and not(P) from the same set of facts and inferencing procedures.

Completeness: A knowledge-base is complete if every valid rule is indeed provable. In other words, for every rule there exists an interpretation under which the variable assignment rules (restricted to the rule) yield a non-empty set. Completeness guarantees the accumulation of enough evidence to fire every rule and to prove the truthness of its conclusions.

Soundness: A knowledge-base is sound if every provable rule is indeed valid. In other words, every premise and every conclusion of a rule are properly connected. It guarantees that no dead-end rules and constructs exist in the knowledge-base.

Validation: Validation is the process by which the knowledge-engineering team insures that the knowledge-base is consistent, sound and complete. It guarantees that the addition, deletion or modification of any set of rules does not leave the system in a state of chaos. Validation allows for the testing of predicates for their values and rules for their validity and proper connectedness. It insures that every element in the knowledge-base is logically accessible and contributes to the overall system. It insures the correct assignment between variables and their conceptualizations.

Verification: Verification is building the system right. It insures that the system correctly implements the specifications of the previous phase. It insures that the system does not have any syntactic errors and every instance of every object is indeed useful.