International Journal Of Computer Science And Applications Vol. 1, No. 3, December 2008
ISSN 0974-1003

A New Evaluation Strategy for AI System Based on Modeling Method

Wei Zhang
Department of Software and Computing Theory
College of Computer
Wuhan, 430079, China
+86 027 63673786
magherozhw@gmail.com

ABSTRACT
A new Evaluation method is proposed which view an Artificial Intelligence system as a model and leads to certain criteria for testing methodologies. This includes a discussion of how certain mathematical techniques for testing Artificial Intelligence systems can be used as criteria for Artificial Intelligence system adequacy when no other models are available. We give an example of an error due to widespread rule interactions. Such errors are keys to understanding why the independent rule assumption does not work, and therefore why Artificial Intelligence systems must be modeled. We examine how testing can be applied both to individual system components as well as the system as a whole, different criteria by which a set of test cases can be assembled and the problems in determining whether the performance of a Artificial Intelligence systems on a set of test cases is acceptable.

Categories and Subject Descriptors
I.2.8 **[Artificial Intelligence]**: Deduction and Theorem Proving – *Answer/reason extraction, Deduction.*

General Terms
Measurement, Theory, Verification.

Keywords
Artificial Intelligence, Evaluation method.

1 INTRODUCTION
Modeling is vital to the design, verification, validation, and testing of Artificial Intelligence systems [1], so we present the basic argument that building an AI System is building a model. One of the problems we discuss is that those developers had an overly rigid concept of what modeling is [2,3]. We show that in fact rules are not independent, and that therefore we must approach Artificial Intelligence systems from a modeling perspective.

We will discuss of how certain mathematical techniques for testing Artificial Intelligence systems can be used as criteria for Artificial Intelligence systems adequacy when no other models are available. The Artificial Intelligence systems models will determine the testing strategy, for example, how we sample and select cases, prioritize functions, and, as we suggest, place probes. The models will also determine the amount and kind of structural and functional testing.

2 BUILDING CORRECT INTELLIGENCE SYSTEMS

2.1 Background
Every Testing analysts knows that reading and understanding code is much harder than reading and understanding high-level descriptions of a system. For example, before reading the "C" code, an analyst might first study some high-level design documents. The problem with conventional software is that there is no guarantee the high-level description actually corresponds to the low-level details of the system [4].

A distinct advantage of model-based Artificial Intelligence systems is that the high-level description is the system [5]. A common technique used in AI is to define a specialized, succinct, high-level modeling language for some domain. This high-level language is then used to model the domain. If another automatic tool is used to directly execute that notation, then we can guarantee that the high-level model a correspondence to the low-level execution details. These models are often declarative and testing analysts can exploit such declarative knowledge for their analysis. Declarative representations can best be understood by comparing them to procedural representations used in standard procedural languages such as "C".

2.2 Examples of Widespread Errors
At the quite beginning, builders of expert systems basically attempted to debug the knowledge base by running test cases and interacting with a human expert in such a fashion that the expert could state objections to the conclusions and to the use of rules. They divide errors that may occur into two types: inconsistency and incompleteness. Their solution to inconsistency is to construct a table of all possible combinations of the antecedents' values and their corresponding values for a given concluding parameter and then to check for redundancies and contradictions among the rules. Although this consistency checking is a valuable tool, it does not pick up all the errors in a rulebase, especially if the errors stretch across several tables.

ISSN 0974-1003

Figure 1 is a picture of a fragment from an imaginary rulebase in a decision aid to help general physicians diagnose and treat patients.

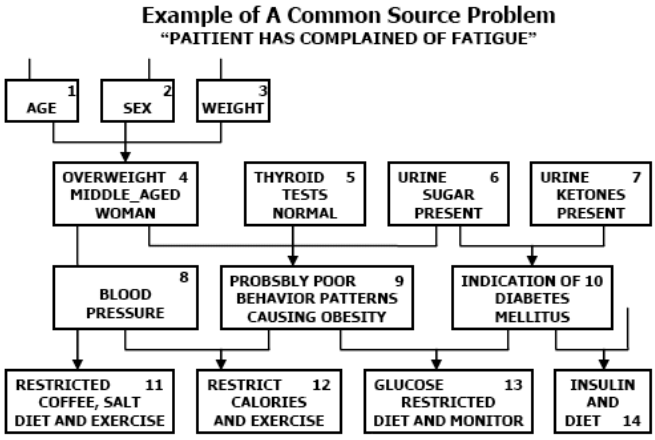

Figure 1. Diabetes flow example

The resulting flow graph is sometimes loosely described as a "logic tree"; in practice, this graph may not satisfy the mathematical definition of a "tree" (as the graph in Figure l does not).

As is common and appropriate in creating a rulebase, the expert will take the same initial assertions and "build" them into chains of assertions that support different possible scenarios or hypotheses, resulting in distinct conclusions. The result of this is that sometimes the same piece of information or an initial assertion will be fed into a number of different "lines of reasoning," some of which may reconverge. When this happens we call that assertion a "common source"; a common source may or may not lead to an error. Table 1 is the subset of rules for the fictitious Diabetes example shown in cure 1 that is concerned with patient fatigue. Because of a common-source error even though there are no "local" inconsistencies, assertion 13 can not be assigned True.

In order to assign assertion 13 True, both assertions 9 and 10 must be assigned True. However, assertion 6 cannot be True (as needed for assertion 10 to be True) and False (as needed for assertion 9 to be True) at the same time.

This example illustrates a relatively plausible (though simplified) story of how interactions among rules can lead to unexpected results. The cumulative effect across several assertions is that some rules can never be activated. If a knowledge-base is expressed in functions or objects that can change the rules during execution, then we cannot detect all common sources, except in special cases.

One response to this error may be to ban common sources by, for example, rewording assertions so that no one can use the same assertion as input to several other assertions. This type of "quick fix" mentality is pervasive among rulebase developers and is a serious mistake. The above common-source error is one way in which information can unwittingly drop out, be misused, or be underutilized by a rulebase. It also suggests that such misuse can involve wide groups of rules.

2.3 Clarifying Model(s) Underlying Diabetes Example

First and most important is the clarification and specification of the goals. In addition to clarifying and tightening the goals, another important quality of a modeling approach is that it explicitly links measurements to the variables they represent, together with the assumptions and limitations underlying the use of those measurements.

Table 4. Example of patient fatigue

Assertion:	**4**	**5**	**6**	**9**
Rule				
R41	F	F	F	F
R42	F	F	T	F
R43	F	T	T	F
R44	F	T	F	F
R45	T	F	F	F
R46	T	F	T	F
R47	T	T	T	F
R48	T	T	F	T

Assertion:	**6**	**7**	**10**
R61	F	F	F
R62	F	T	F
R63	T	F	F
R64	T	T	T

Assertion:	**9**	**10**	**13**
R91	F	F	F
R92	F	T	F
R93	T	F	F
R94	T	T	T

A lot of errors both at the semantic and syntactic levels occur because ① It is tricky to combine disparate types of information safely and usefully, and ② Knowledge base developers have often lost track of what the original measurement or assertion was supposed to represent, and have often reinterpreted it, cleverly but untrustworthily, in other parts of the knowledge base. Different levels of detail in a model require different fundamental terms and relationships. More generally, different models are invoked by different questions and situations.

3 MODEL-BASED TESTING DESIGN ON ARTIFICIAL INTELLIGENCE SYSTEMS

3.1 Building Testable Artificial Intelligence Systems

Two questions always arise when any system is tested: How many cases are needed? Which cases should be selected? The

models in an Artificial Intelligence system are fundamental to all these suggestions, from characterizing and prioritizing the Artificial Intelligence systems functions, to modularizing the Artificial Intelligence systems, or forming intervening variables to deciding what capabilities are worth what testing efforts.

In order to design testable Artificial Intelligence systems, we need to be able to relate and track the functional requirements (goals of the Artificial Intelligence systems) to the Artificial Intelligence system implementation. Another key factor in making more testable Artificial Intelligence systems is the development of better Artificial Intelligence systems languages.

3.2 Limit Artificial Intelligence System to Propositional Logic

Propositional calculus (Boolean or sentential logic) is finitely decidable (though the size of the finite decision process may be impractically large), but rules that require predicate calculus are not finitely decidable (and may therefore be undecidable). Nevertheless, some rulebases which are written in syntax that appears to involve predicate calculus, while in reality the rules can be completely translated into Boolean polynomials; thus, the appearance of needing predicate calculus may be illusory.

For example, a rule in one of our rulebases, used the phrase "all phenomena". However, in this case it meant merely a list of phenomena covered by the rulebase and did not essentially involve predicate logic. If your system is written in, or translated into, Boolean expressions, then an upper bound on the number of input conditions needed for exhaustive testing is 2 to the power of the number of (Boolean) input variables. If this is too many for practical testing (which includes not only the number of tests that need to be constructed and carried out, but also includes the time that we have to spend on checking and analyzing the results), then consider intervening variables.

3.3 Intervening Variables in Reducing Testing

Summary variables can greatly reduce the number of test conditions required for complete testing of a rule-based system. If (A) a system has 40 variables and no intervening or summary variables, then there are 2^{40} = 1 099 511 627 776 (about a trillion) conditions to test the system exhaustively. If (B) one devises four intervening variables, each of which responds to l0 of the original variables with a binary value, then there are only $4\times2^{10} + 2^4 = 4$ 128 conditions to test.

In most systems, exhaustive testing is infeasible, and one must instead select the tests based on a variety of more careful considerations, ranging from practical criteria, reflecting the time, cost, and difficulty in performing the tests, through mathematical and statistical criteria, regarding the sampling over the spaces of possible input and output variables, to functional criteria, corresponding to the importance of various system performance requirements.

4 VALIDATION ANALYSES FOR ARTIFICIAL INTELLIGENCE SYSTEMS

Validation is the process of ensuring that a software system satisfies the requirements of its users. Assuming that the requirements stated in the problem specification are correct, and then verification is part of validation concerned with establishing formalized properties of the system [6]. In addition to verification, validation also includes empirical evaluation techniques in which experiments are performed on the system, the results of which are analyzed carefully to determine whether the system is acceptable.

4.1 Component Testing for Artificial Intelligence Systems

Most often, knowledge bases are tested only as black boxes, that is, as an indivisible whole. However, there is potential for component or module testing for these systems in certain circumstances. One definite validation requirement for all Artificial Intelligence systems is to test that the inference engine satisfies the requirements specified in the system design model. As described earlier, depending on the exact approach followed by the developers, verification of the inference engine may be accomplished either by formal methods or empirical testing.

In addition to performing testing of the inference engine separately, it may be possible to test components of the knowledge base independently, if the problem can be decomposed in a suitable manner. It may be possible to test task knowledge by providing simulated domain level knowledge. As a trivial example, if the domain knowledge consists only of the rules $p\rightarrow q$ and $q\rightarrow r$, and we have a test case that says that, for input p, the system should produce output $\neg r$, and then it is easy to show that the knowledge base is inconsistent with this test case, and is therefore unacceptable.

4.2 Obtaining a Test Suite

A serious difficulty in obtaining such a set is that the diversity of situations in which an Artificial Intelligence system must perform is such that a representative set of cases will likely be very large, even when accounting for the existence of equivalence partitions within the input domains (that is, test cases that should be treated identically by the system).

The problem of creating a suite of test cases is exacerbated by the fact that each test case is likely to be complex, because a typical Artificial Intelligence system solves complex problems presented as complex test cases. This means that, not only will it be difficult and time-consuming to create or transcribe each case, but also it may be difficult to specify what constitutes an acceptable output for the case. Although we cannot avoid the necessity of testing, we can seek to minimize the effort involved by minimizing the number of test cases required.

Although random generation of test cases has proven very useful in testing conventional software, the complexity of Artificial Intelligence systems testing requires a more focused approach to test case creation. The most practical method for creating a set of test cases would seem to entail a combination of both structural and functional approaches.

4.3 Judging System Acceptability

One repercussion of this fact is that it may be difficult to choose an appropriate level of performance for the system to achieve in order to be accepted. A second repercussion is that it may be difficult to define a standard against which to judge the acceptability of the system.

5 CONCLUSION

Despite the exaggerated claims and grandiosity of some of their developers, Artificial Intelligence systems have proven to be useful in approaching and solving some problems. The challenge is to build more correct and testable ones. A lot of what we suggest is simply good modeling and good software engineering,

We consider models to be a fundamental part of improving Artificial Intelligence system design, verification, validation, and testing. A model is a framework for understanding and for doing something. Through models we view individual facts, view or work with the input, and evaluate whether overall the system is performing well—or whether input, output, and intermediate results were appropriate. We need to make the underlying models used in an Artificial Intelligence system explicit, well formed, and justified, and then we need to use these models to design the Artificial Intelligence systems System correctly from the outset and to form the basis of our testing decisions. Perhaps what we need the most is a change in our attitude towards testing.

6 REFERENCES

[1] Garcez, A. S., Lamb, L. C., Broda, K. et al. 2004. Applying Connectionist Modal Logics to Distributed Knowledge Representation Problems. International Journal on Artificial Intelligence Tools.

[2] Mili, A., Jiang, G., Cukic, B. 2004. Towards the Verification and Validation of Online Learning Systems: General Framework and Applications. Proc. 37th Hawaii International Conference on System Sciences (HICSS-37 2004), Big Island, Hawaii.

[3] Bracchi, P., Cukic, B., Cortellessa, V. 2004. Performability Modeling of Mobile Software Systems. Proc. 15th Int'l IEEE Symposium on Software Reliability Engineering (ISSRE'04), St. Malo, France.

[4] Barringer, H., Goldberg, A., Havelund, K. 2004. Rule-based runtime verification. Proc. Fifth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'04), Venice, Italy.

[5] Menzies, T., Pecheur, C. 2005. Verification and Validation and Artificial Intelligence. Advances in Computing, Elsevier.

[6] Menzies T., Port, D., Chen, Z. et al. 2005. Validation methods for calibrating software effort models. Proc. 27th International Conference on Software Engineering, St Louis, MO.