Proceedings of ANNIE 2002
Smart Engineering System Design
November 10–13, 2002, St. Louis, Missouri, USA

VERIFICATION AND VALIDATION OF COMPLEX SYSTEMS

D. E. Stevenson
442 R. C. Edwards Hall
Department of Computer Science
Clemson University
PO Box 341906
Clemson, SC 29634-1906
Email: steve@cs.clemson.edu

ABSTRACT

Verification and validation of complex system simulations are generally considered to be independent concepts. We consider the system model verified if there is a proof of the properties in question. We demonstrate that validation is a *contravariant* form of verification. We do this by considering the refutation tree produced by a resolution proof of the system properties. We indicate areas requiring further development.

Keyword I: Complex systems

Keyword II: Systems, modeling, theory, logic, theorem proving

Introduction

The goal of science is to explain the natural world. While some phenomena can be explained without reference to mathematics, most phenomena are modeled using mathematics (**?**). As our understanding increases, we can model more complex phenomena more completely . These complex models now most often require supercomputers to solve. These computational models — simulations —

add a new dimension of software and hardware correctness to the already complex question of scientific and mathematical correctness.

The Defense Modeling and Simulation Office (DMSO) is a central repository of information on modeling, simulation, verification, and validation terminology. From DMSO's site (www.dmso.mil) we find that *verification* is "The process of determining that a model or simulation implementation accurately represents the developer's conceptual description and specification. Verification also evaluates the extent to which the model or simulation has been developed using sound and established software engineering techniques." *Validation* is " The process of determining the degree to which a model or simulation is an accurate representation of the real-world from the perspective of the intended uses of the model or simulation." Validation, then, is the process of justifying (epistemology) a model to the physical processes it claims to represent (ontology). Note that the terms *verification* and *validation* are often treated as synonyms but in the specialized language of modeling and simulation they are not.

Although mentioned together, the two are generally regarded as two distinct concepts. Our goal is to show that they can be related through the logical theory that establishes the system. Normal proofs proceed from the axioms to the conclusion in a *covariant* manner. Validation proceeds from the conclusions to the axioms in a *contravariant* manner. We use the covariantly generated proof tree to explore validation. Our approach is to generate the tree by resolution theorem proving. We assume the reader is familiar with standard logical terminology and a basic understanding of resolution theorem proving.

Our approach is to generate a semantic structure based on resolution. Validation is a probabilistic statement on the acceptability of the tree. Before we can apply resolution theorem proving, we must first describe the logical system.

We proceed as follows. Section **??** is our logical foundations. Section **??** reviews resolution theorem proving and establishes the verification aspect. Validation is considered in **??**.

Logical Systems in Carnap-Hempel Logic

The logic of science is informal by mathematical logical standards, with no mention of axioms. There are instances of special logical systems, such as quantum logic, that have evolved from certain special cases; however, we do not pursue those systems here. We also do not pursue systems developed on Jaynes work (**?**).

We use a standard proposed by philosophers of science Rudolf Carnap and Carl Hempel. The system was used for normative purposes only and never used to actually describe a system. We now describe the logical system.

The Formal Languages for CH

The logical system developed primarily by Rudolf Carnap and Carl Hempel, which we refer to as **CH**, over the period 1936–1977. **CH** is complex due to non-logical languages for theory and observation. Each language has a vocabulary $\mathcal{V}$, a structure of objects, functions and relations that are the basis of interpretations. Formally,

$\mathfrak{L}_L$ The logical language is a first order predicate calculus with equality. This language may include modality, temporality, etc.

$\mathfrak{L}_T$ The theoretical language is the language of the science with *no* reference to the observational language. This language would be naïvely be the normal non-logical language. Generally, there will be no non-logical axioms.

$\mathfrak{L}_{o'}$ The pure observational language ($\mathfrak{L}_o$) that has only objects and events for its vocabulary. That is, $\mathfrak{L}_o$ is a pure term language. $\mathfrak{L}_{o'}$ is a logical language that incorporates $\mathfrak{L}_o$ as terms but introduces quantifiers, temporality, etc. The question of units and statistics, for example, are dealt with here.

$\mathfrak{L}_M$ The mixed language $\mathfrak{L}_M$ combines $\mathfrak{L}_T$ and $\mathfrak{L}_{o'}$ in non-vacuous ways. These terms are related to one another through *correspondence rules*, described in Section **??**

Interpretation for CH

Interpretation in **CH** takes place across the four languages. The vocabulary $\mathcal{V}_o$ is that of concrete observables: events, objects, and attributes, including issues of units. The relations and properties in $\mathcal{V}_o$ must be also observable. Every variable in $\mathfrak{L}_o$ must take on values of expressions in $\mathfrak{L}_o$ only. The key issue is that there can be no partial interpretation of $\mathfrak{L}_T$ based on observations *other than* those provided by $\mathcal{V}_o$ and $\mathcal{V}_T$.

The mixed language $\mathfrak{L}_M$ introduces a special type of interpretation known as correspondence rules. *Correspondence rules* are admissible procedures for applying the theory to the observations. They partially interpret $\mathcal{V}_T$ by specifying observable content. An example of a correspondence rule is the following example:

The Resolution Theorem Proving

As is well known, one approach to automatic theorem proving is known as resolution theorem proving (**?**; **?**; **?**).

Let $\mathfrak{L}$ be a first order predicate language . Let Δ be a set of statements in $\mathfrak{L}$ and let Γ be a statement in $\mathfrak{L}$ representing a conclusion. The resolution principle states that Δ proves Γ if and only there is no counterexample.

Resolution can be implemented in the following manner. Let $\Sigma = \Delta \cup \{\neg\Gamma\}$. Σ can be put into conjunctive normal form Σ_0' such that the elements of Σ_0' are of the form $l_1, l_2, \ldots, l_K$ where each l_k is either atomic formula or the negation of an atomic formula. Let Φ and Ψ be two such elements such that ϕ_i and psi_j are contradictory. If we conjoin the two, then we have a new clause with a term of the form $\alpha \wedge \neg\alpha$, an obvious contradiction:

$$\frac{\Phi \cup \phi_i \quad \Psi \cup \psi_j}{\Phi \cup \Psi}\, Resolution$$

We then induce a data base Σ_∞ that contains all possible combinations of statements and their derivatives. If we are ever able to generate the *empty clause*, then we will have derived a contradiction.

Why the process works becomes the departure for **CH**. Consider again Σ'_0. Let **HU** be the set of all constants appearing in Σ. **HU** is called the *Herbrand universe*. Let **HB** be the set of all possible terms that can be computed by the functions named in Σ'. **HB** is the *Herbrand base* and if there is a contradiction then it must come from here because of the interpretation rules. Artificial intelligence has made great use of this mechanism, greatly extending its range to cover applications of diverse logical content through procedural attachment. The difference between applications now is evident: these procedures could well be a super computing application.

The **CH** system considerably complicates this procedure because of there are different languages: $\mathfrak{L}$, $\mathfrak{L}_T$, $\mathfrak{L}_M$ and $\mathfrak{L}_{o'}$. The relationship of these languages is as follows: $\mathfrak{L}$ and $\mathfrak{L}_T$ can be mixed in any way, as can $\mathfrak{L}_o$ and $\mathfrak{L}_{o'}$ but the two groups can only be mixed through $\mathfrak{L}_M$.

The next complication comes from the difference in inference rules between the formal language $\mathfrak{L}$ on the one hand and $\mathfrak{L}_T$, $\mathfrak{L}_M$ and $\mathfrak{L}_{o'}$ on the other. The rules for each are discussed in (**?**).

Validation

To review, the tree generated by the resolution process represents a proof of the contradiction $\Delta \cup \neg\Gamma$. The tree itself is rooted by the empty clause and the leaves are clauses. Intermediate nodes are the result of the resolution rule

$$\frac{\Phi \cup \phi_i \quad \Psi \cup \psi_j}{\Phi \cup \Psi}\,\mathit{Resolution}$$

where ϕ_i and ψ_j are contradictory.

The tree produced by resolution

$$\begin{array}{c} \underline{\Phi \cup \phi} \quad \underline{\Psi \cup \psi} \\ \vdots \\ \Box. \end{array}$$

Inverse Considerations

From validation standpoint, the refutation tree must be processed backwards. For example, if we had a node that dealt with a simulation that has two arguments, say

$$B = A(x,y)$$

then the tree would have to determine x and y based on A^{-1}. This actually poses serious, but very interesting, difficulties. In the most general case, this is a problem of automatic program construction that would have to have an extensive knowledge of mathematical analysis and numerical methods.

Numerical simulations introduce another difficulty: numerical stability and numerical error. Numerical error is generally measure as *relative error*:

$$\text{relative error}_f \triangleq \frac{\partial f}{f}.$$

Numerical stability is a property of numerical methods. A method is stable if the method does not expand the relative error too much. While academic problems can be solved with most any numerical method that fits the problem, this is not true in serious science and engineering applications.

Probabalistic Considerations

Now consider an observation $\mathbb{E}$ that is confimatory of the theory but for which we are uncertain as to the exact observation. Let $p_{\mathbb{E}}$ be the probability of the event. What can be said about the probability of $\Phi \cup \phi_i$ and $\Psi \cup \psi_j$? This problem becomes more difficult when we realize that the reasoning rules in $\mathfrak{L}_T$ themselves may be uncertain. Therefore, any system must understand statistical reasoning as well as logical reasoning.

Unlike verification, which is all or nothing, there are degrees of validation but these degrees may not be distributions. An example often used to illustrate subjective probability (**?**) can be used to illustrate validation. Using probability to measure validation has a distinct advantage. There are several measures.

One measure can be defined in terms of the refutation process is as follows. Let Ξ be the set of formulas $\phi_i \wedge \psi_j$ from the refutation. These formulas represent the reason *why* the proof works. That is,

$$\Xi \equiv \xi_1 \vee \xi_2 \vee \ldots \xi_n$$

Ξ evaluates to `false` but only if every $\phi_i \wedge \psi_j$ is a contradiction. In experimental sciences we generally only have the *probability* that $\phi_i \wedge \psi_j$ is false. Therefore, the measure of confidence one has in the conclusion above is exactly the probability that $\models \Xi$ is 0.

The Ξ model is one half the story. In scientific theories any of the various predicates or reasoning steps may have an associated probability. The tree can be processed using linear optimization methods to assign probabilities to logical systems following the methods of probability logic (**?**). These methods are distinct from the “many worlds” interpretation of probability logic described in (**?**). The results of these evaluations would represent a probability that the proof is true subjectively.

Conclusion

We have investigated aspects of logic in science and engineering, relying on the the insights of Feynman (**?**). We have noted throughout that the scientific application of logic is quite different from the original logics used in automatic theorem proving.

Logical systems are used in verification. We have considered **CH**, a logical system for proving theorems about complex scientific and engineering models. **CH** requires several logical languages to relate to theoretical and observational systems. **CH** requires rethinking the resolution principle and its implementation.

Scientists use verification to derive their understanding of physical systems but rely on experiments to validate. We have shown that the tree developed during resolution theorem proving can be used to develop measures of validation. One measure is denoted Ξ that represents the terms discarded during resolution. The second measure represents the assignment of probability to the nodes of the resolution tree itself.

We are pursuing system development to implement the details of this verification and validation definition.