

Characterization of Software Quality Assurance Methods: *Five Methods for Verification of Learning Systems*

***Int'l Neural Networks Joint
Conference: V&V Workshop***

Montreal – August 5, 2005

**Frederick T. Sheldon (ORNL) and
Ali Mili (NJIT)**

Tentative Assessment

Current V&V Techniques for Adaptive Systems:

- Check partial functional properties.
- Check / enforce structural properties.
- Monitor/ detect/ filter novelty (of learning data).
- Monitor invariance of some functional properties.

What is Needed,...

Orthogonally,

- **Ability to quantify/ characterize all these measures in a uniform model.**
- **Ability to characterize subsumption, redundancy/ overlap and complementarity/ independence between methods.**
- **Ability to compose verification claims.**
- **Ability to decompose verification goals.**

Premises of Uniform Model

- **All claims are represented by refinement or refinement like formula**
 - to distinguish reliability from security.
- **All claims represented by probabilistic statement,**
 - possibly/generally conditional (to reflect implicit assumptions).
- **All claims have associated failure costs**
 - to distinguish reliability from safety.
- **All methods have associated verification costs**
 - to control/ budget verification cost.

Deploying an Eclectic Mix of Methods/ Tools/ Techniques?

- *Economic rationale.* Law of diminishing returns.
- *Technical rationale.* Each method is best adapted for some V&V aspect.
- *Pragmatic imperative.* Lack of planning, making use of available resources, tools, skills, etc.
- *Adaptive Software is a moving target.* Its unclear what method will provide the best return on investment.

Motivation

- Deploying an eclectic mix of methods is more crucial for *online learning systems* than for *traditional systems*.
- No single method is yet known to work (all traditional methods are provably not operational, all new methods admittedly ad-hoc/ partial).
- *Applications of online learning systems are typically critical.*
- *Applications of online learning systems are typically complex.*

A Refinement Based Approach

- *Conceptual Argument.* Refinement calculi good model for reliability, safety. Extension to security tempting/feasible;
 - we are working on a refinement like relation for security.
- *Pragmatic Argument.* Refinement calculi support composition of claims, decomposition of goals ---key strategic capability.
- *Methodological Argument.* Automated support enables us to manage dependability in a comprehensive, integrated fashion.

Modeling Verification Methods

- *Proving*: Proving that **P** is correct with respect to specification **V**:

$$P \supseteq V.$$

- *Testing*: Certification testing, Oracle Ω , test data **D'**, successful test on **D**.

$$P \supseteq T,$$

where $T =_{D'} \Omega$.

Modeling Verification Methods, II

- ***Fault Tolerance:*** Upon each recovery block, check condition C , invoke recovery routine R .

Because we do not know which outcome we have each time, all we can claim is:

$$P \supseteq F,$$

where $F = C \cap R$.

Cumulating Results

- **Proving:** $P \supseteq V$.
- **Testing:** $P \supseteq T$.
- **Fault Tolerance:** $P \supseteq F$.
- **Lattice Identity:**

$$P \supseteq (V \cup T \cup F).$$

Cumulating verification results into a comprehensive correctness claim.

Decomposing Verification Goals

Premises:

- **A Complex Specification can be decomposed into simpler sub-specifications in a refinement-compatible manner:**

$$S = S_1 \cup S_2 \cup \dots \cup S_N.$$

- **We can consider each S_i in turn, mapping it to the method that is most efficient for it.**

A Uniform Representation for Dependability Measures

Logical representation of verification results unrealistic:

- **Most verification results are best viewed as probabilistic, not logical, statements.**
- **Most verification results are conditional, contingent upon different conditions.**
- **Many verification results can be interpreted in more than one way.**

Probabilistic (vs Logical) Claims

- **No absolute certainty.**
- **Even highly dependable, totally formal, verification systems may fail.**
- **We want to quantify level of confidence.**

Verification Results are Conditional

- ***Proving:*** Conditional on verification rules being consistent with actual compiler/ being borne out by runtime environment.
- ***Testing:*** Conditional on testing environment being identical to/ more stringent than operating environment.
- ***Fault Tolerance:*** Conditional on system preserving recoverability.

Multiple Interpretations

- *Testing, first interpretation: $P \supseteq_{D \setminus \Omega} \Omega$, with probability 1.0.*
- *Testing, second interpretation: $P \supseteq \Omega$, with probability $p < 1.0$, conditional on D being representative.*

Which interpretation do we choose? We do not have to choose, in fact. We can keep both, and learn to add/ cumulate them.

Characterizing Verification Claims

- *Property.* Correctness preservation, recoverability preservation, security property, operational attribute.
- *Reference.* Functional Specification, Safety Requirement, Security Requirement, etc.
- *Assumption.* Implicit conditions in each method.
- *Certainty.* Quantified by probability.
- *Stake/ failure cost.* Cost of failure to satisfy a property with respect to a reference.
- *Penalty/ verification cost.* The cost of performing a verification task (for a given property/ reference/ assumption, etc).

Uniform Representation

- **Conditional probability:**

$$\Pi(S \supseteq R \mid A) = P.$$

Two additional cost functions:

- **Verification cost:**

$$\text{Prop} \times \text{Ref} \times \text{Meth} \times \text{Assum} \rightarrow \text{Cost}.$$

- **Failure cost:**

$$\text{Prop} \times \text{Ref} \rightarrow \text{Cost}.$$

Inference Rules

- **Collecting claims is insufficient.**
 - **Cumulating/Synthesizing claims (as we did with logical results) is impractical.**
- ⇒ **Build inference mechanisms that can infer conclusions from a set of claims.**

We will explore applications of this capability, subsequently.

Inference Rules

Derived from refinement calculus (tentative; to be double checked):

- $\Pi(\mathbf{S} \supseteq (\mathbf{R}_1 \cup \mathbf{R}_2) \mid \mathbf{A})$
 $\geq \Pi(\mathbf{S} \supseteq \mathbf{R}_1 \mid \mathbf{A}) \times \Pi(\mathbf{S} \supseteq \mathbf{R}_2 \mid \mathbf{A}).$
- $\Pi(\mathbf{S} \supseteq (\mathbf{R}_1 \cap \mathbf{R}_2) \mid \mathbf{A})$
 $\leq \Pi(\mathbf{S} \supseteq \mathbf{R}_1 \mid \mathbf{A}) + \Pi(\mathbf{S} \supseteq \mathbf{R}_2 \mid \mathbf{A}).$
- $\Pi(\mathbf{S} \supseteq (\mathbf{R}_1 \cap \mathbf{R}_2) \mid \mathbf{A})$
 $\geq \max(\Pi(\mathbf{S} \supseteq \mathbf{R}_1 \mid \mathbf{A}), \Pi(\mathbf{S} \supseteq \mathbf{R}_2 \mid \mathbf{A})).$

Inference Rules

Derived from Probability Calculus (tentative):

- $\mathbf{R_1 \supseteq R_2 \Rightarrow \Pi(S \supseteq R_1 | A) \leq \Pi(S \supseteq R_2 | A)}$.
- $\mathbf{(A_1 \Rightarrow A_2) \Rightarrow \Pi(S \supseteq R | A_1) \leq \Pi(S \supseteq R | A_2)}$.
- $\mathbf{\Pi(S \supseteq R | A \wedge B) =}$
 $\mathbf{\Pi(S \supseteq R | A) \times \Pi(S \supseteq R | B) / \Pi(S \supseteq R)}$.
- **Bayes' Theorem.**

Inference Rules

Derived from Cost functions (tentative/ illustrative):

- $R_1 \supseteq R_2 \Rightarrow VC(\supseteq, R_1, M, A) \geq VC(\supseteq, R_2, M, A)$.
- $(A_1 \Rightarrow A_2) \Rightarrow$
$$VC(\supseteq, R, M, A_2) \geq VC(\supseteq, R, M, A_1)$$
.
- $R_1 \supseteq R_2 \Rightarrow FC(\supseteq, R_1) \geq FC(\supseteq, R_2)$.

Sample Queries

- *Given a set of claims, with what probability can we claim that S refines R subject to A ?*
 - **Greatest lower bound for $\Pi(S \supseteq R | A)$.**
- *Given a set of claims, and R, A, p , can we claim that S refines R under assumption A with probability at least p ?*
 - **Theorem $\Pi(S \supseteq R | A) \geq p$.**
- *Given a set of claims, provide a weighted average of failure costs.*
 - **Mean of a random variable.**

Specializing Model to Adaptive Systems

Analyzed a number of existing methods to cast them in terms of the model

- *Monotonic Learning, Safe Learning* [Mili et al].
- *Novelty Detection* [Cukic, Liu, Schumann].
- *Periodic Rule Extraction* [Taylor et al].

Illustration: Composing Verification Claims

- *Very elementary prototype.*
- Illustrates what we mean by composing claims;
 - submits query that is not directly related to claims,
 - depends on more than one claim.
- Much to add: refinement dimension; cost functions; lattice composition; etc.
- Focused on security, for specific purpose.
- Encompasses dependability in all its dimensions.