Dagstuhl Seminar 13051, January 2013

Software Certification: Methods and Tools

Logic and Epistemology in Assurance Cases

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

My Dream

- cf. Leibniz' Dream: "let us calculate"
- To be able to evaluate the certification argument for a system by systematic and substantially automated methods
- So that the precious resource of human insight and wisdom can be focused on just the areas that need it
- A step toward intellectual justification of certification arguments
- Caveat: I'm concentrating on (functional and safety) requirements
    - All aviation software incidents arise in the transition from system to software requirements
    - Implementation assurance is fairly well managed, modulo derived requirements

Assurance Cases as a Framework

- No matter how certification is actually organized and undertaken
- We can describe, understand, and evaluate it within the framework of an assurance case
  - Claims
  - Argument
  - Evidence
- For example, in objectives-based guidelines such as DO-178C, the claims are largely established by regulation, guidelines specify the evidence to be produced, and the argument was presumably hashed out in the committee meetings that produced the guidelines
- But in the absence of a documented argument, it's not clear what some of the evidence is for: e.g., MC/DC testing
- Need to reconstruct the argument for purpose of evaluation

Assurance Cases and Verification

- The argument aims to justify the claims, based on the evidence
- This is a bit like logic
  - A proof justifies a conclusion, based on given assumptions and axioms
- Formal verification provides ways to automate the evaluation, and sometimes the construction, of a proof
- So what's the difference between an assurance case and a formal verification?
- An assurance case also considers why we should believe the assumptions and axioms and the interpretation of the formalized claims
- As an exercise, consider my formal verification in PVS of Anselm's Ontological Argument (for the existence of God)

Logic And The Real World

- Software is logic

- But it interacts with the world
  - Actual semantics of its implementation
  - Sensors, actuators, devices, the environment, people, other systems

- So we must consider what we know about all these

Epistemology

- This is the study of knowledge
- What we know, how we know it, etc.
    - Traditionally taken as justified true belief
    - But that's challenged by Gettier examples
    - And other objections
    - So there are alternative characterizations
    - e.g., should be obtained by a generally reliable method
- I'd hoped that philosophy would provide some help
    - It does provide insight and challenges
    - But no answers (but I need to look at philosophy of law)
- At issue here is the accuracy and completeness of our knowledge of the world
    - Insofar as it interacts with the system of interest
    - This seems mechanistic, not philosophical

Logic and Epistemology in Assurance Cases

- We have just two sources of doubt in an assurance case
- Logic doubt: the validity of the argument
  - Can be eliminated by formal verification
  - Subject to caveats discussed elsewhere
  - Automation allows what-if experimentation to bolster reviewer confidence
  - We can allow "because I say so" proof rules
- Epistemic doubt: the accuracy and completeness of our knowledge of the world in its interaction with the system
  - This is where we need to focus
- Same distinction underlies Verification and Validation (V&V)

Epistemology And Models

- We use formal verification to eliminate logic doubt
- That means we must present our assumptions in logic also
- This is where and how we encode our knowledge about the world
  - As models described in logic
- So our epistemic doubt then focuses on these models

Sometimes Less Is More

- Detail is not necessarily a good thing
- Because then we need to be sure the detail is correct
- For example, Byzantine faults
  - Completely unspecified, no epistemic doubt
- *vs.* highly specific fault models
  - Epistemic doubt whether real faults match the model

An Aside: Resilience

- To some extent, it is possible to trade epistemic and logic doubts
  - Weaker assumptions, fewer epistemic doubts
  - *vs.* more complex implementations, more logic doubt

- I claim resilience is about favoring weaker assumptions

- And it is the way of the future

Reducing Epistemic Doubt: Validity

- We have a model and we want to know if it is valid
- One way is to run experiments against it
- That's why simulation models are popular
- But models that support simulation are not so useful in formal verification nor, I think, in certification
  - To be executable, have to include a lot of detail
  - But our task is to describe assumptions about world, not implement it
- Recent advances in formal verification help overcome this
  - Infinite bounded model checking, enabled by SMT solving
  - Allows use of uninterpreted functions
  - With axioms/constraints encoded as synchronous observers
  - While still enjoying full automation

Reducing Epistemic Doubt: Completeness

- In addition to validity, we are concerned with the completeness of models
- E.g., have we recorded all hazards, all failure modes, etc.
- Traditional approaches: follow generally reliable procedure
  - E.g., ISO xxx for hazard analysis in medical devices
  - HAZOP, FMEA, FTA etc.
- Most of these can be thought of as manual ways to do model checking (state exploration) with some heuristic focus that directs attention to the paths most likely to be informative
- With suitable models we can do automated model checking and cover the entire modeled space
  - e.g., infinite bounded model checking, again
  - `check: FORMULA (system || assumptions) |- G(AOK => safe)`
  - Counterexamples guide refinements to system design and/or assumptions

Reducing Epistemic Doubt: Completeness (ctd).

- I have done examples illustrating the method above
  - e.g., propose `A1 implies P`, examine counterexample, hence discover `A1 and A2 implies P`

  This helps discover missing assumptions involving existing state variables

- I speculate that we can explore missing variables by adding an uninterpreted factor `X` to assumptions and examining the consequences through model checking

- e.g., `A1 and A2 and/or X implies P`

Compositional Assurance

- Use a MILS- or IMA-like architecture
  - Partitioning constrains possible interaction paths
- Then (epistemic) assumptions of one component
  - Become requirements on its environment
  - i.e., on the components it interacts with
- Can discover weak(est) environment assumptions by formal analysis/machine learning

Complications

- Some epistemic assumptions may be only probabilistically true
- Or we may have doubts/ignorance
  - Maybe these can be expressed probabilistically also
  - Or maybe some of the deductions in our verification are probabilistic (e.g., "because I say so")

  i.e., probabilities on terms vs. probabilities on rules
- System components that are not software
  - Use models
- Aside: top-level claims are often probabilistic
  - e.g., failure rate below 10^{-7} per hour for Level B
  - But the assurance objectives are all about correctness
  - Do more of them (or different ones) for higher levels

  Aha! it's about probability of perfection

Summary

- Two kinds of doubt: logic and epistemic
- Can eliminate logic doubt by automated verification
- Should focus on reducing epistemic doubt
- Often best accomplished by minimizing epistemic assumptions
- Hence models described by constraints not simulation models
- Can use automated verification to explore these
- SMT solvers are an enabling technology