What Use Is Verified Software?

John Rushby

Computer Science Laboratory
SRI International
Menlo Park, California, USA

Software and Systems

- The world at large cares little for verified software
- What it cares about is trustworthy systems
- So we need to examine the relationship between these
- I consider two perspectives

  Analytic: how does verified software contribute to system assurance?

  Synthetic: how can the technology of software verification best contribute to development of trustworthy systems?

Verified Software and System Assurance

- The system is generally more than software
  - Context, environment, hardware, people
- And trustworthiness is generally more than the properties we verify
  - Reliability, resilience, felicity, . . .
- So software verification is just one element in a larger body of evidence and argument, and we want to know how it all fits together
- This is worked out best in the context of assurance cases for certification of safety-critical systems
- In particular the idea of multi-legged assurance cases

Multi-Legged Assurance Cases

- We may use different kinds of evidence to support different (sub)claims
  - Field trials for user acceptance
  - Formal verification for algorithmic correctness
- Or multiple sources of evidence to support each other in a single claim
  - Testing
  - Plus verification
- We're interested in the second of these
  - Naively, an appeal to diversity
  - More credibly, consideration of uncertainties in each leg

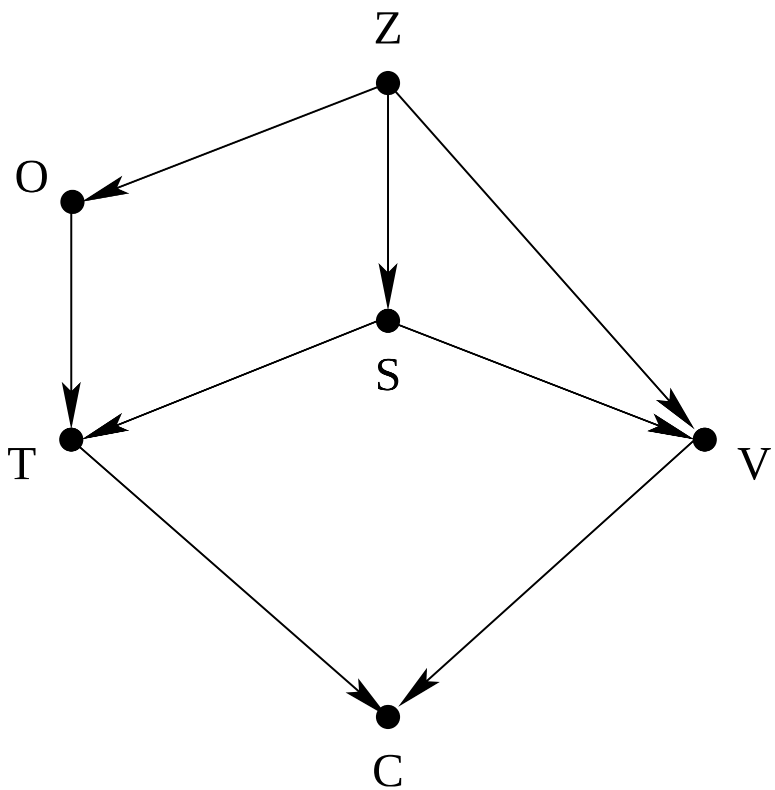
Two Kinds of Uncertainty In Certification

- One kind concerns failure of a claim, usually stated probabilistically (frequentist interpretation)
  - E.g., 10^{-9} probability of failure per hour, or 10^{-3} probability of failure on demand
- The other kind concerns failure of the assurance process
  - Seldom made explicit
  - But can be stated in terms of subjective probability
    - ⋆ E.g., 95% confident this system achieves 10^{-3} probability of failure on demand
    - ⋆ Note: this does not concern sampling theory and is not a confidence interval
- Demands for multiple sources of evidence are generally aimed at the second of these

Bayesian Belief Nets

- Bayes Theorem is the principal tool for analyzing subjective probabilities
- Allows a prior assessment of probability to be updated by new evidence to yield a rational posterior probability
  - E.g., P(C) vs. P(C | E)
- Math gets difficult when the models are complex
  - i.e., when we have many conditional probabilities of the form p(A | B and C or D)
- BBNs provide a graphical means to represent these, and tools to automate the calculations
- Can allow principled construction of multi-legged arguments

A BBN Example



Z: System Specification

O: Test Oracle

S: System's true quality

T: Test results

V: Verification outcome

C: Conclusion

Absolute Claims in Multi-Legged Arguments

- Can get surprising results (Littlewood and Wright)
  - Under some combinations of prior belief, increasing the number of failure-free tests may decrease our confidence in the test oracle rather than increase our confidence in the system reliability
- The anomalies disappear and calculations are simplified if one of the legs in a two-legged case is absolute
  - E.g., 95% confident that this claim holds... period
  - Formal methods deliver this kind of claim
  - E.g., Spark Ada (with the Examiner): guaranteed absence of run time exceptions
- Extends to multiple unconditional claims

Flies in the Ointment

- These results assume the verification leg considers the same system description and requirements as the other leg
- But this is seldom the case
  - Verification of weak properties: static analysis etc.
  - Verification of specific critical properties (subclaims)
  - Verification of abstractions of the real system
- It's a research challenge to develop the theory to cover these issues
- Aside: philosophers studying confirmation theory (part of Bayesian Epistemology) formulate measures of support differently than computer scientists
  - e.g., $c(C, E) = P(E \mid C) - P(E \mid \text{not } C)$

Verified Software and System Assurance, Redux

- The things we care about are system properties
- So certification focuses on systems
  - E.g., the FAA certifies airplanes, engines and propellers
- Dually, modern interpretations of accidents focus on systems issues, not component reliability
- Cf. Normal Accidents (Perrow)
  - Sufficiently complex systems can produce accidents without a simple cause—it's the system that fails
- Perrow identified interactive complexity and tight coupling as important factors

Verified Software and System Synthesis

- First, let's note that system accidents are dominant only because components have become reliable
  - And verified software can contribute here
- Next, let's apply formal verification to the dominant causes of system failure
  - Requirements (the integration explosion is a symptom)
  - Component interactions

Formal Analysis of Requirements

- Traditional requirements engineering is pre-scientific
- Asked to imagine the system and its interaction with its environment
- Then anticipate component interactions and malfunctions
- Outputs are documents in Word
- Model-based design provides an opportunity to do better
- Build models of environment, components, faults, people
- And calculate their interactions
- Formal methods provide the technology to calculate all possible scenarios (within the model)
  - This is its unique capability
  - Opportunity to mechanize hazard analysis, FTA etc.
  - Will often involve infinite-state and hybrid systems

Verified Software Interactions

- We should extend the focus of formal verification from correctness of components to correctness of interactions
- This requires new(er) kinds of specification
  - e.g., interface automata
- And new(er) kinds of analysis
  - e.g., assumption generation
- And new(er) roles for formal methods
  - e.g., monitor synthesis
  - e.g., test generation for integration and system tests

Conclusions

- The Verified Software Initiative will not achieve its full potential if it focuses narrowly on code verification
- One challenge is to better understand contribution of verification to multi-legged system assurance cases
    - In particular, the value of verified weak properties
- Another is to extend verification technology in ways that help system developers
    - Formal requirements exploration and analysis
    - Verification of interfaces and interactions
    - Generation of system tests
- All of this has to be automated
    - Additional benefit is that we can then cope with change
- Exploit the unique benefit of formal verification: ability to consider all possible cases