1st International Workshop on Argument for Agreement and Assurance (AAA 2013), Kanagawa Japan, October 2013

Mechanized Support For Assurance Case Argumentation

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Introduction

- I'm from a group that does formal verification
  - PVS, SAL, Yices, ETB are some of our tools

  Everything looks like a proof to me
- But I have come to realize an assurance case is not a proof
- There are inherent uncertainties, like identifying all hazards
- So an assurance case is an inductive argument
  - Probable truth of premises

    indicates probable truth of conclusion
- But then we really ought to quantify the probabilities
- Lots of ideas for combining logic and probability
  - Probabilistic logics, Dempster-Shafer, BBNs etc.

  But none are universally accepted
- So does that condemn us to informal reasoning?

Assurance Case Evaluation

- An assurance case is a big argument
  - Needs reliable review
- John Knight and colleagues examined three cases
  - All had basic reasoning flaws
  - Different reviewers found different flaws
- Need mechanized support for reliability and economy
- Some of the subcases are going to involve formal verification
  - And those tools are powerful (SMT solvers etc.)
- So can we extend these tools to larger aspects of the case?
- Classical method
  - Embed uncertainty in the premises
  - Argument based on these should be deductively sound
- Cf. formal verification of fault tolerant algorithms in 1980s
  - Formally verify algorithm, assuming no more than f faults
  - Separately, estimate probability of $> f$ faults

Epistemic and Logic Uncertainty

- Epistemic uncertainty
  - How accurate is our knowledge of the system, its environment, assumptions, etc.?
- Logic uncertainty
  - Given our knowledge, how accurate is our reasoning?
- Proposal: encode our knowledge in logic
  - How best to do that? See SafeComp 2013 paper
  - Software is logic
  - Encode the rest as constraint-based models

  These are our premises
- Formally verify the argument based on the premises
  - Eliminates logic doubt (modulo soundness of prover)
- But need to incorporate the evidence supporting our premises

Attaching Informal Justifications

- We will have premises that say things like
  - A, B, and C are all the hazards

  And the assurance argument enumerates over these
- We need a way to attach the evidence for this premise
  - e.g., description of the hazard analysis process used

  To the formal verification that uses it
- Simple proposal in SSS 2010
- Given premise named `N`, formalized as `p` write it as

  `good_doc(N) IMPLIES p`
- Attach evidence for `N` to uninterpreted predicate `good_doc(N)`
  - Formally this is just a comment
- Enable the predicate (i.e., set it to `true`) only when reviewers are satisfied with the evidence
  - Can have more complex arrangement if multiple reviewers
  - ETB mechanisms should make this more transparent

Argumentation

- But this all neglects the argumentation aspect
  - Need to allow reviewers to challenge and explore
  - e.g., conduct "what if" experiments

- Vast amount of work on formal argumentation, defeasible reasoning etc.
  - That's why I'm here: to learn about that

- But I'm also interested in how the representation of an assurance case in a verification system can be extended to support defeasible reasoning

Argumentation: A Really Simple Proposal

- Like the `good_doc` predicates we use to attach evidence, we can attach defeater predicates

- Premise named `N`, formalized as `p` becomes

  `NOT` d_N `IMPLIES p`

  Where d_N is the defeater for `N`, initially `false`

- Conduct "what if" exercises by toggling defeaters and letting the automation rip
  - Counterexamples (often) help insight

- With SMT automation, it would be easy to provide a GUI with switches and dials

Example

- In the paper I do a small example from Michael Holloway
- I do it PVS (an interactive theorem prover)
  - Mainly because we lack a sugared syntax for SMT
- Demo in the final session
- Here's the idea. . .

Idea of the Example

- Three hazards identified: `H1`, `H2`, `H3`
- Subarguments that each has been adequately mitigated
- Assumption (premise): No other hazards
- Therefore safe by "enumerate over hazards" pattern √
- Challenge: what about joint occurrence of two hazards?
  - Specifically `H2` and `H3`
- Aha! Add new hazard `H23` and assert that it is mitigated by evidence provided for `H2` and `H3` separately √
- Evidence for each not evidence for both: turn on defeater X
- New evidence: combo used previously in similar system √
- Not similar enough: turn on defeater X
- OK, neither argument is convincing on its own
  - But together they are persuasive
- Hmm, modify so either defeater can be on, but not both √

Discussion

- Technical
  - The manipulations performed here are all propositional
  - The horsepower of SMT etc. is needed only in the details
  - So could maybe combine SAT-based methods of argumentation with powerful lower-level automation
    - ⋆ Just as an SMT solver is SAT plus decision procedures
  - Alternatively put an outer loop above the SMT solver
    - ⋆ That is how MaxSAT and AllSAT are done

- Philosophical
  - Does this really capture what argumentation is about?
  - Does argumentation really capture what assurance cases are about?