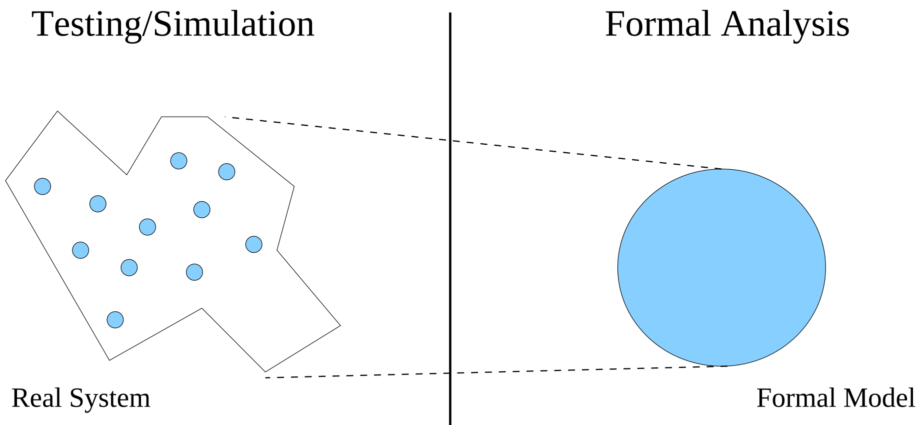
Formal Methods Assurance for TTP

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Formal Methods for Analysis and Assurance: The Idea



- ● Partial coverage

- ● Complete coverage (of the modeled system)

  Accurate model: verification

  Approximate model:debugging

Formal Methods

- Build a mathematical model of the system or algorithm of interest
- And of its environment
  - If it's a fault-tolerant system, the environment includes the fault model
- Examine all possible behaviors of the system in interaction with its environment

  Model checking: by brute force enumeration (finite model)

  Theorem proving: using symbolic representations

  cf. `5*5-3*3 = (5-3)(5+3)` and $x^2 - y^2 = (x - y)(x + y)$
- Model checking is largely automatic; it's good for debugging
  - Must often approximate to get finite model, but experience is that you find more bugs by exploring all the behaviors of an approximation than just some of the behaviors of the real thing (then called formal refutation)
- Theorem proving is automated but requires skilled human guidance;

  it's good for assurance (then called formal verification)

Formal Methods Assurance and TTP

- TTP/C is already highly assured by traditional means
  - Testing; fault injection; field experience
- But formal methods provide complementary assurance
  - Complete exploration of model vs. partial exploration of implementation
- Valuable for highly critical applications with strong certification requirements
  - Allowed or encouraged for some (DO-178B and DO-180 for aircraft)
  - Required by others (UK DEF-STAN 00-55)
- Provides insight and design tools for developers of advanced TTP concepts
- Provides precise, accurate formal specification of system and its assumptions, properties, interfaces
  - Including exact characterization of worst case fault model
- Provides foundation for formal assurance of applications that use TTP

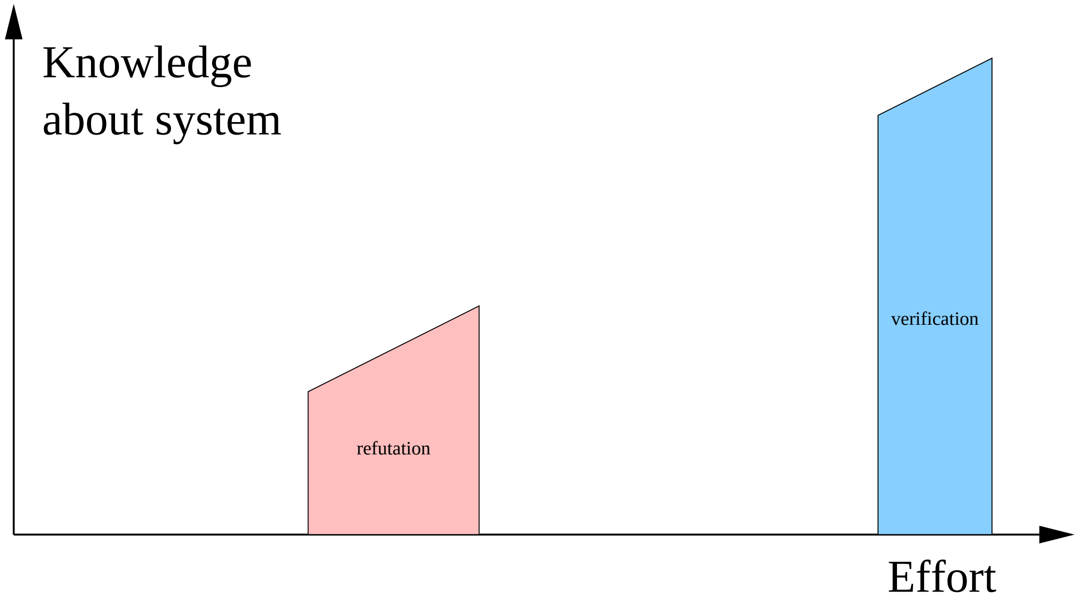
Context For Our Work

- NASA project to develop and apply formal methods assurance for advanced FADEC being developed by GE/Honeywell
- Architecture uses a novel TTP configuration
- Formal assurance being performed by SRI, with collaboration from University of Ulm (Holger Pfeifer in the group of Prof. Friedrich von Henke)
- Uses various model checkers and SRI's theorem proving system PVS
- Initial focus on clock synchronization and group membership

Status and Near-Term Plans

- Have formally verified a simplified version of the membership algorithm using PVS
  - Required development of new formal verification method
  - Developing formal verification of the full algorithm
- Also have a formal specification suitable for model checking
  - Will use to help evaluate FADEC architecture
- Have formally verified clock synchronization algorithm using PVS
  - Rather complex, strong fault model
  - Developing new treatment, hybrid fault model
- Developing formal model of the interaction of synchronization and group membership (each depends on the other)
- Completion: end of 2001

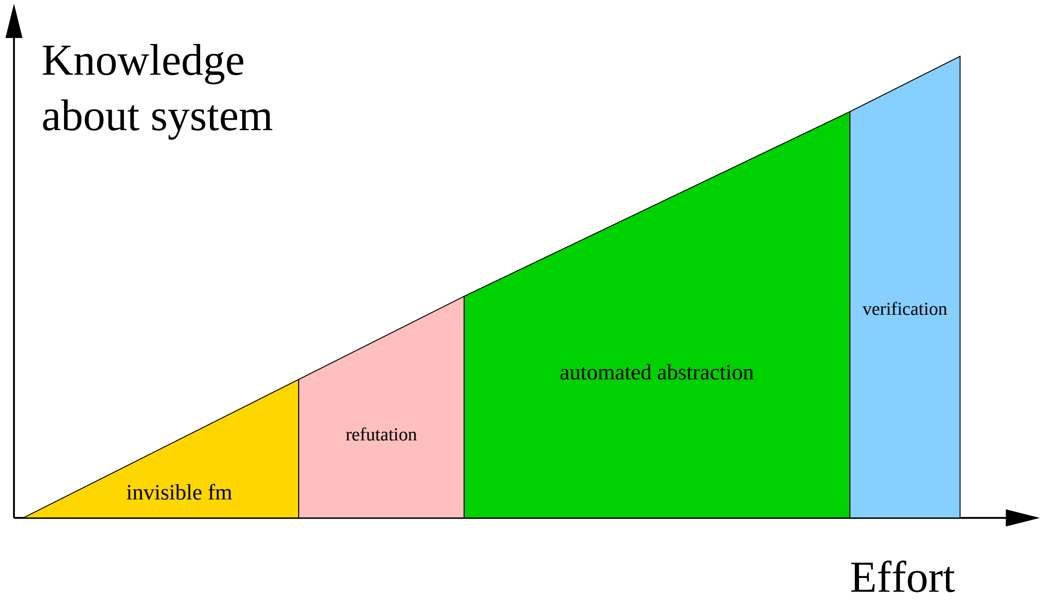
Effort for Formal Refutation and Verification



Longer Term Plans

- Near-term focus is on the algorithms of TTP itself (internal concerns)
- Longer-term addresses systems built on TTP (external concerns)
  - Sponsored by DARPA and by NASA
  - Formal assurance from control model (e.g., in Matlab)
  - Down to fault-tolerant implementation
  - That uses the services of TTP (and its tool chain)
- Already formally verified a model of time-triggered computation
- One goal is to make “lite” formal methods “disappear” into the standard engineering process; for example
  - Strong static checking for Stateflow (all cases covered etc.)
  - Formal test-case generation
- Deployment: 2003

Disappearing Formal Methods



Summary and Prospects

- Formal assurance for the internal mechanisms of TTP provides added value to all users at no cost
  - An additional discriminator in some markets
- Integrated formal assurance path from external application to TTP implementation will be valuable in highly critical systems
  - Could also provide formal verification for TTP implementation (μarch and μcode)
  - We previously verified a complete avionics processor at this level (Collins AAMP)
- And “disappearing” formal methods could reduce development time and costs for all systems
- Formal methods tools are under continuous development and rapidly becoming more powerful and usable
  - We’ve been building them for 25 years
  - New version of PVS, and new systems SAL and ICS will be released in 2001

To Learn More

- Check out papers and technical reports at `http://www.csl.sri.com/fm.html`, in particular
  - ○ Verification Diagrams Revisited: Disjunctive Invariants for Easy Verification `http://www.csl.sri.com/~rushby/cav00.html`
  - ○ Formal Verification of the TTP Group Membership Algorithm `http://www.informatik.uni-ulm.de/ki/PVS/membership.html`
  - ○ Formal Verification for Time-Triggered Clock Synchronization `http://www.informatik.uni-ulm.de/ki/PVS/tta-clocksync-dcca7.html`
  - ○ Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms `http://www.csl.sri.com/reports/html/tse99.html`
- Information about our verification system, PVS, and the system itself are available from `http://pvs.csl.sri.com`
  - ○ Freely available under license to SRI
  - ○ Built in Allegro Lisp for Solaris, or Linux