Synthesis of Distributed Protocols from Scenarios and Requirements

Abhishek Udupa, Christos Stergiou, Mukund Raghothaman, Stavros Tripakis, Milo Martin, Rajeev Alur
University of Pennsylvania, University of California, Berkeley, Aalto University

**Goal: Assist Programmers of Distributed Protocols**

- Descriptions only specify common-case behaviors
- Corner-cases often unspecified
- Traditional methodologies need global reasoning
- **T**edious **c**ase-by-**c**ase analysis
- Fixing errors/bugs also needs global reasoning
- Can we simplify the design process?

**The Scenario Based Approach**

- Skeleton (Communication Architecture, State Variables)
- Symbolic or Concrete Scenarios
- Describe executions of the system
- Formal requirements also used

**Protocol Description = Skeleton + Scenarios + Formal Specs**

**TRANSIT**

- Programmer in the loop
- Used to design the SGI Origin Cache Coherence Protocol (An Industrial Protocol)
- Programmer provides (symbolic) scenarios initially
- Errors fixed by adding (concrete) scenarios

**Synthesis of Finite-State Protocols**

- **F**ully automated synthesis
  - Automata with no state variables
  - Programmer provides *incomplete automata* and spec
  - Goal: produce complete automata that satisfy spec
  - Synthesized ABP with just **one** scenario out of four

**Ongoing Work**

- Symbolic Distributed Protocols
- Extended Finite State Machines
- Machines have state variables
- Transitions update variables **symbolically**
- Standard way of describing protocols

**Example EFSM:**

- Candidate space of solutions is much larger
- Counterexamples more complex to generalize
- Utilize allowed syntax for symbolic expressions to prune large search space

**Publications:**

- TRANSIT: Specifying Protocols with Concolic Snippets (PLDI 2013)
- Synthesizing Finite-state Protocols from Scenarios and Requirements (HVC 2014)