Verifying Distributed Adaptive Real (DART) Systems

Functional Verification
Prove application-controller contract for unbounded time
- Previously limited to bounded verification only
Prove controller-platform contract via hybrid reachability analysis
- Done by AFRL
Working on automation and asynchronous model of computation

Proactive Self-Adaptation Using Probabilistic Model Checking

DART Vision
A sound engineering approach based on the judicious use of precise semantics, formal analysis and design constraints leads to assured behavior of (DART) systems while accounting for:
- critical requirements
- probabilistic requirements
- uncertain environments
- necessary coordination
- assurance at source code level

DART Process
1. Enables compositional and requirement specific verification
2. Use proactive self-adaptation and mixed criticality to cope with uncertainty and changing context
3. ZSRM Schedulability (Timing)
4. Software Model Checking (Functional)
5. Statistical Model Checking (Probabilistic)

DART Architecture
Software for guaranteed requirements, e.g., collision avoidance protocol must ensure absence of collisions
Software for probabilistic requirements, e.g., adaptive path-planner to maximize area coverage within deadline

DMPL: DART Modeling and Programming Language
- C-like language that can express distributed, real-time systems
- Semantics are precise
- Supports formal assertions usable for model checking and probabilistic model checking
- Physical and logical concurrency can be expressed in sufficient detail to perform timing analysis
- Can call external libraries
- Generates compilable C++
- Developed syntax, semantics, and compiler (dmplc)

Example: Self-Adaptive and Coordinated UAS Protection

Distributed Statistical Model Checking
Batch Log and Analyze

DMPL supports the right level of abstraction, github.com/cps-set/dart

DMPL: DART Modeling and Programming Language
- Technical and operational validity
- Engaged stakeholders
- Problem (DART prototype)
- Demonstrate on DoD-relevant model
- Technical and operational validity

Example: Self-Adaptive and Coordinated UAS Protection

Challenges:
- compute the probability of mission success & compare between different adaptation strategies.
- Solution: Statistical Model Checking

Statistical Model Checking of Distributed Adaptive Real-Time Software. David Work, Jeffrey Hansen, Sagar Chaki in Proc. of Runtime Verification 2015

Contact: Sagar Chaki | chaki@sei.cmu.edu