Quantitative Verification: Correctness, Reliability and Beyond

Dave Parker
University of Birmingham

December 2012
Verification

• Checking the correctness of (computerised) systems using rigorous, mathematically-sound techniques
  – in essence: proving that a piece of software, or hardware, or a protocol behaves correctly

• Automated verification: model checking
  – correctness properties expressed in temporal logic
  – exhaustive construction/analysis of finite-state model
Model checking

• Successful in practice
  – e.g. Windows device drivers, circuit designs, ...

• Example properties
  – “acquire/release of spinlock is always done in strict alternation”
  – “no array is accessed outside its bounds”

• Why it works
  – temporal logic: expressive, tractable
  – fully automated, tools available
  – not just verification, but falsification of properties, i.e. bug hunting
Quantitative verification

- Adds quantitative aspects (to models and properties)
  - probability, time, costs, rewards, ...

- Probability
  - physical components can fail
  - communication media are unreliable
  - algorithms/protocols use randomisation

- Time
  - delays, time-outs, failure rates, ...

- Costs & rewards
  - power consumption, resource usage, ...
  - profit, incentive schemes, ...
Probabilistic model checking

• Construction and analysis of probabilistic models
  – Markov chains, Markov decision processes, ...

• Correctness properties in probabilistic temporal logic
  – $P_{>0.999} \Box (\text{trigger} \rightarrow \Diamond \leq 20 \text{ deploy})$
  – “the probability of an airbag always deploying within 20ms of being triggered is at least 0.999”
  – correctness, reliability, performance, ...

• Model checking algorithms (and tools)
  – graph algorithms, linear equations, linear programming, numerical fixed points, numerical approximations, ...
Probabilistic models

+ probabilities

Markov chain

e.g. communication protocol

+ exponential time delays

Continuous-time Markov chain

e.g. systems biology

+ game theory

Stochastic game

e.g. energy management
1) Adding: Probabilities

• Model: Markov chain
  – add probabilities to transitions

• Properties
  – probability of airbag failure < 0.001
  – numerical queries: what is the probability of failure?

• Key ideas:
  – exact numerical results
  – combines numerical + exhaustive analysis
  – results show system flaws, anomalies

• Applications
  – network protocols, security, biology, robotics & planning,
    power management, nanotechnology...
Example: Bluetooth

- Device discovery between a pair of Bluetooth devices
  - performance essential for this phase

- Complex discovery process
  - two asynchronous 28-bit clocks
  - pseudo-random hopping between 32 frequencies
  - random waiting scheme to avoid collisions
  - 17,179,869,184 initial configurations

- Probabilistic model checking
  - “worst-case expected discovery time is at most 5.17s”
  - “probability discovery time exceeds 6s is always < 0.001”
2) Adding: Exponential delays

- Continuous-time Markov chains
  - random delays on transitions between states
  - delays are exponentially distributed
  - e.g. failure rates, reaction times, ...

- Applications
  - network performance models
  - biological reactions

- Properties
  - probability of disk-failure within 1 month?
  - expected number of molecules of X at time instant T?
Example: Systems biology

• Markov model of reactions
  – states represent molecule counts
  – transitions correspond to reactions

• Key ideas
  – “in-silico” experiments
  – aim: validate biologists’ models
  – probabilistic model checking can be cheaper than simulation
  – small models yield useful results

• Case study: FGF pathway
  – model developed with biologists
  – validated against lab experiments
3) Adding: Game theory

- Multi-player stochastic games
  - states controlled by players
  - players choose (probabilistic) actions

- Key ideas
  - automated methods essential to reason about complex player strategies, and interaction with probabilities

- Property specifications
  - does player 1 have a strategy to ensure that the probability of is <0.01, regardless of the strategies if players 2 and 3?

- Applications
  - controller synthesis (controller vs. environment), security (system vs. attacker), distributed algorithms, ...
Example: Energy management

- Energy management protocol for Microgrid
  - Microgrid: local energy management
  - randomised demand management protocol
  - probability: randomisation, demand model, ...

- Existing analysis
  - simulation-based
  - assumes all clients are unselfish

- Our analysis
  - stochastic multi-player game
  - clients can cheat (and cooperate)
  - exposes protocol weakness
  - propose/verify simple fix

[Hildmann/Saffre’11]
Conclusions

• Quantitative verification
  – formal methods to build/analyse probabilistic models
  – temporal logics for correctness, reliability, performance, ...
  – exact results, combines numerical + exhaustive analysis
  – wide range of applications

• Challenges
  – scalability + efficiency
  – wider property classes, e.g. partial information for games
  – richer models: timed games, hybrid automata, ...