Quantitative Verification: Correctness, Reliability and Beyond

Dave Parker
University of Birmingham

Google, October 2013
Outline

• Verification and model checking
• Quantitative verification
• Probabilistic model checking and PRISM

- Discrete time Markov chains
- Adding continuous-time...
  - continuous-time Markov chains
- Adding nondeterminism...
  - Markov decision processes
- Adding game theory...
  - stochastic multi-player games
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

Ariane 5, flight 501

Toyota Prius

Infusion pumps
Model checking

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

• Successful in practice
  – e.g. Windows device driver development
  – example property: “acquire/release of spinlock always strictly alternate”

• Why it works
  – temporal logic: expressive, tractable
  – fully automated, tools available
  – not just verification, but falsification, 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
  – energy consumption, resource usage, ...
  – profit, incentive schemes, ...
Quantitative verification

- Correctness properties are **quantitative**
  - "the probability of an airbag failing to deploy within 0.02 seconds of being triggered is at most 0.001"
  - "with probability 0.99, the packet arrives within 10 ms"

- Beyond correctness:
  - reliability, timeliness, performance, efficiency, ...
  - "the expected energy consumption of the sensor"
  - "the expected number of FGF ligands after 20 minutes"
Probabilistic model checking
Probabilistic model checking

• Construction and analysis of probabilistic models
  – for example: discrete-time Markov chains (DTMCs)
  – transitions labelled with probabilities
  – from a description in a high-level modelling language

• Correctness properties expressed in probabilistic temporal logic, e.g. PCTL
  – $\text{trigger} \rightarrow P_{\geq 0.999} [ F_{\leq 2} \text{deploy} ]$
  – “the probability of the airbag deploying within 2 time units of being triggered is at least 0.999”
Probabilistic model checking

• Computation of "exact" results (e.g. probabilities)
  – graph algorithms, linear equations, linear programming, numerical fixed points, numerical approximations, ......

• Combines numerical and exhaustive analysis
  – results show system flaws, anomalies

• Flexible and widely applicable
  – many types of models, properties
  – fully automated + tool support

• Scalability and efficiency remains a challenge
  – but many advances in efficient techniques
PRISM

• PRISM: open source probabilistic model checker
  – developed at Birmingham/Oxford University, since 1999
  – wide range of probabilistic models, temporal logics
  – modelling language, GUI, scalable/efficient techniques

• Leading probabilistic verification tool
  – research/teaching in 50+ institutions
  – 34,000 downloads, 250 external PRISM-related papers

• Case studies
  – network protocols, security, biology, robotics, power management, airbag system, cloud computing...

• See: www.prismmodelchecker.org
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 (PRISM)
  – “probability discovery time exceeds 6s is always < 0.001”
  – “worst-case expected discovery time is at most 5.17s”
Outline

- Discrete time Markov chains

- Adding continuous-time...
  - continuous-time Markov chains

- Adding nondeterminism...
  - Markov decision processes

- Adding game theory...
  - stochastic multi-player games
Adding continuous time...
Adding continuous time...

- Continuous-time Markov chains
  - random (real-valued) transition delays
  - delays are exponentially distributed
  - e.g. failure rates, reaction times, ...

 Failures/repairs in a cluster of 3 workstations
Adding continuous time...

- Continuous-time Markov chains
  - random (real-valued) transition delays
  - delays are exponentially distributed
  - e.g. failure rates, reaction times, ...

Reactions between proteins A, B & AB

\[ A + B \underset{k_3}{\overset{k_2}{\rightleftharpoons}} AB \]

\[ 1 - e^{-\lambda t} \]

Diagram: States A, A,B,B, A,B,AB, AB,AB, A,B, B,AB, B,B with transitions 4k_1, k_1, 2k_2, k_2, 2k_3, 2k_1, k_3.
Continuous-time Markov chains

• Properties (temporal logic CSL)
  – $S_{>0.999 \ [ up ]}$: "long-run probability of availability is $>0.999$"
  – $P_{=? \ [ down \ U^{\geq60} \ repair ]}$: "what is the probability that it takes longer than 1 hour to recover from a server failure?"
  – $R_{A=? \ [ I=T ]}$: "expected number of molecules of A at time T?"

• Applications
  – performance evaluation and reliability analysis
  – systems biology: “in-silico” experiments to validate biologists’ models; later compared to lab results
Example: DNA computing

• DNA Strand Displacement language (DSD)
  – for designing DNA circuits [Cardelli, Phillips, et al.]
  
  ```
  t x
  t a
  t x t a t a
  y t
  x t y t a t
  ```
  – reactions naturally modelled as CTMCs

• Analysis of a DNA transducer design
  – correctness: A [ G deadlock → all_done ]
    design flaw (due to cross talk)
    automatically detected
  – performance-based design decisions:
    with or without garbage collection?
Adding nondeterminism...
Adding nondeterminism...

- Markov decision processes (MDPs)
  - generalise DTMCs by adding **nondeterminism**

- Nondeterminism: unknown behaviour
  - concurrency, abstraction, user input, control

- Strategies (or "policies", "adversaries")
  - resolve nondeterminism based on current history
Markov decision processes

• Two (dual) problems:

• 1. Verification
  – quantify over all possible strategies (i.e. worst-case)
  – $P_{<0.01} [ F_{\text{err}} ]$: “the probability of error is always $< 0.01$”
  – applications: randomised communication protocols, randomised distributed algorithms, security, ...

• 2. Strategy synthesis
  – $P_{<0.01} [ F_{\text{err}} ]$: "does there exist a strategy for which the probability of an error occurring is $< 0.01$?"
  – applications: robotics, power management, security, ...
Example: Power management

• Dynamic power management controllers
  – for an IBM TravelStar VP disk drive
  – switch between power modes: active/idle/idlelp/stby/sleep
  – PRISM model of power manager, disk request queue, etc.

• Build controllers that
  – minimise energy consumption, subject to constraints on e.g.
  – (i) probability that a request waits more than K steps
  – (ii) expected number of lost disk requests
Adding game theory...
Adding game theory...

• **Stochastic multi-player games**
  – states controlled by players
  – players choose actions in states
  – strategies for each player

• **Key ideas**
  – models *competitive* and/or *collaborative* behaviour
  – automated methods essential to reason about complex player strategies, and interaction with probabilities
Stochastic multi-player games

• Property specifications (temporal logic rPATL)
  – $\langle\langle\{1,2\}\rangle\rangle \ P_{\geq 0.95} [ F^{\leq 45} done ]$ : "can nodes 1 and 2 collaborate so that the probability of the protocol terminating within 45 seconds is at least 0.95, whatever nodes 3 and 4 do?"

• Model checking
  – zero sum properties: analysis reduces to 2-player game
  – PRISM-games: www.prismmodelchecker.org/games

• 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
  – random back-off when demand is high

• Original analysis [Hildmann/Saffre'11]
  – protocol increases "value" for clients
  – simulation-based, clients are honest

• Stochastic multi-player game model
  – clients can cheat (and cooperate)
Example: Energy management

- Exposes protocol weakness
  - incentive for clients to act selfishly

- We propose a simple fix (and verify it)
  - clients can be punished

### Graphs

**Value per client**
- All follow alg.
- No use of alg.
- Deviations of varying size

**Value per client, with fix**
- All follow alg.
- Deviations of varying size

**Graph Details**

- **X-axis**: Number of clients
- **Y-axis**: Value per client

---

*Note: Graphs show a decrease in value per client as the number of clients increases, with and without the fix.*
Conclusions

• Quantitative verification
  – probabilistic model checking & PRISM
  – formal methods to build/analyse probabilistic models
  – temporal logics for correctness, reliability, performance, …
  – exact results, combines numerical + exhaustive analysis
  – flexible approach, wide range of applications

• Key challenges
  – scalability + efficiency: state space explosion
  – richer models: continuous space, hybrid systems, …
  – user friendly languages for model/property specification