Lecture 1
Introduction

Dr. Dave Parker

Department of Computer Science
University of Oxford

Probabilistic model checking

- Probabilistic model checking…
  - is a formal verification technique
    for modelling and analysing systems
    that exhibit probabilistic behaviour
- Formal verification…
  - is the application of rigorous,
    mathematics-based techniques
    to establish the correctness
    of computerised systems

Outline

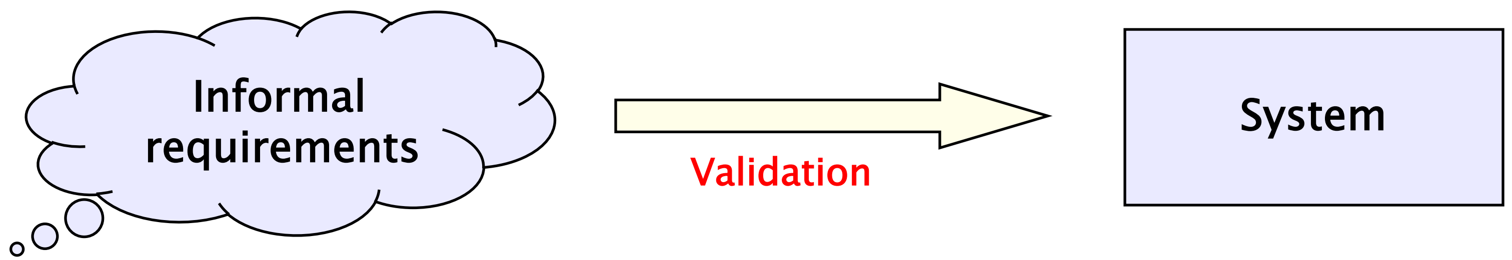
- Introducing probabilistic model checking…

- Topics for this lecture
  - the role of automatic verification
  - what is probabilistic model checking?
  - why is it important?
  - where is it applicable?
  - what does it involve?

- About this course
  - aims and organisation
  - information and links

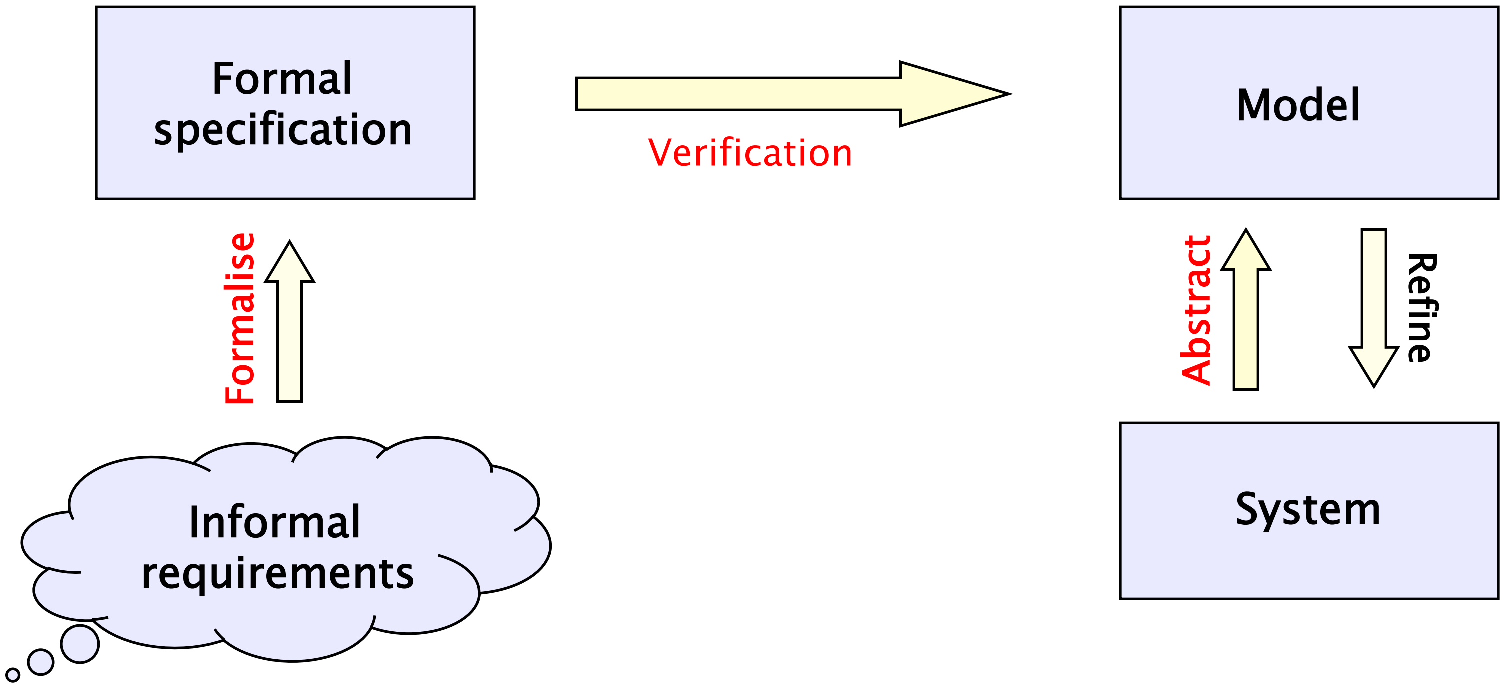
Conventional software engineering

- From requirements to software system
  - apply design methodologies
  - code directly in programming language
  - validation via testing, code walkthroughs



Formal verification

- From requirements to formal specification
  - formalise specification, derive model
  - formally verify correctness



But my program works!

- True, there are many successful large-scale complex computer systems…
  - online banking, electronic commerce
  - information services, online libraries, business processes
  - supply chain management
  - mobile phone networks
- Yet many new potential application domains with far greater complexity and higher expectations
  - automotive drive-by-wire
  - medical sensors: heart rate & blood pressure monitors
  - intelligent buildings and spaces, environmental sensors
- Learning from mistakes costly…

Toyota Prius

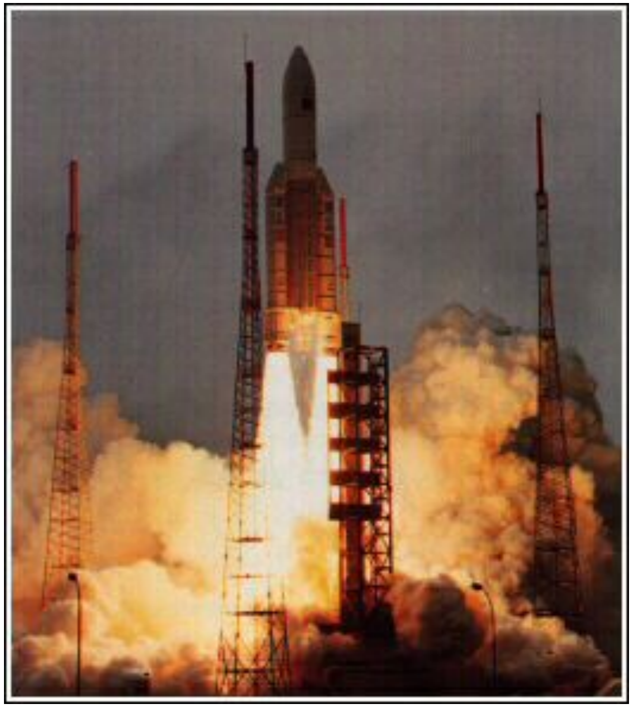
- Toyota Prius
  - first mass-produced hybrid vehicle

- February 2010
  - software "glitch" found in anti-lock braking system
  - in response to numerous complaints/accidents

- Eventually fixed via software update
  - in total 185,000 cars recalled, at huge cost
  - handling of the incident prompted much criticism, bad publicity

Ariane 5

- ESA (European Space Agency) Ariane 5 launcher
  - shown here in maiden flight on 4th June 1996
- 37secs later self-destructs
  - uncaught exception: numerical overflow in a conversion routine results in incorrect altitude sent by the on-board computer
- Expensive, embarrassing...



The London Ambulance Service

- London Ambulance Service computer aided despatch system
  - Area 600sq miles
  - Population 6.8million
  - 5000 patients per day
  - 2000-2500 calls per day
  - 1000-1200 999 calls per day

- Introduced October 1992
- Severe system failure:
  - position of vehicles incorrectly recorded
  - multiple vehicles sent to the same location
  - 20-30 people estimated to have died as a result

What do these stories have in common?

- Programmable computing devices
  - conventional computers and networks
  - software embedded in devices
    - airbag controllers, mobile phones, etc
- Programming error direct cause of failure
- Software critical
  - for safety
  - for business
  - for performance
- High costs incurred: not just financial
- Failures avoidable…

Why must we verify?

"Testing can only show the presence of errors, not their absence."

To rule out errors need to consider all possible executions often not feasible mechanically!

- need formal verification…

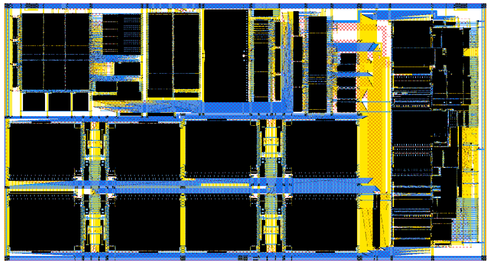
"In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, computers are without precedent in the cultural history of mankind."

Edsger Dijkstra

1930–2002

Automatic verification

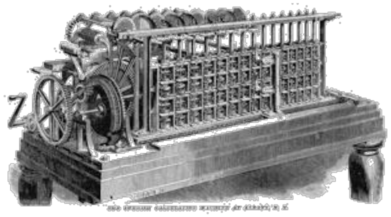
- Formal verification…
  - the application of rigorous, mathematics-based techniques to establish the correctness of computerised systems
  - essentially: proving that a program satisfies it specification
  - many techniques: manual proof, automated theorem proving, static analysis, model checking, …



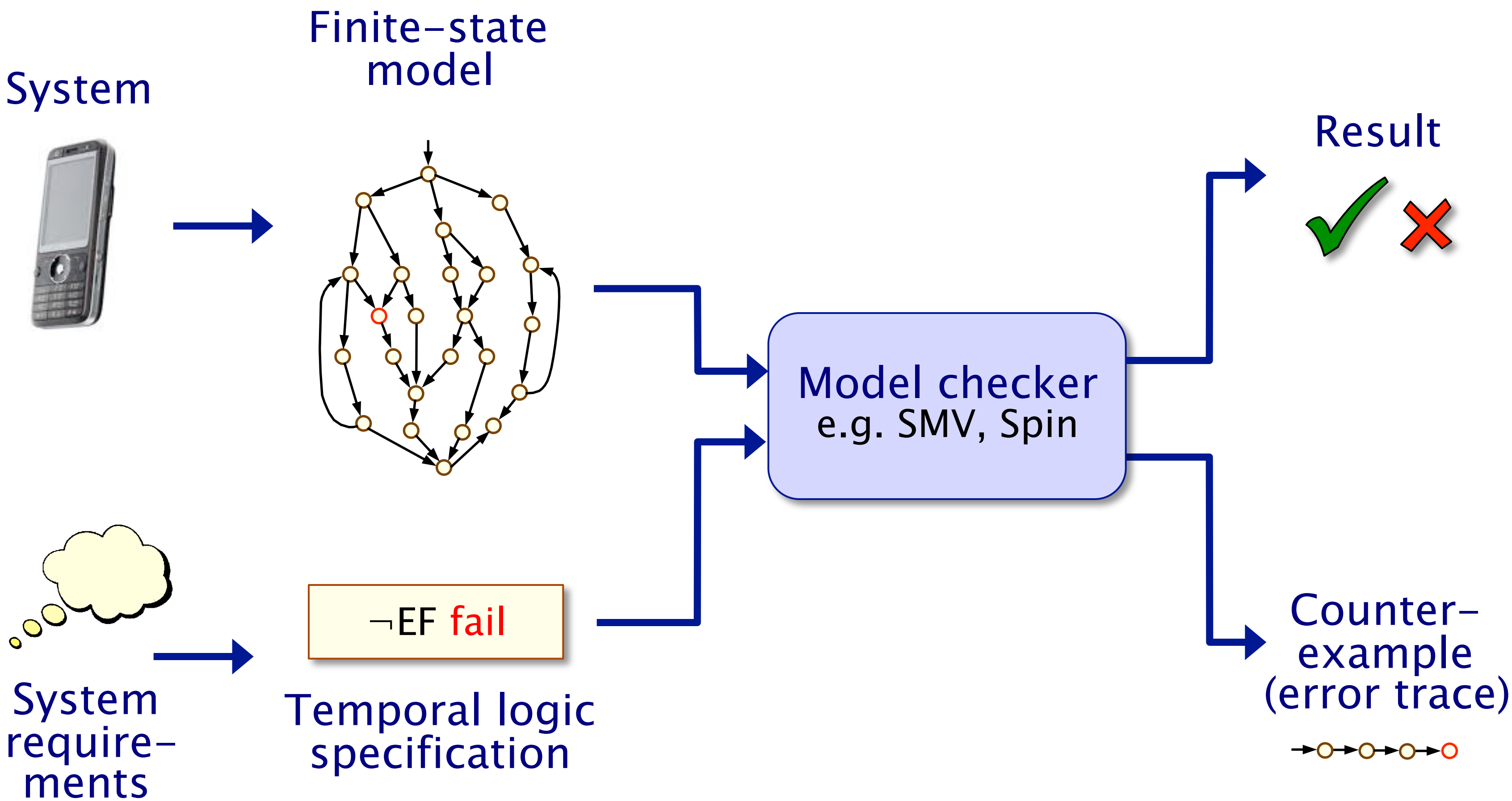
$10^{500,000}$ states

10^{70} atoms

- Automatic verification =
  - mechanical, push-button technology
  - performed without human intervention



Verification via model checking



Model checking in practice

- Model checking now routinely applied to real-life systems
  - not just “verification”…
  - model checkers used as a debugging tool
  - at IBM, bugs detected in arbiter that could not be found with simulations
- Now widely accepted in industrial practice
  - Microsoft, Intel, Cadence, Bell Labs, IBM,...
- Many software tools, both commercial and academic
  - smv, SPIN, SLAM, FDR2, FormalCheck, RuleBase, ...
  - software, hardware, protocols, …
- Extremely active research area
  - 2008 Turing Award won by Edmund Clarke, Allen Emerson and Joseph Sifakis for their work on model checking

New challenges for verification

- Devices, ever smaller
  - laptops, phones, sensors…
- Networking, wireless, wired & global
  - wireless & internet everywhere
- New design and engineering challenges
  - adaptive computing, ubiquitous/pervasive computing, context-aware systems
  - trade-offs between e.g. performance, security, power usage, battery life, …

New challenges for verification

- Many properties other than correctness are important
- Need to guarantee…
  - safety, reliability, performance, dependability
  - resource usage, e.g. battery life
  - security, privacy, trust, anonymity, fairness
  - and much more…

- Quantitative, as well as qualitative requirements:
  - "how reliable is my car's Bluetooth network?"
  - "how efficient is my phone's power management policy?"
  - "how secure is my bank's web-service?"

- This course: probabilistic verification

Why probability?

- Some systems are inherently probabilistic…

- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding

- Examples: real-world protocols featuring randomisation
  - Randomised back-off schemes
    - IEEE 802.3 CSMA/CD, IEEE 802.11 Wireless LAN
  - Random choice of waiting time
    - IEEE 1394 Firewire (root contention), Bluetooth (device discovery)
  - Random choice over a set of possible addresses
    - IPv4 Zeroconf dynamic configuration (link-local addressing)
  - Randomised algorithms for anonymity, contract signing, …

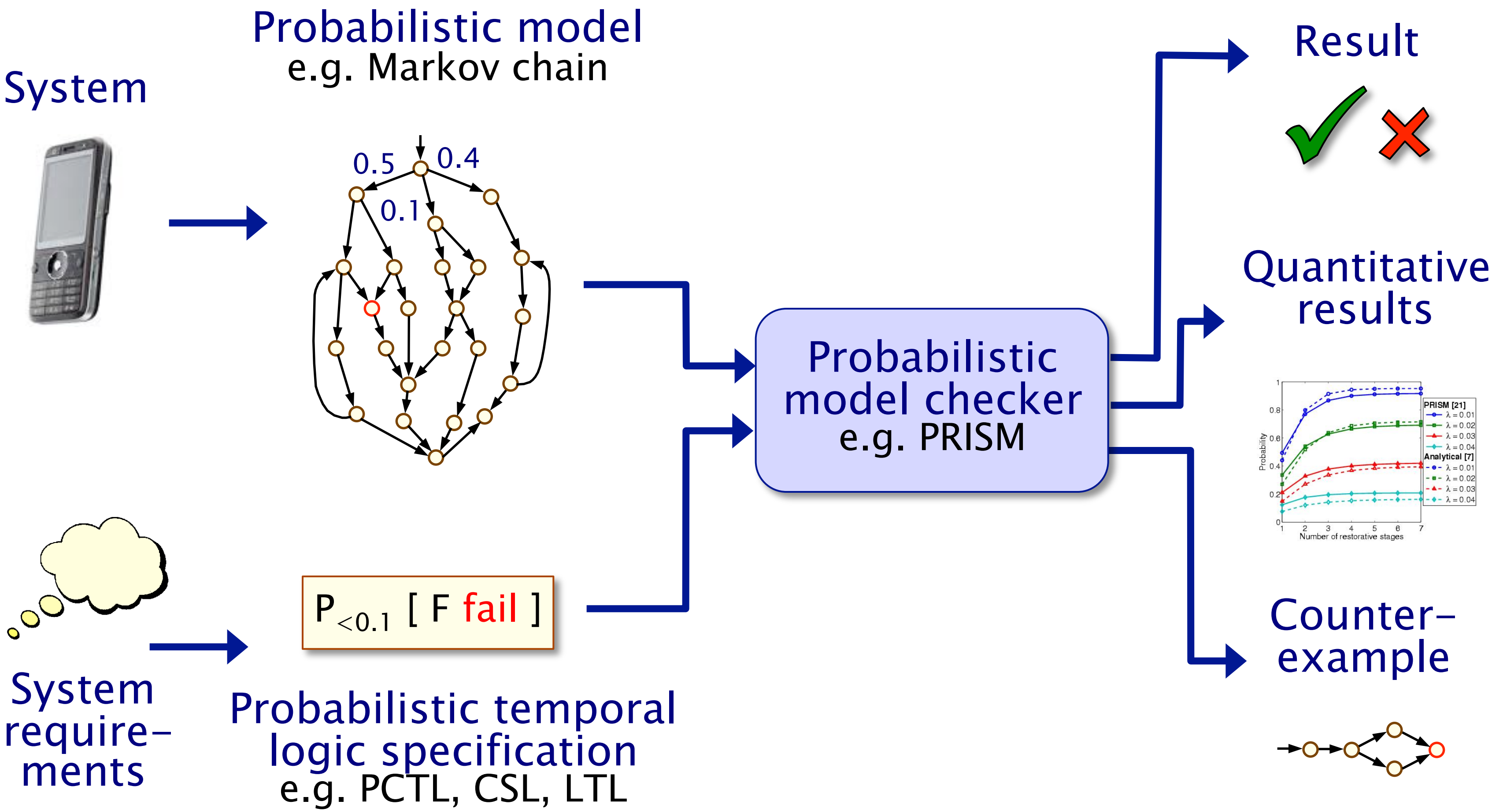
Why probability?

- Some systems are inherently probabilistic…
- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding
- Modelling uncertainty and performance
  - to quantify rate of failures, express Quality of Service
- Examples:
  - computer networks, embedded systems
  - power management policies
  - nano-scale circuitry: reliability through defect-tolerance

Why probability?

- Some systems are inherently probabilistic…
- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding
- Modelling uncertainty and performance
  - to quantify rate of failures, express Quality of Service
- For quantitative analysis of software and systems
  - to quantify resource usage given a policy
    "the minimum expected battery capacity for a scenario…"
- And many others, e.g. biological processes

Probabilistic model checking

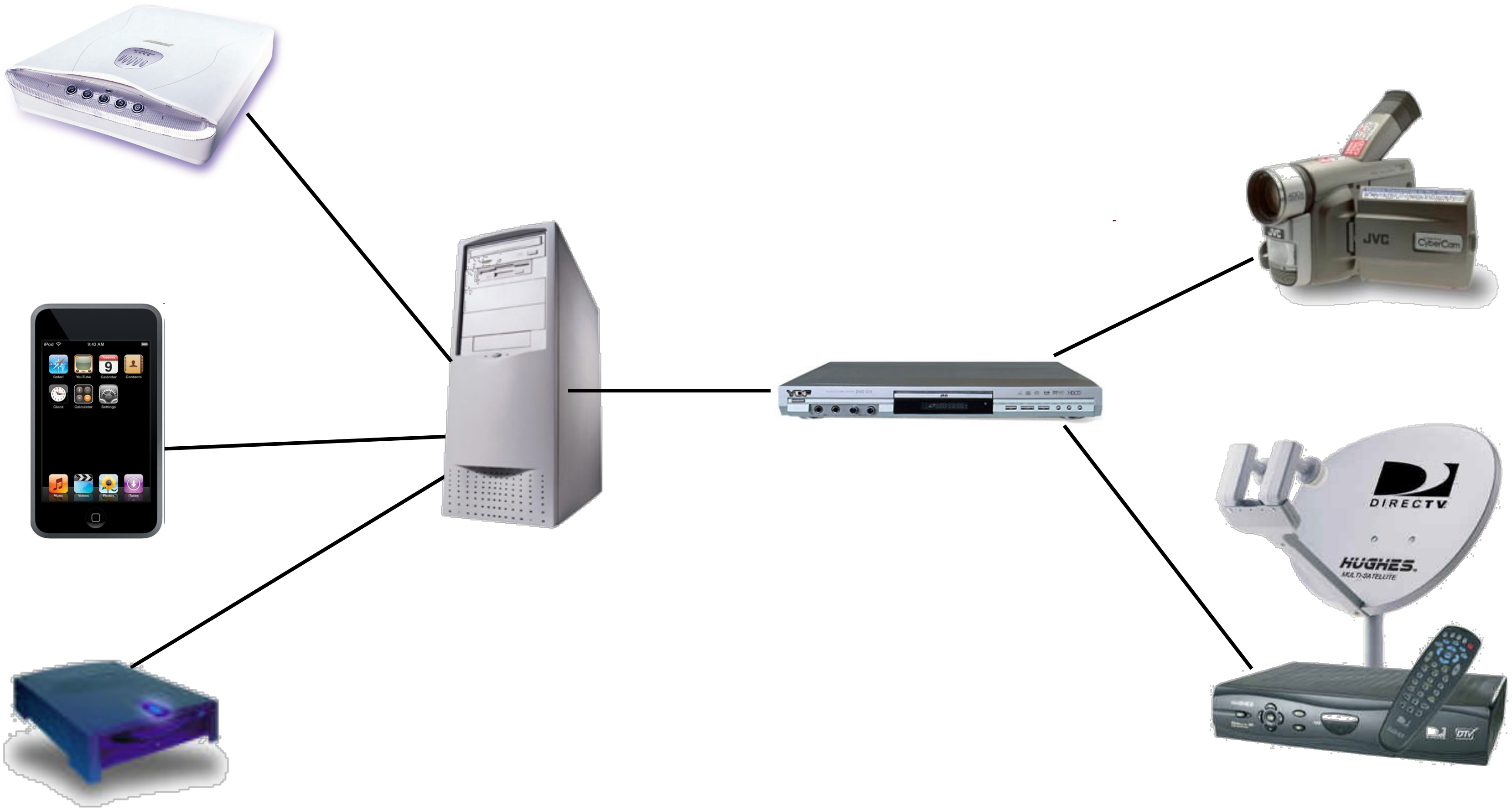


Case study: FireWire protocol

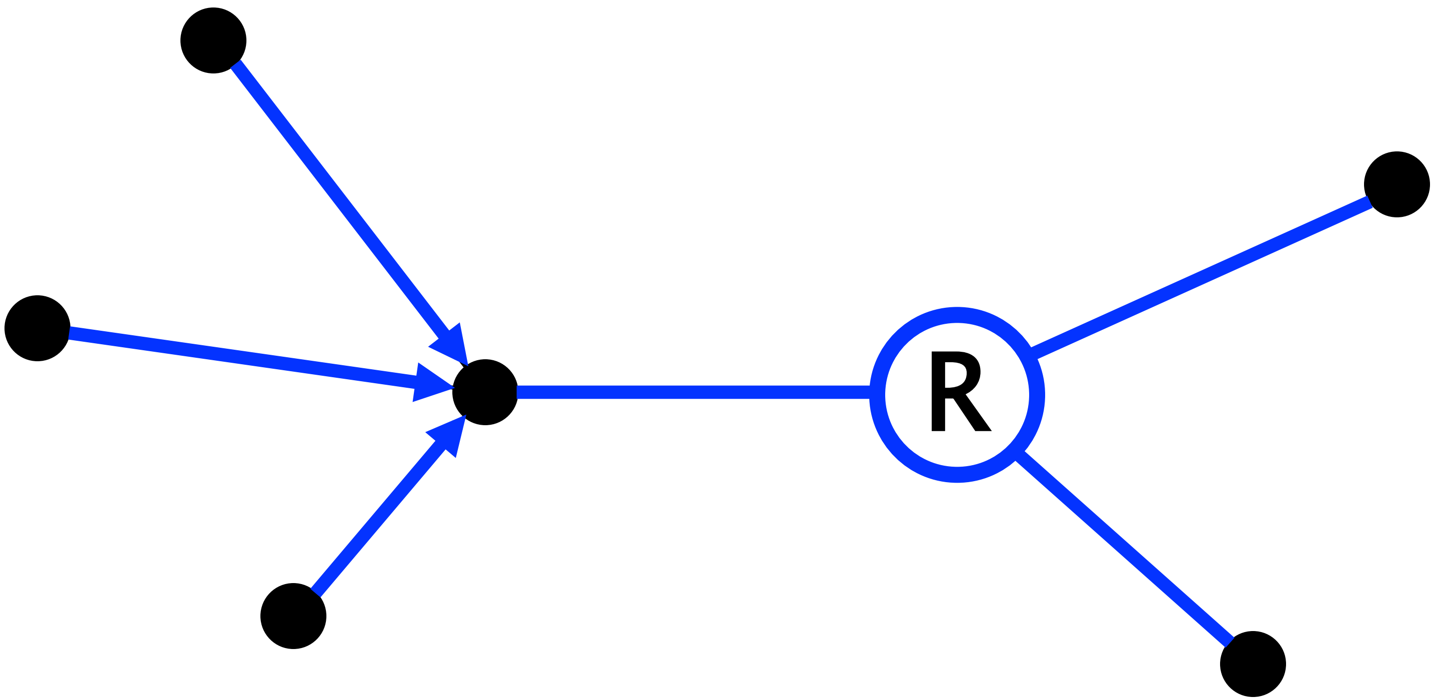
- FireWire (IEEE 1394)
  - high-performance serial bus for networking multimedia devices; originally by Apple
  - "hot-pluggable" - add/remove devices at any time
  - no requirement for a single PC (need acyclic topology)

- Root contention protocol
  - leader election algorithm, when nodes join/leave
  - symmetric, distributed protocol
  - uses electronic coin tossing and timing delays
  - nodes send messages: "be my parent"
  - root contention: when nodes contend leadership
  - random choice: "fast"/"slow" delay before retry

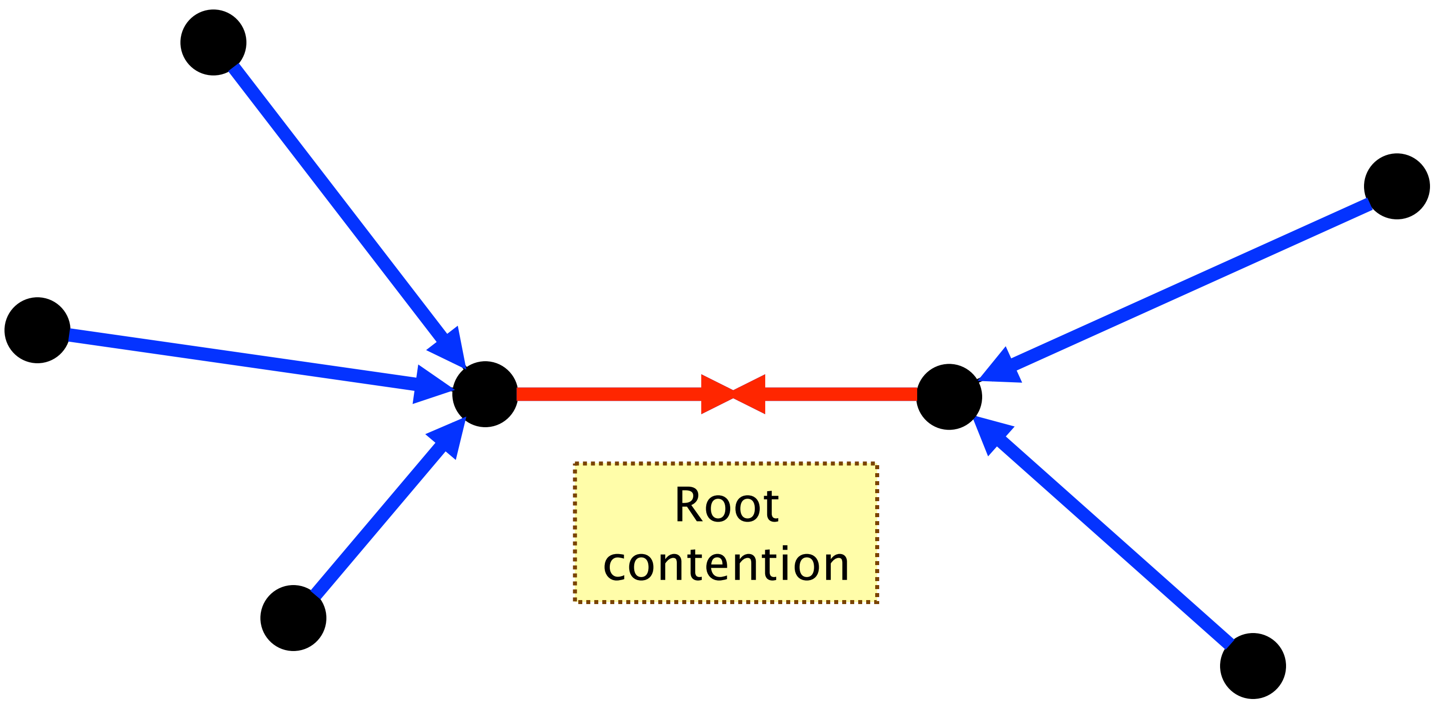
FireWire example



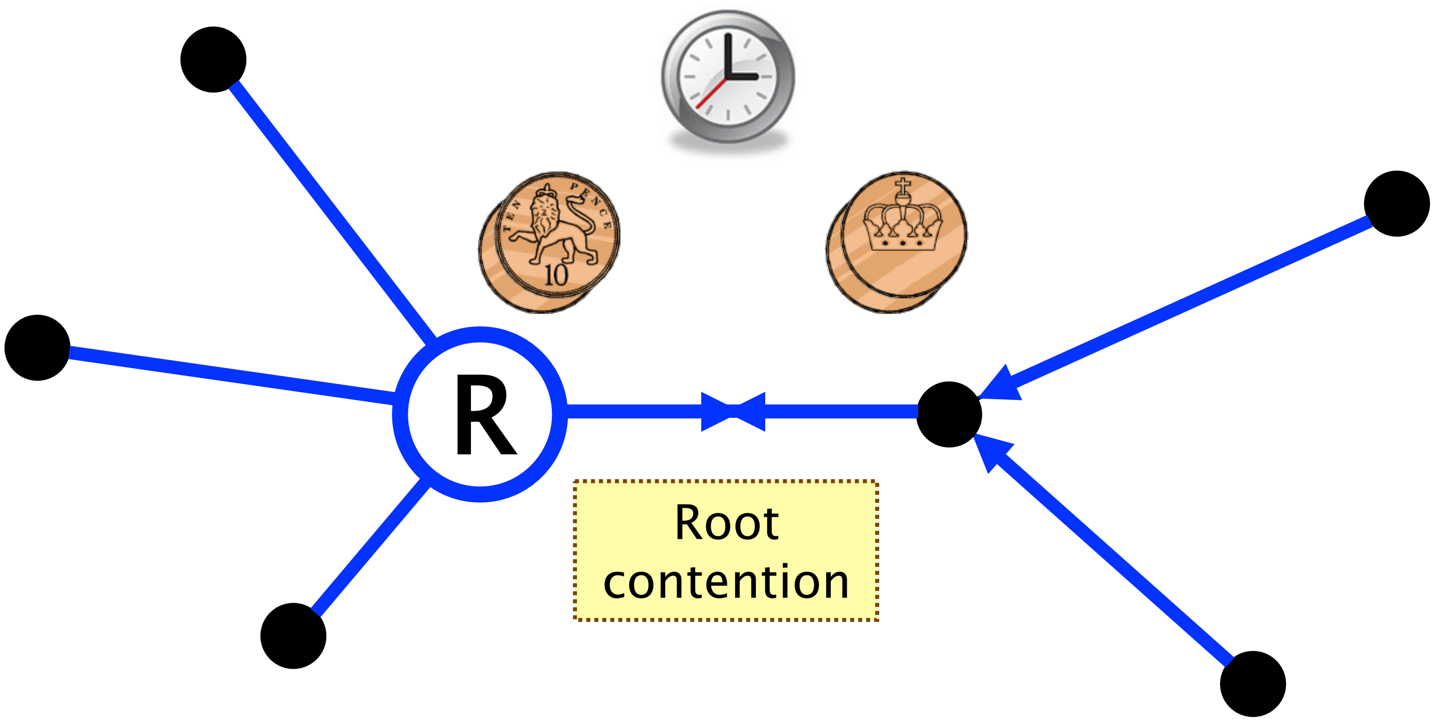
FireWire leader election



FireWire root contention

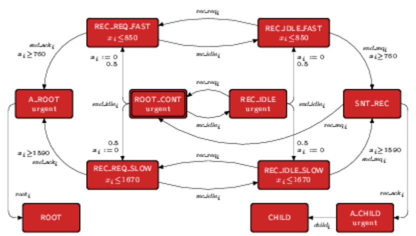


FireWire root contention

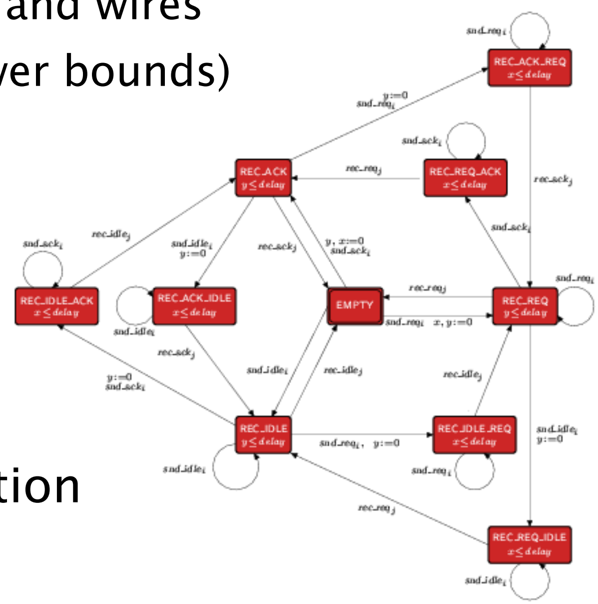


FireWire analysis

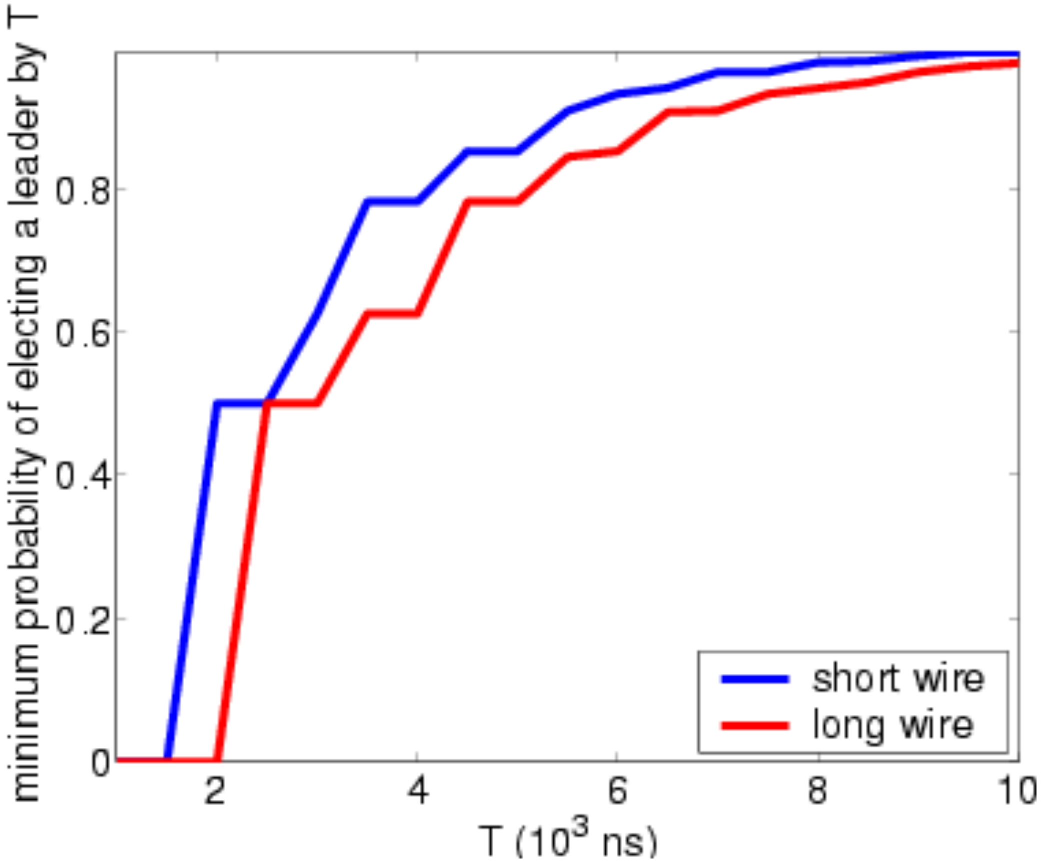
- Probabilistic model checking
  - model constructed and analysed using PRISM
  - timing delays taken from IEEE standard
  - model includes:
    - concurrency: messages between nodes and wires
    - underspecification of delays (upper/lower bounds)
  - max. model size: 170 million states



- Analysis:
  - verified that root contention always resolved with probability 1
  - investigated time taken for leader election
  - and the effect of using biased coin
    - based on a conjecture by Stoelinga

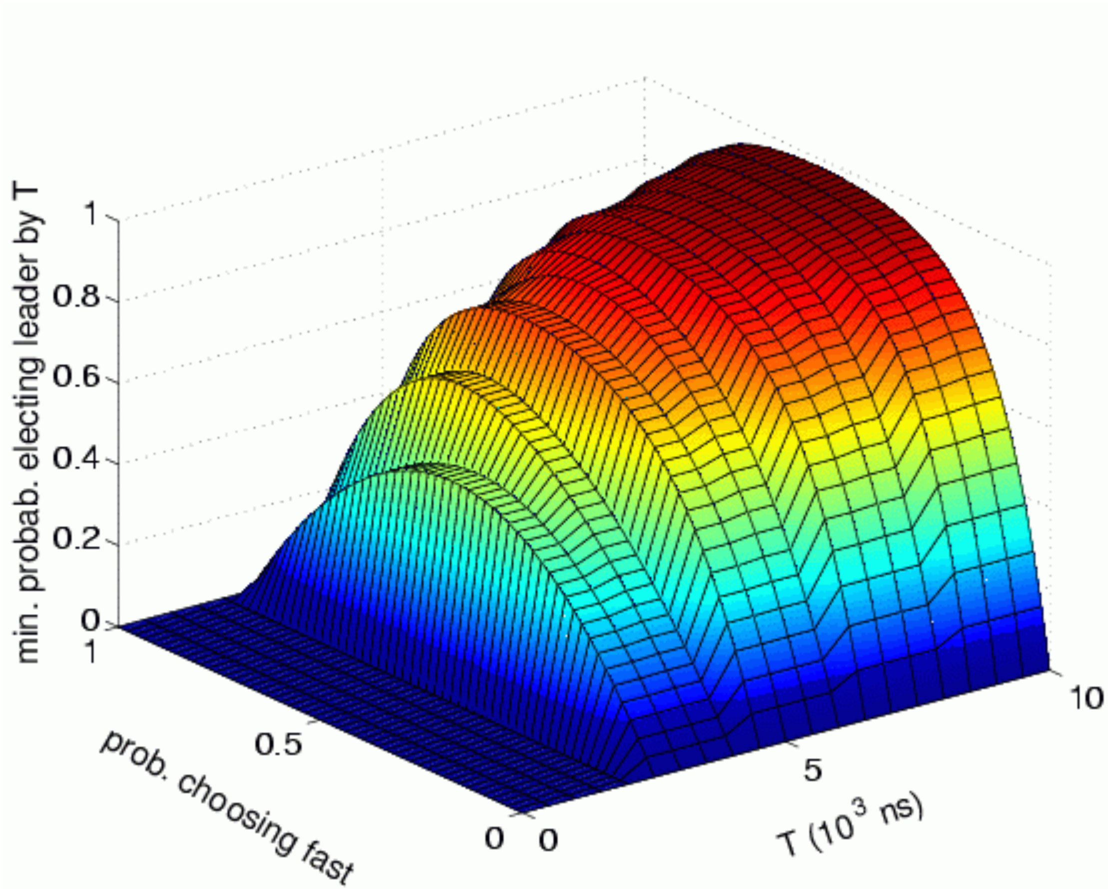


FireWire: Analysis results



"minimum probability of electing leader by time T"

FireWire: Analysis results

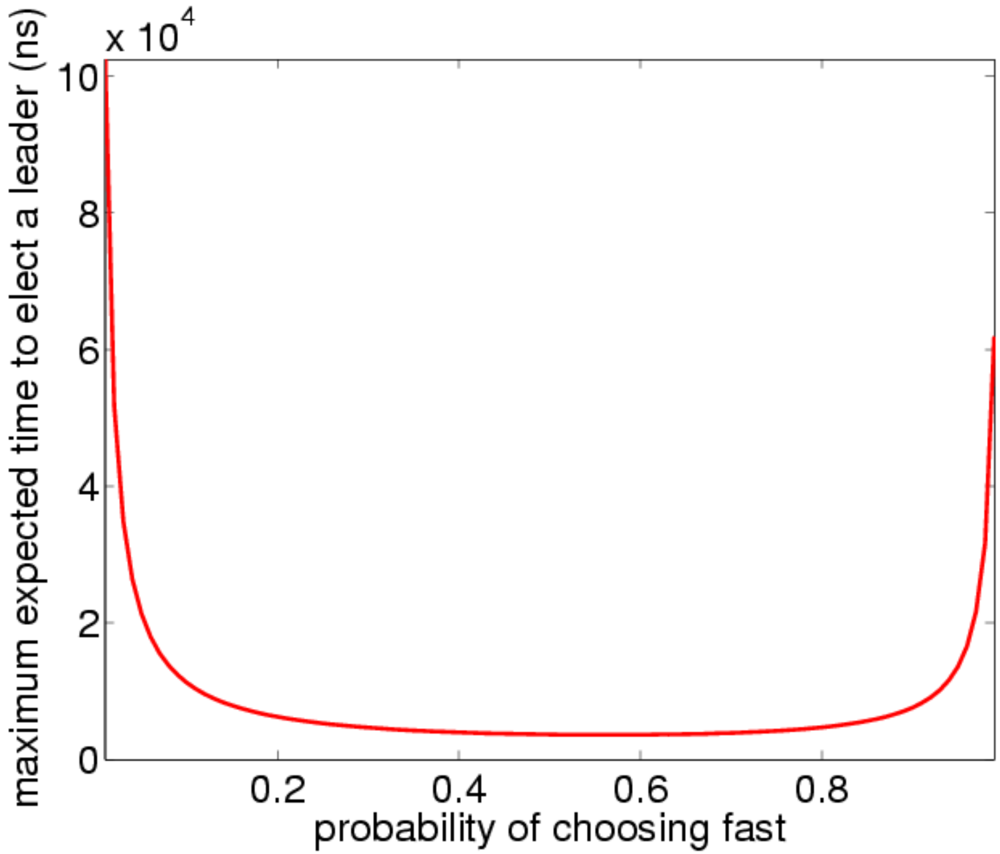


"minimum probability of electing leader by time T"

(short wire length)

Using a biased coin

FireWire: Analysis results

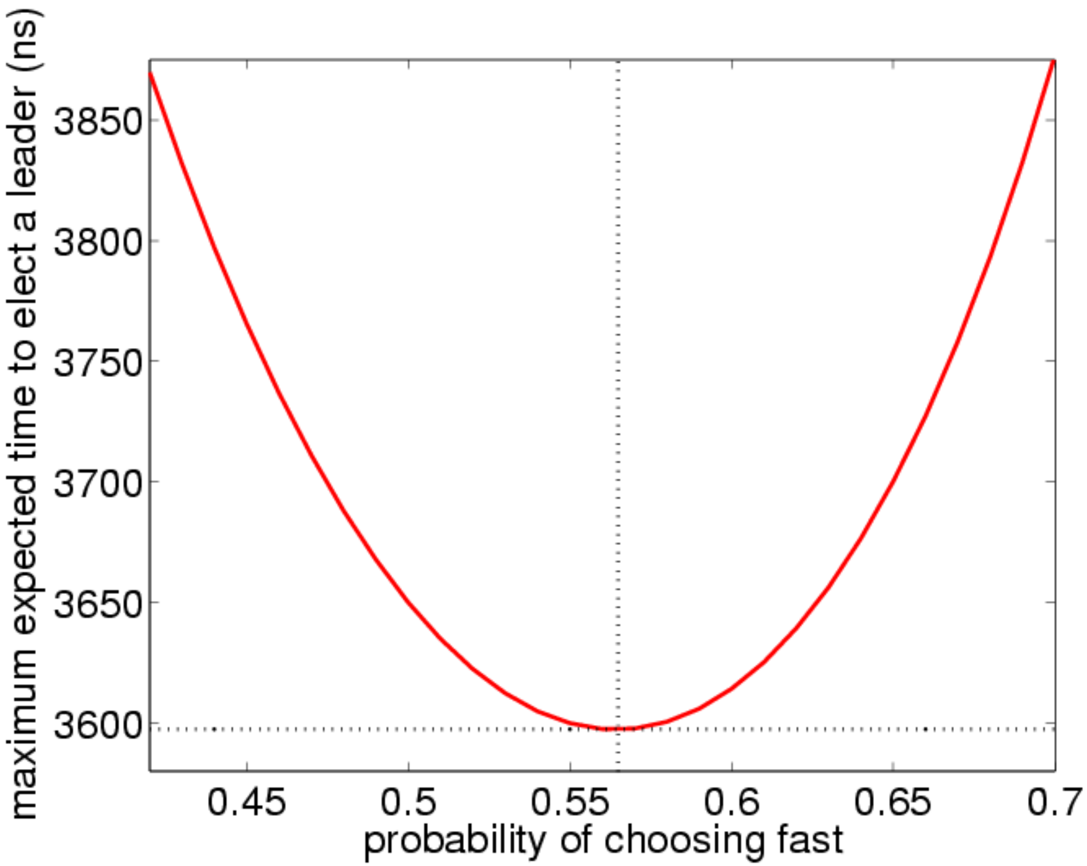


"maximum expected time to elect a leader"

(short wire length)

Using a biased coin

FireWire: Analysis results

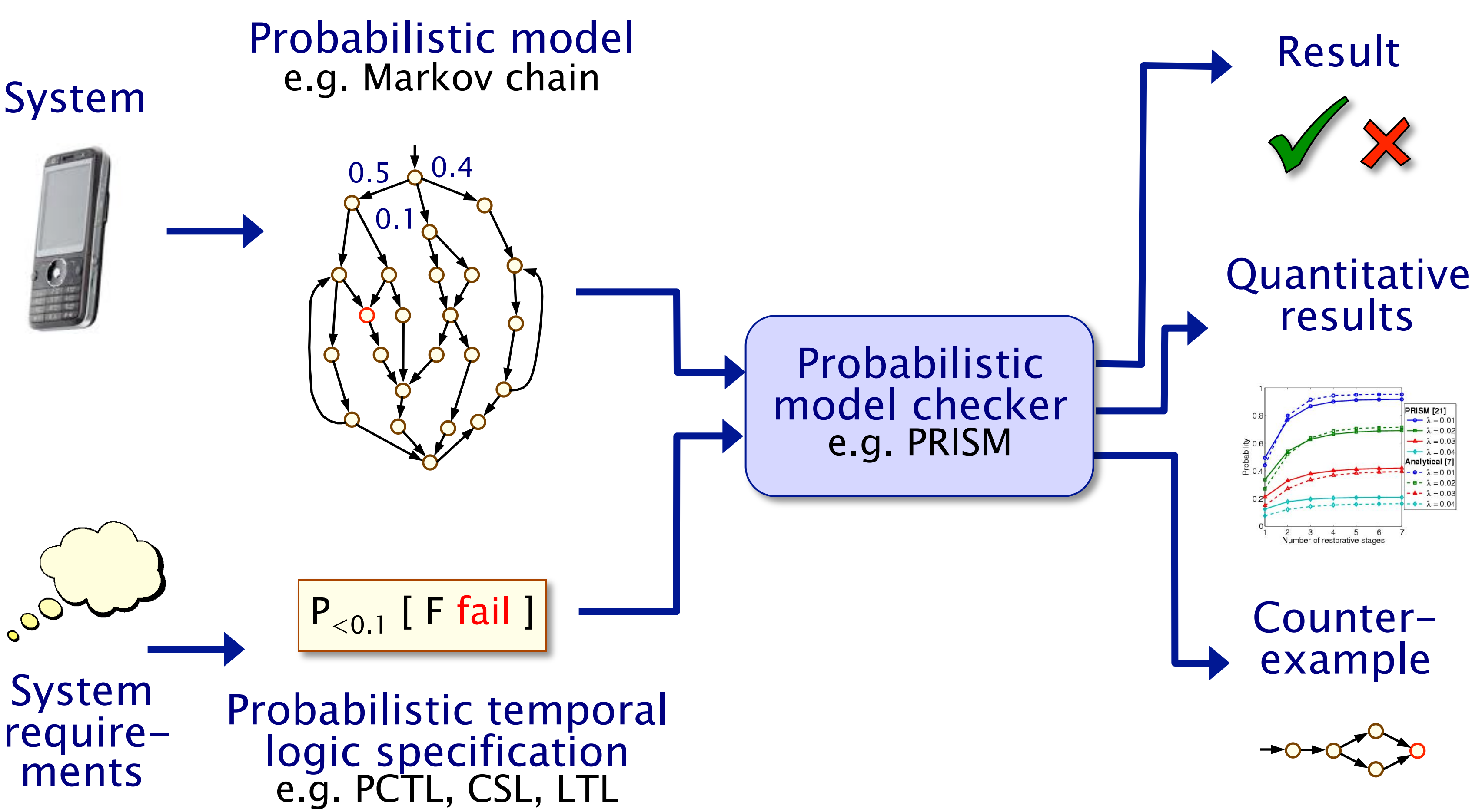


"maximum expected time to elect a leader"

(short wire length)

Using a biased coin is beneficial!

Probabilistic model checking



Probabilistic model checking inputs

- Models: variants of Markov chains
  - discrete-time Markov chains (DTMCs)
    - discrete time, discrete probabilistic behaviours only
  - continuous-time Markov chains (CTMCs)
    - continuous time, continuous probabilistic behaviours
  - Markov decision processes (MDPs)
    - DTMCs, plus nondeterminism
- Specifications
  - informally:
    - "probability of delivery within time deadline is …"
    - "expected time until message delivery is …"
    - "expected power consumption is …"
  - formally:
    - probabilistic temporal logics (PCTL, CSL, LTL, PCTL*, …)
    - e.g. $P_{<0.05}$ [F err/total>0.1], $P_{=?}$ [$F^{\leq t}$ reply_count=k]

Probabilistic model checking involves…

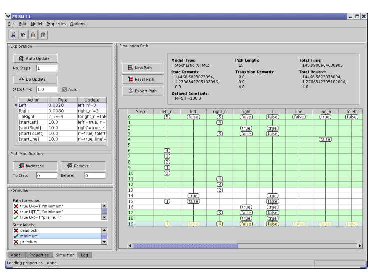
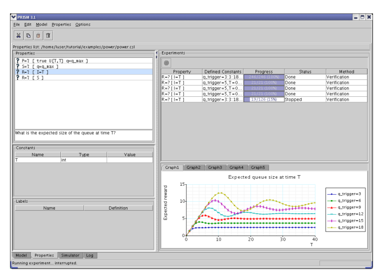
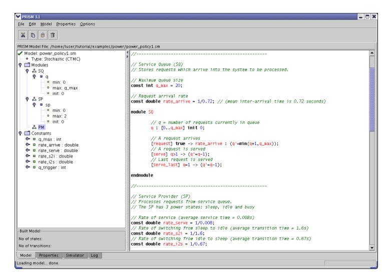
- Construction of models
  - from a description in a high-level modelling language

- Probabilistic model checking algorithms
  - graph-theoretical algorithms
    - e.g. for reachability, identifying strongly connected components
  - numerical computation
    - linear equation systems, linear optimisation problems
    - iterative methods, direct methods
    - uniformisation, shortest path problems
  - automata for regular languages
  - also sampling-based (statistical) for approximate analysis
    - e.g. hypothesis testing based on simulation runs

Probabilistic model checking involves…

- Efficient implementation techniques
  - essential for scalability to real-life systems
  - symbolic data structures based on binary decision diagrams
  - algorithms for bisimulation minimisation, symmetry reduction

- Tool support
  - PRISM: free, open-source probabilistic model checker
  - currently based at Oxford University
  - supports all probabilistic models discussed here

Course aims

- Introduce main types of probabilistic models and specification notations
  - theory, syntax, semantics, examples
  - probability, expectation, costs/rewards
- Explain the working of probabilistic model checking
  - algorithms & (symbolic) implementation
- Introduce software tools
  - probabilistic model checker PRISM
- Examples from wide range of application domains
  - communication & coordination protocols, performance & reliability modelling, biological systems, …

- Mix of theory and practice

Course outline

- Discrete-time Markov chains (DTMCs) and their properties
- Probabilistic temporal logics: PCTL, LTL, etc.
- PCTL model checking for DTMCs
- The PRISM model checker
- Costs & rewards
- Continuous-time Markov chains (CTMCs)
- Counterexamples & bisimulation
- Markov decision processes (MDPs)
- Probabilistic LTL model checking
- Implementation and data structures: symbolic techniques

Course information

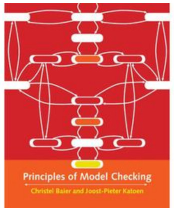
- Prerequisites/background
  - basic computer science/maths background
  - no probability knowledge assumed

- Lectures
  - 20 lectures: Mon 2pm, Wed 3pm, Thur 12pm (wks 1-4)
- Classes/practicals (please sign up on-line)
  - 4 problem sheets + 1 hr classes
    (Tue 3pm, Wed 12pm, wks 3, 5, 7, 8)
  - 4 practical exercises, based on PRISM,
    4 scheduled 2 hr practical sessions (Tue 4pm, wks 3, 4, 6, 7),
    + work outside lab sessions
- Assessment
  - take-home assignment

Further information

- Course lecture notes are self-contained
  - www.cs.ox.ac.uk/teaching/materials11-12/probabilistic/

- For further reading material…
  - two online tutorial papers also cover a lot of the material
    - Stochastic Model Checking
      Marta Kwiatkowska, Gethin Norman and David Parker
    - Automated Verification Techniques for Probabilistic Systems
      Vojtěch Forejt, Marta Kwiatkowska, Gethin Norman, David Parker
  - DTMC/MDP material also based on Chapter 10 of:



Principles of Model Checking
Christel Baier and Joost-Pieter Katoen
MIT Press

  - PRISM web site: http://www.prismmodelchecker.org/

Next lecture(s)

- Wed 3pm
- Thur 12pm

- Discrete-time Markov chains

Acknowledgements

- Much of the material in the course is based on an existing lecture course prepared by:
  - Marta Kwiatkowska
  - Gethin Norman
  - Dave Parker

- Various material and examples also appear courtesy of:
  - Christel Baier
  - Joost-Pieter Katoen