Probabilistic Systems

Introduction

Dr. Gethin Norman

School of Computing Science
University of Glasgow

gethin.norman@glasgow.ac.uk

Why probability?

Some systems are inherently probabilistic…

Randomisation, e.g. in distributed coordination algorithms

- as a symmetry breaker in leader election protocols

Examples: real-world protocols featuring randomisation

- IEEE 802.3 CSMA/CD, IEEE 802.11 Wireless LAN (WiFi)
  - use randomised back-off schemes
- IEEE 1394 Firewire (root contention), Bluetooth (device discovery)
  - have a random choice of waiting time
- IPv4 Zeroconf dynamic configuration (link-local addressing)
  - makes a random choice over a set of possible addresses
- 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 leader election protocols

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 leader election protocols

Modelling uncertainty and performance

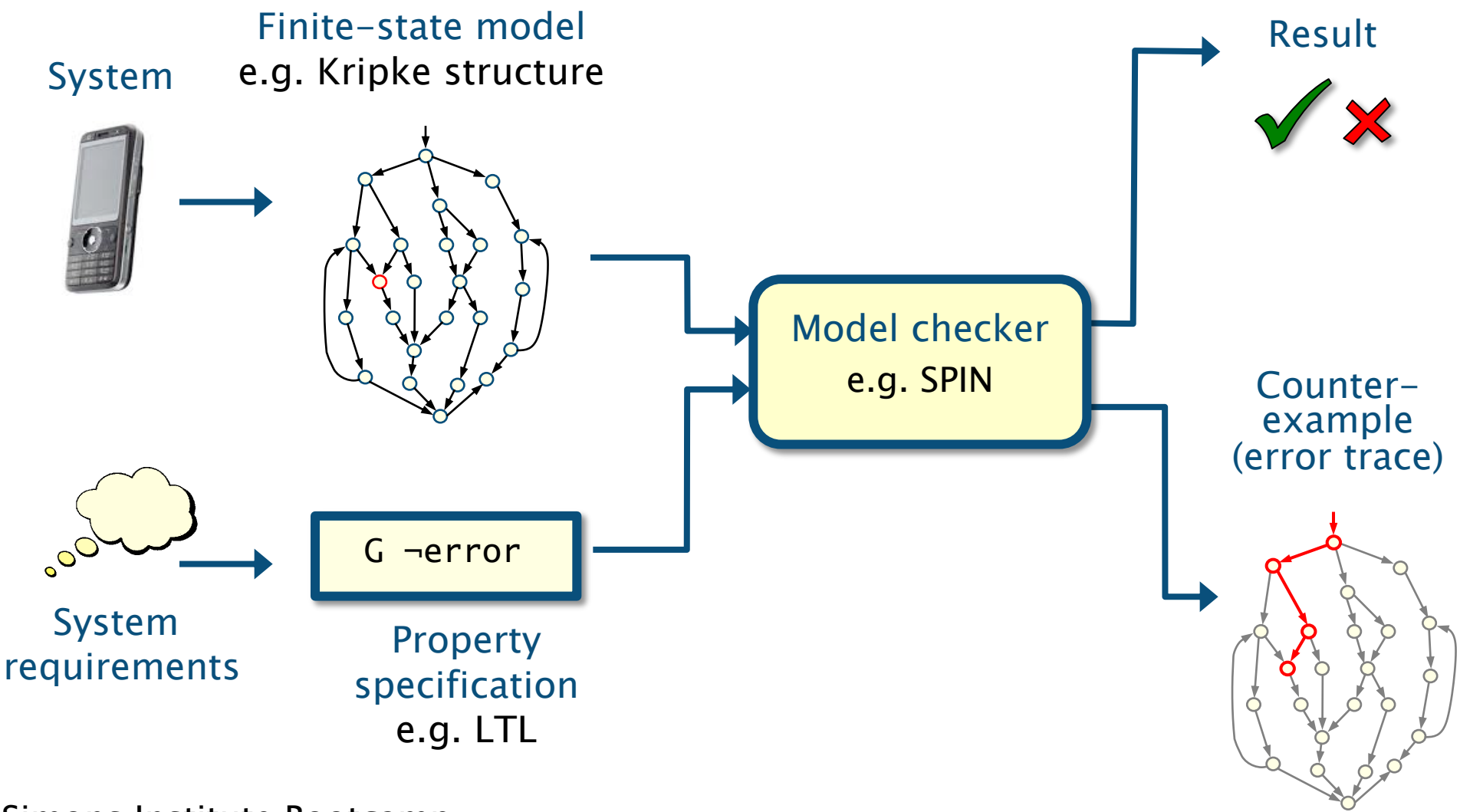
- to quantify rate of failures, express Quality of Service

For quantitative analysis of software and systems

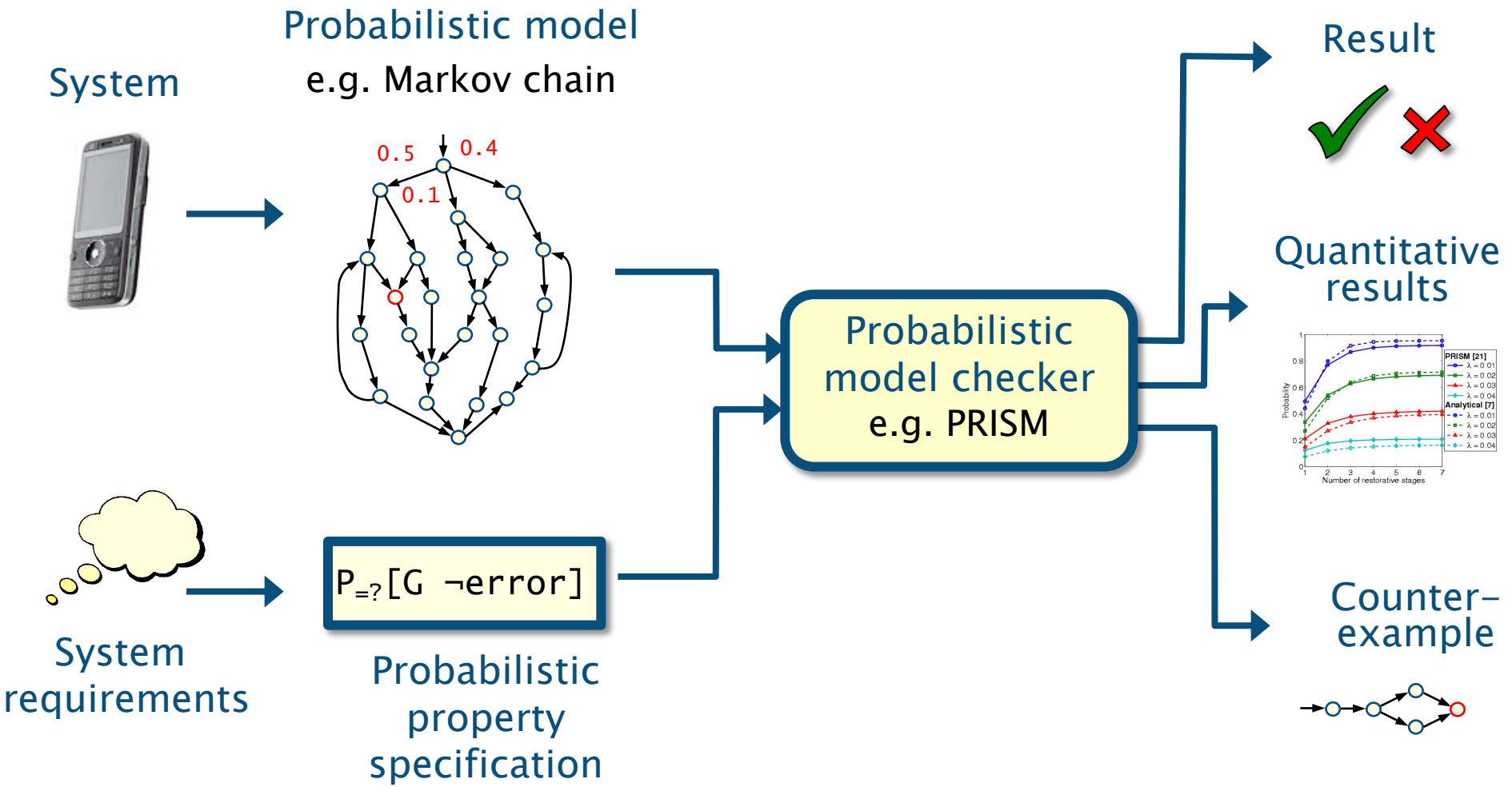
- to quantify resource usage given a policy
  "the minimum battery capacity for a given scenario is .."

And many others, e.g. biological systems

Model checking



Probabilistic model checking



Case study: FireWire protocol

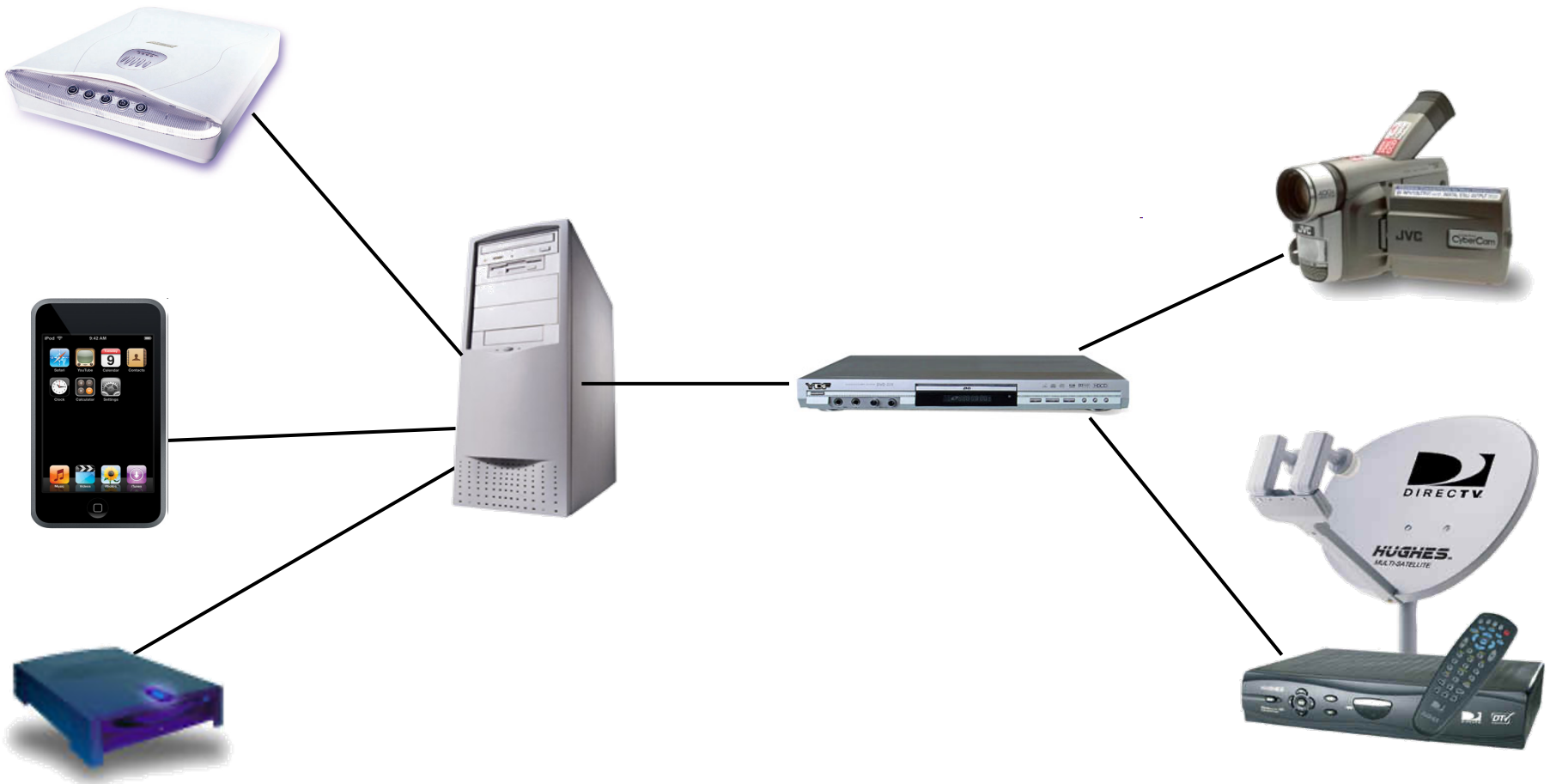
FireWire (IEEE 1394)

- 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

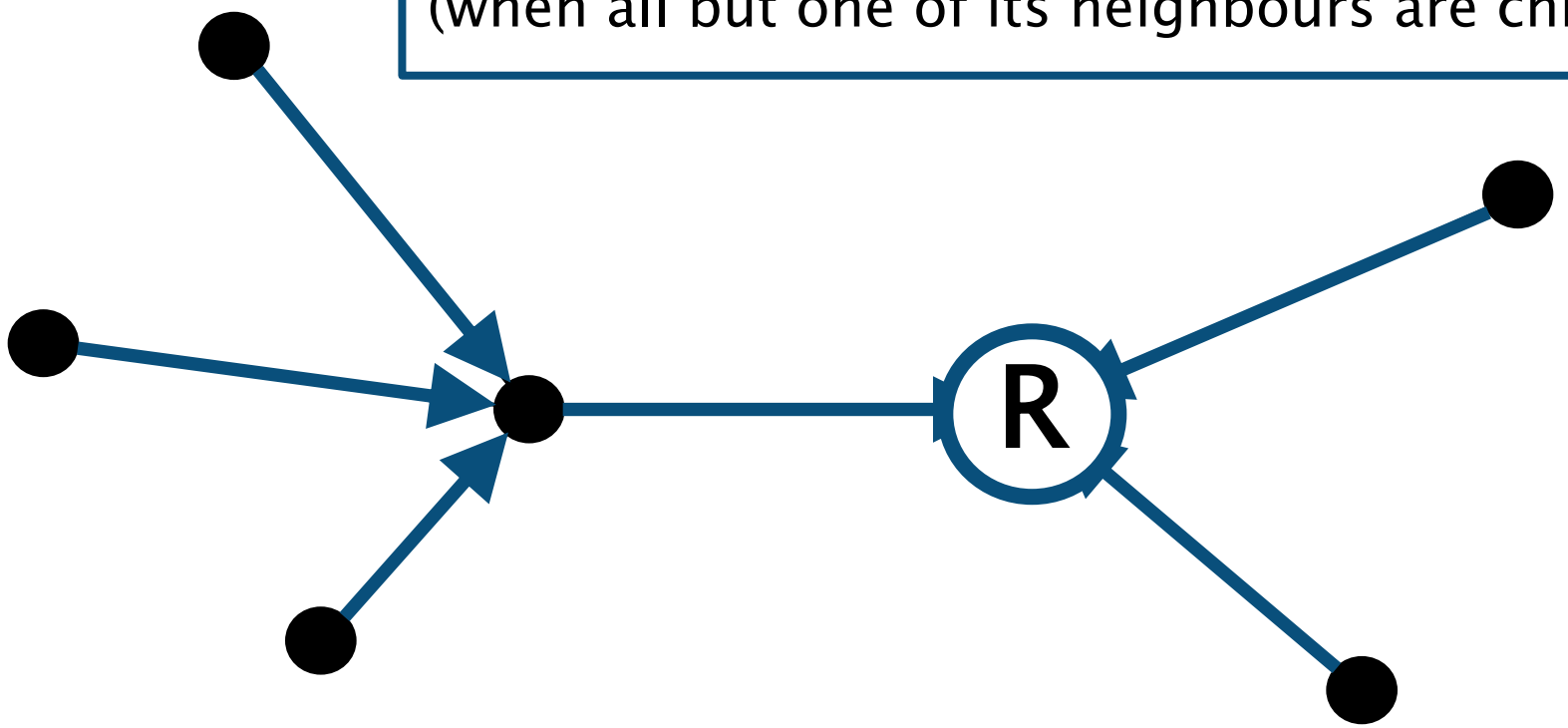
- leader election algorithm, when nodes join/leave
- symmetric, distributed protocol
- uses electronic coin tossing and timing delays
- nodes send messages: "be my parent"
- root contention: when nodes contend leadership
- random choice: "fast"/"slow" delay before retry

FireWire example

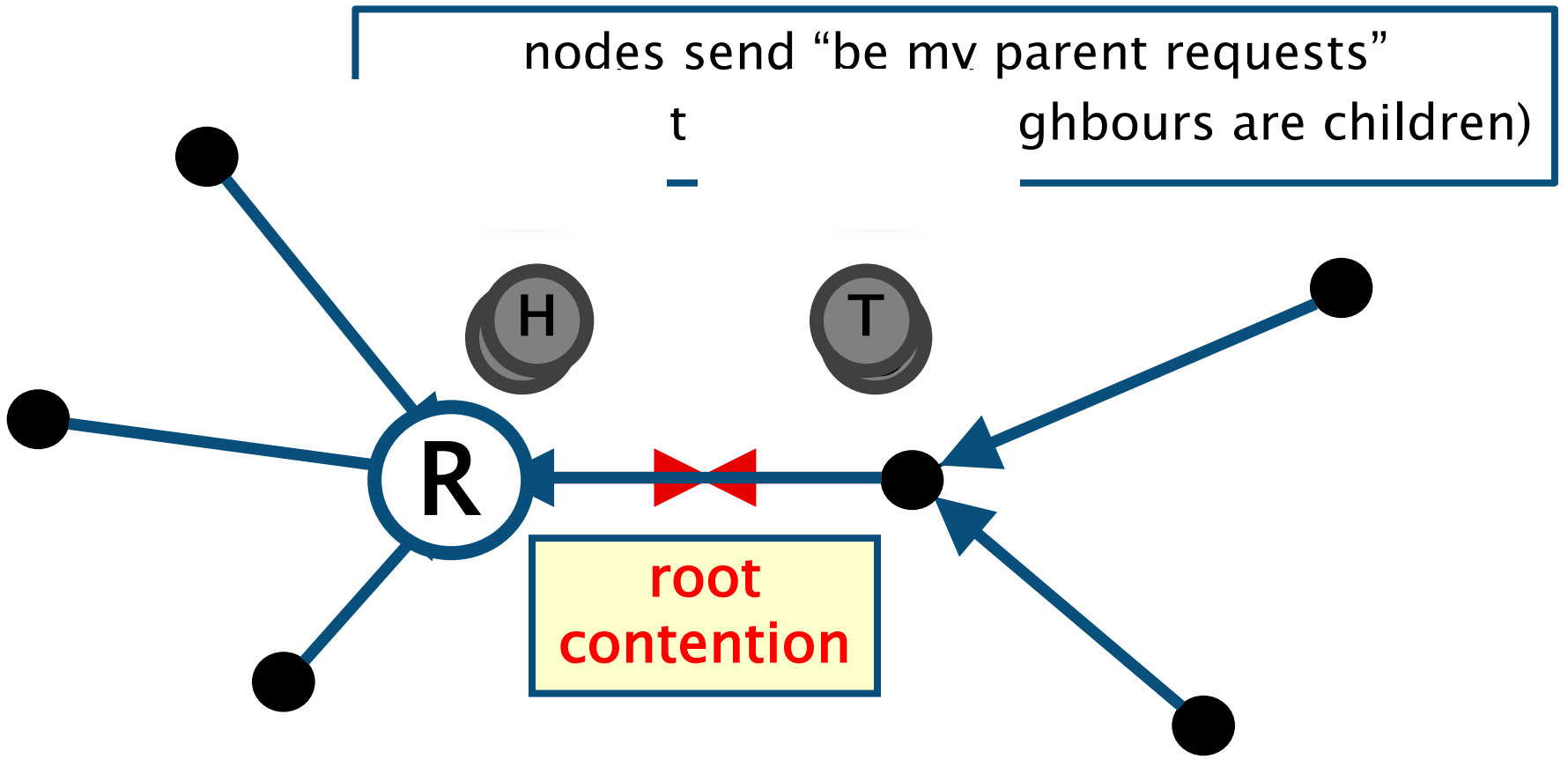


FireWire leader election

nodes send “be my parent requests”
(when all but one of its neighbours are children)



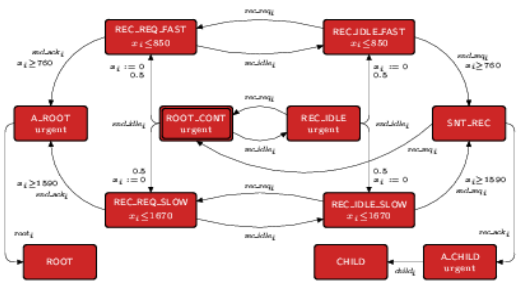
FireWire root contention



FireWire analysis

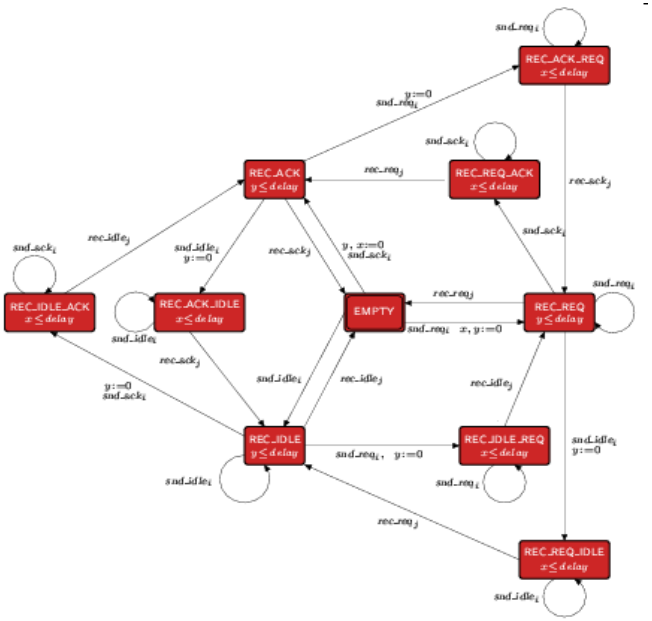
Probabilistic model checking using PRISM

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

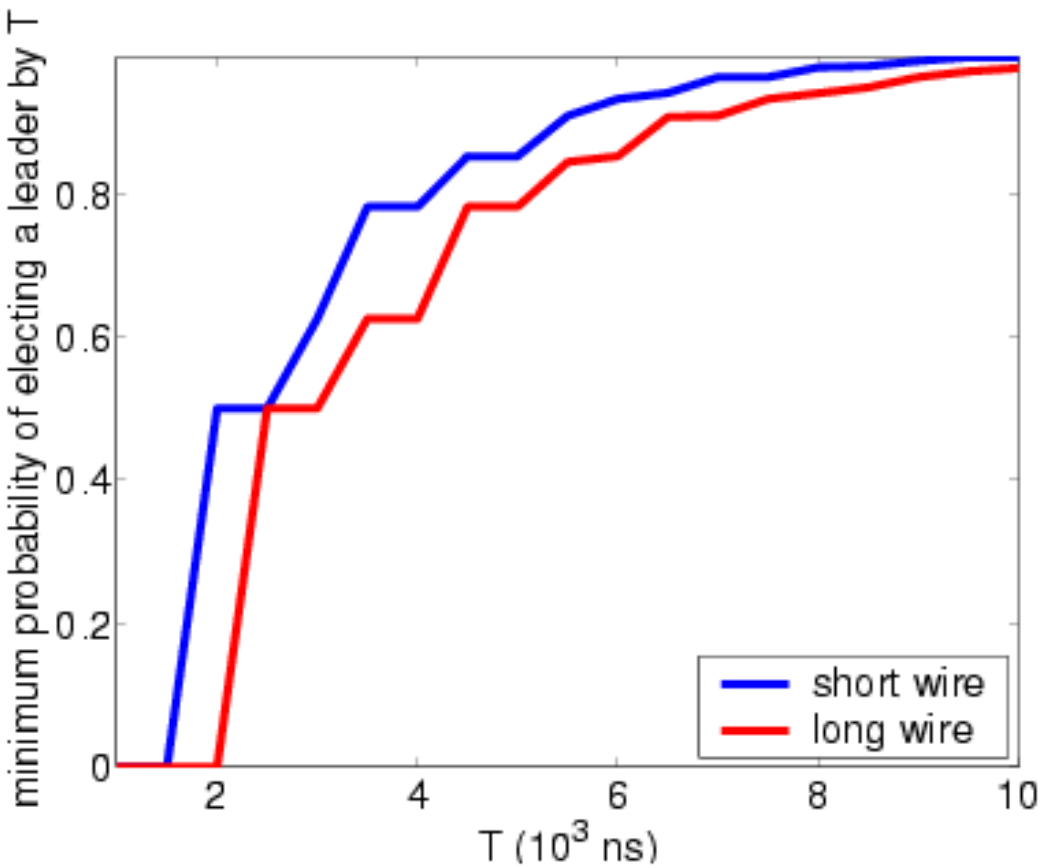


Analysis:

- verified that root contention always resolved with probability 1
- investigated time taken for leader election and the effect of using biased coin

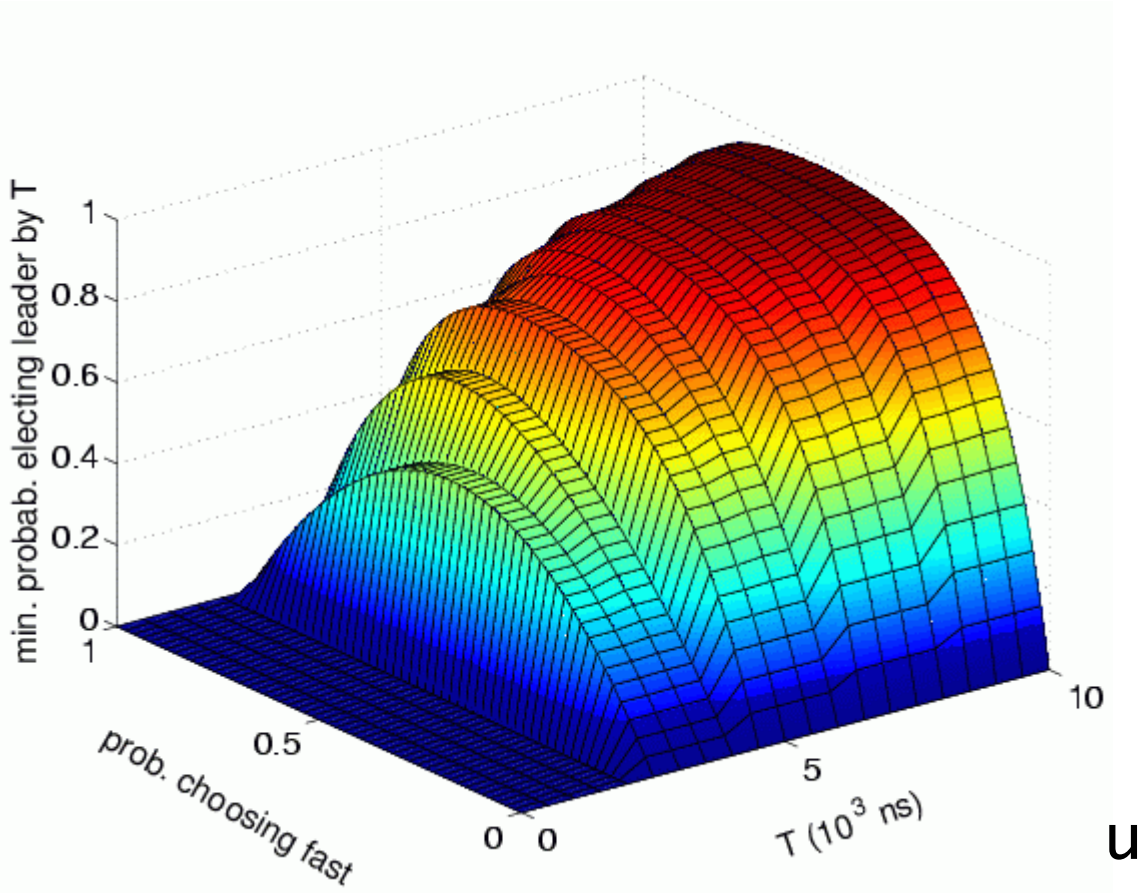


FireWire: Analysis results



"minimum probability of electing leader by time T"

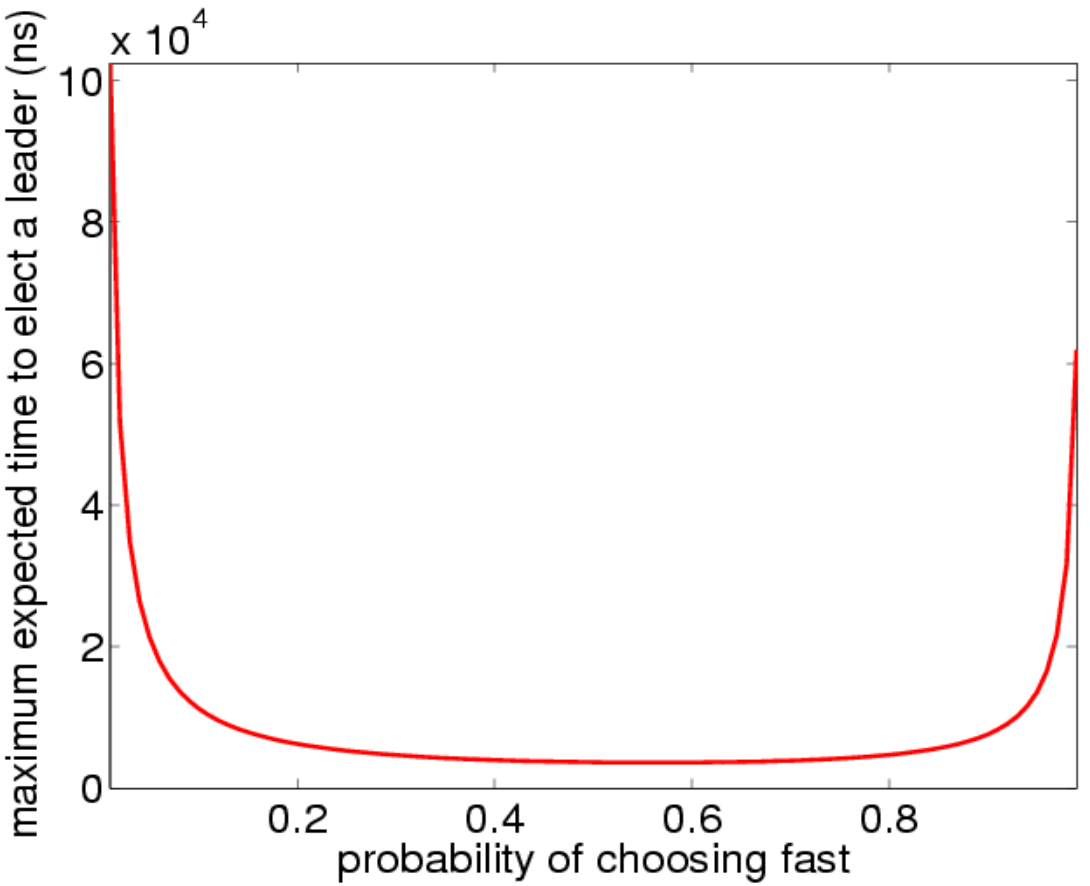
FireWire: Analysis results



"minimum probability of electing leader by time T"

using a biased coin (short wire)

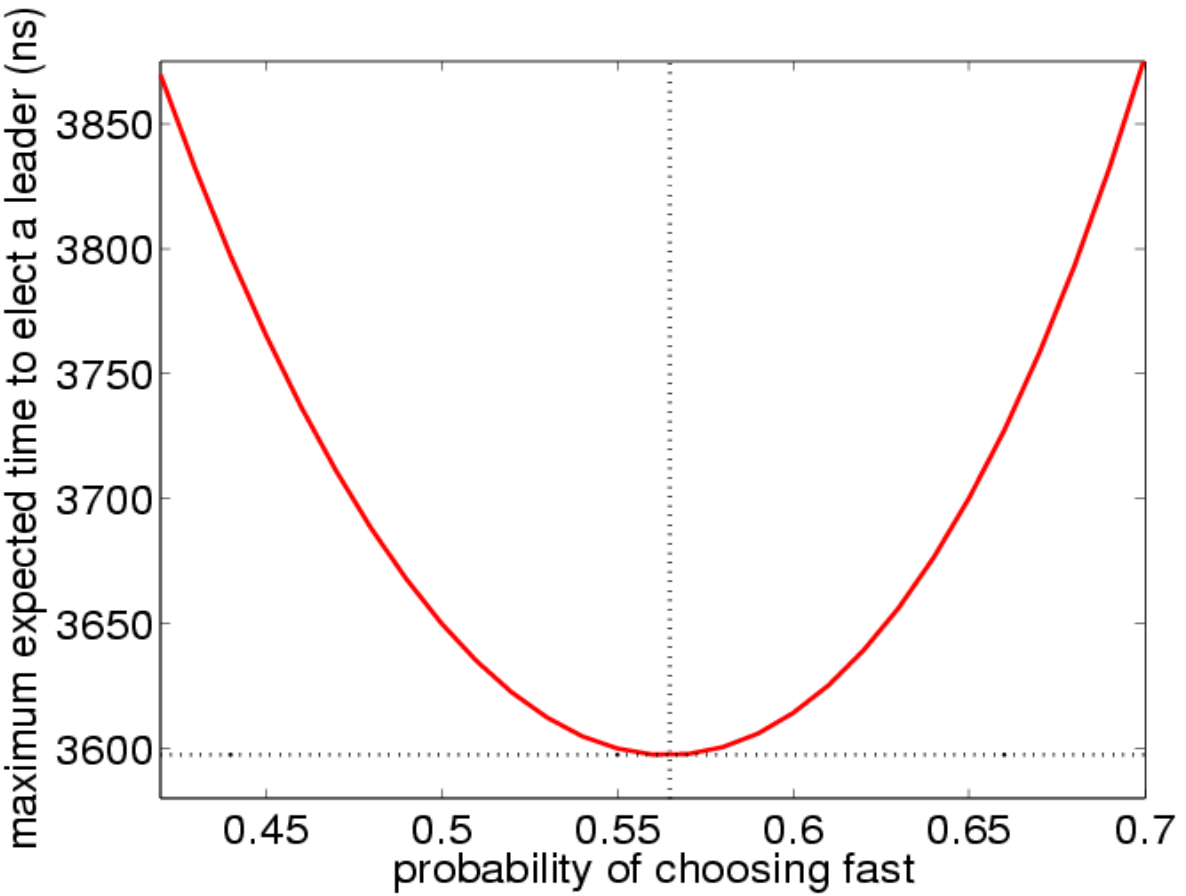
FireWire: Analysis results



"maximum expected time to elect a leader"

using a biased coin

FireWire: Analysis results



"maximum expected time to elect a leader"

using a biased coin is beneficial

Probabilistic model checking inputs

Discrete-time models: variants of Markov chains

- discrete-time Markov chains
- Markov decision processes
- turn-based stochastic games
- concurrent stochastic games
- partially observable Markov decision processes and games

Continuous and real-time models

- continuous time Markov chains
- probabilistic timed automata
  - plus game-based and partially observable variants
- stochastic hybrid automata

Probabilistic model checking inputs

Game-based models

- allow the modelling of collaborative and competitive behaviour between between agents, possibly with differing or opposing goals
- e.g. security (system vs. attacker),
- e.g. controller synthesis (controller vs. environment)

Partial observability

- resolve actions based on observations only
- e.g. a robot can only make decisions based on sensors
- e.g. a scheduler cannot probe state of a component

Probabilistic model checking inputs

Specifications informally:

- "probability of delivery within time deadline is …"
- "expected time until message delivery is …"
- "expected power consumption is …"

Specifications formally:

- probabilistic temporal logics: PCTL, LTL, CSL, RPATL …

Will focus on probabilistic and expected reachability

- these are fundamental properties

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 and qualitative properties (with probability 0 or 1)
- numerical computation
  - linear equation systems and 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

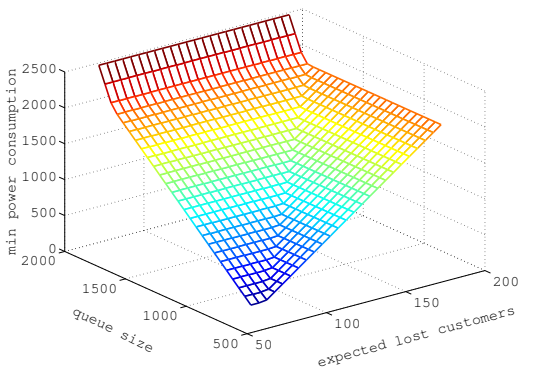
Extensions – Strategy/controller synthesis

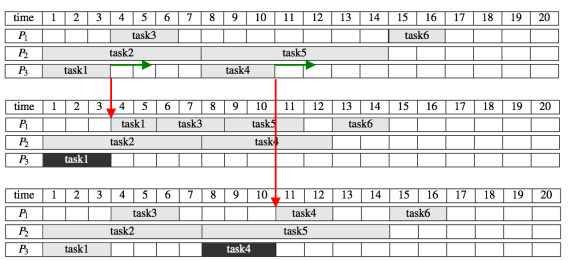
Verification vs. control

- verify that a system is “correct” for any environment/adversary/…
  - counterexample yields flaw/attack/…
- synthesise a "correct-by-construction” controller from formal specification
  - witness yields strategy/controller

Applications

- dynamic power management,
  robots/autonomous vehicle navigation,
  task/network scheduling,
  security, …





Task schedule

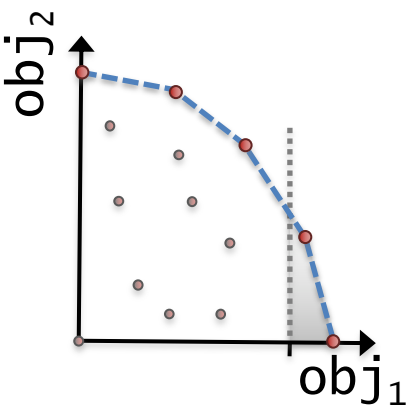
Extensions – Multiple objectives

Multi-objective controller synthesis

- trade-offs between conflicting objectives
  - e.g. cost vs. quality of service

Mix of optimisation and guarantees

- e.g. "what strategy maximises probability of message transmission, whilst guaranteeing expected battery life-time is >10 hrs?"
- Pareto curve generation/approximation



Extensions – Parameter synthesis

Synthesising models that are guaranteed to satisfy quantitative correctness properties is difficult

- but we can synthesise controllers and parameters

Parameter synthesis

- given a parametric model and a property ϕ...
- find the optimal parameter values, with respect to an objective function `obj`, such that the property ϕ is satisfied, if such values exist

Quantitative parameter synthesis

- parameters: timing delays, rates
- objectives: optimise probability, reward

Probabilistic model checking in practice

PRISM: Probabilistic symbolic model checker

- developed at Birmingham/Oxford University, since 1999
- free, open source (GPL)
- versions for Linux, Mac OS X and Windows

Modelling and verification of:

- DTMCs, CTMCs, MDPs, POMDPs, probabilistic timed automata (PTAs)
- PRISM-games extension (www.prismmodelchecker.org/games/)

PRISM website: www.prismmodelchecker.org/

- tool download: binaries, source code (GPL)
- on-line example repository (50+ case studies)
- on-line documentation: PRISM manual and tutorial
- support: help forum, bug tracking, feature requests
- related publications, talks, tutorials, links

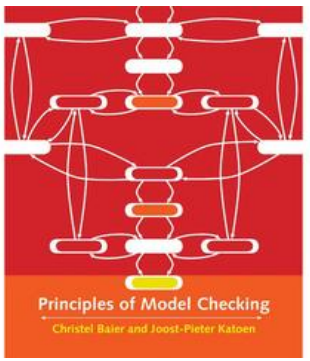
Acknowledgements

Material in parts 1 and 2 is based on existing lecture courses prepared by:

- Dave Parker, Marta Kwiatkowska and Gethin Norman

Various material and examples also appear courtesy of:

- Christel Baier and Joost-Pieter Katoen



Course outline

Part 1: Discrete-time Markov chains (DTMCs)

- paths and probabilities for DTMCs
- probabilistic reachability
- reward structures
- expected reachability

Part 2: Markov Decision Processes (MDPs)

- paths, strategies and probabilities for MDPs
- probabilistic reachability for MDPs
  - qualitative probabilistic reachability
  - optimality equations
  - computing reachability probabilities

parts 1 and 2 are present the basics and material suitable to newcomers for entering the field

Live lecture (advanced topic): Partially observable probabilistic systems