Probabilistic Model Checking

Marta Kwiatkowska
Gethin Norman
Dave Parker

University of Oxford

Part 1 – Introduction

Overview

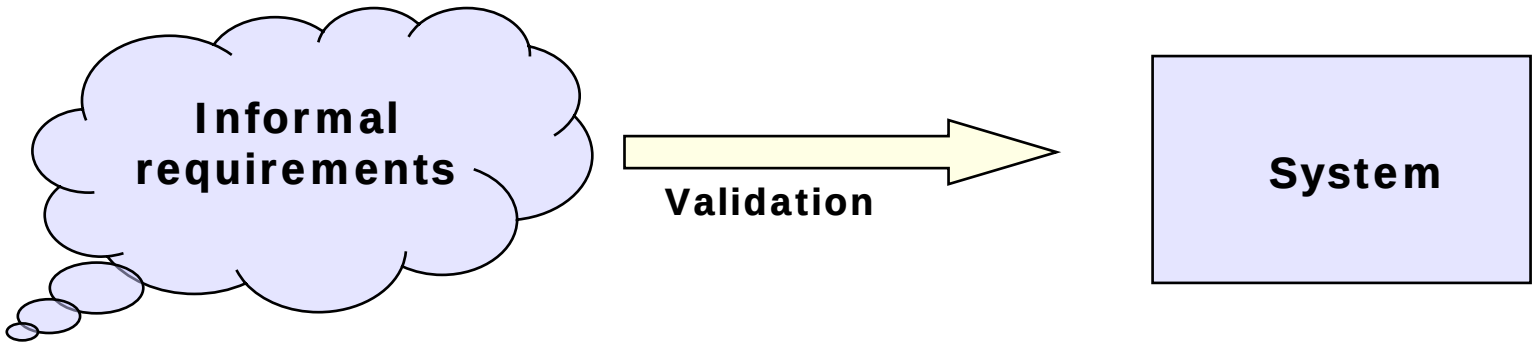
- Introducing probabilistic model checking…
- Topics for this lecture
  - modern software engineering
  - the role of automatic verification
  - what is probabilistic model checking?
  - why is it important?
  - where is it applicable?
- About this lecture course
  - aims, organisation
  - the PRISM tool
  - further reading, information and links

Ubiquitous computing – The trends…

- Devices, ever smaller
  - Laptops, phones, PDAs, sensors…
- Networking, wireless, wired & global
  - Wireless & Internet everywhere
- Systems/software
  - Self-*
  - Mobile
  - Adaptive
  - Context-aware
- How to design & engineer
  - Adaptive systems and networks?
- How to ensure
  - Dependability and performance?

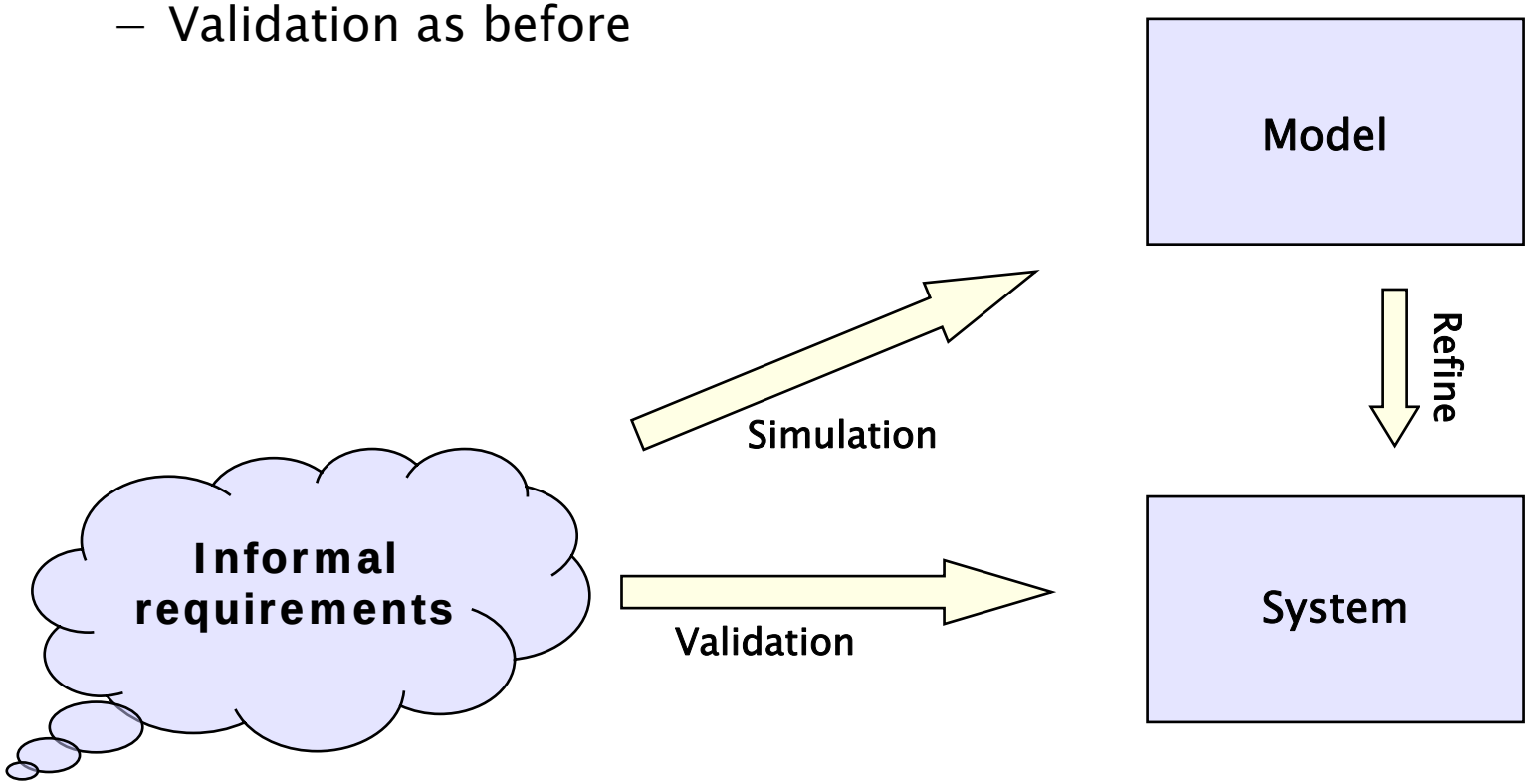
Conventional software engineering

- From requirements to software system
  - Apply design methodologies
  - Code directly in programming language
  - Validation via testing, code walkthroughs



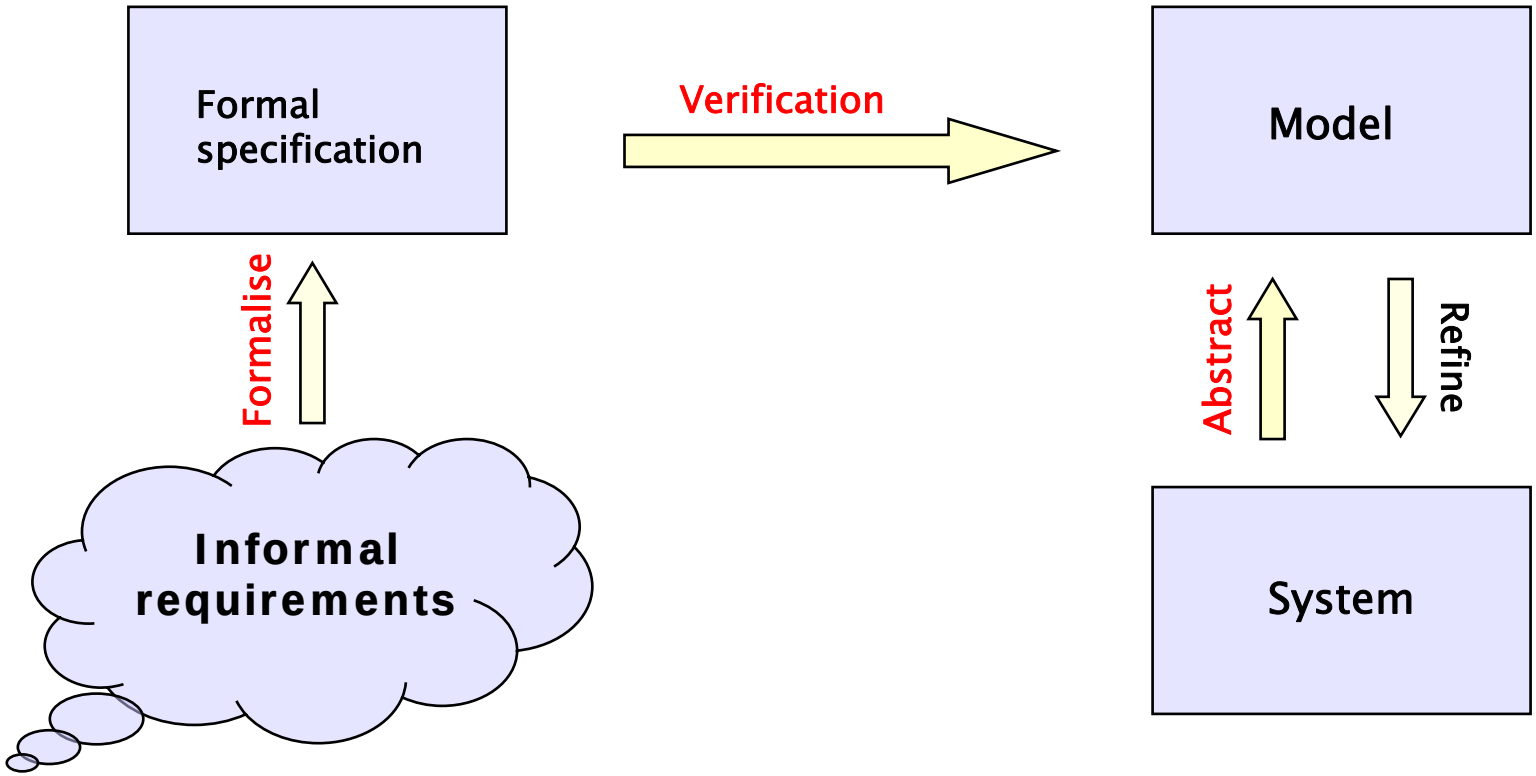
Model-based software engineering

- From requirements to model
  - Simulate the model
  - Generate code
  - Validation as before



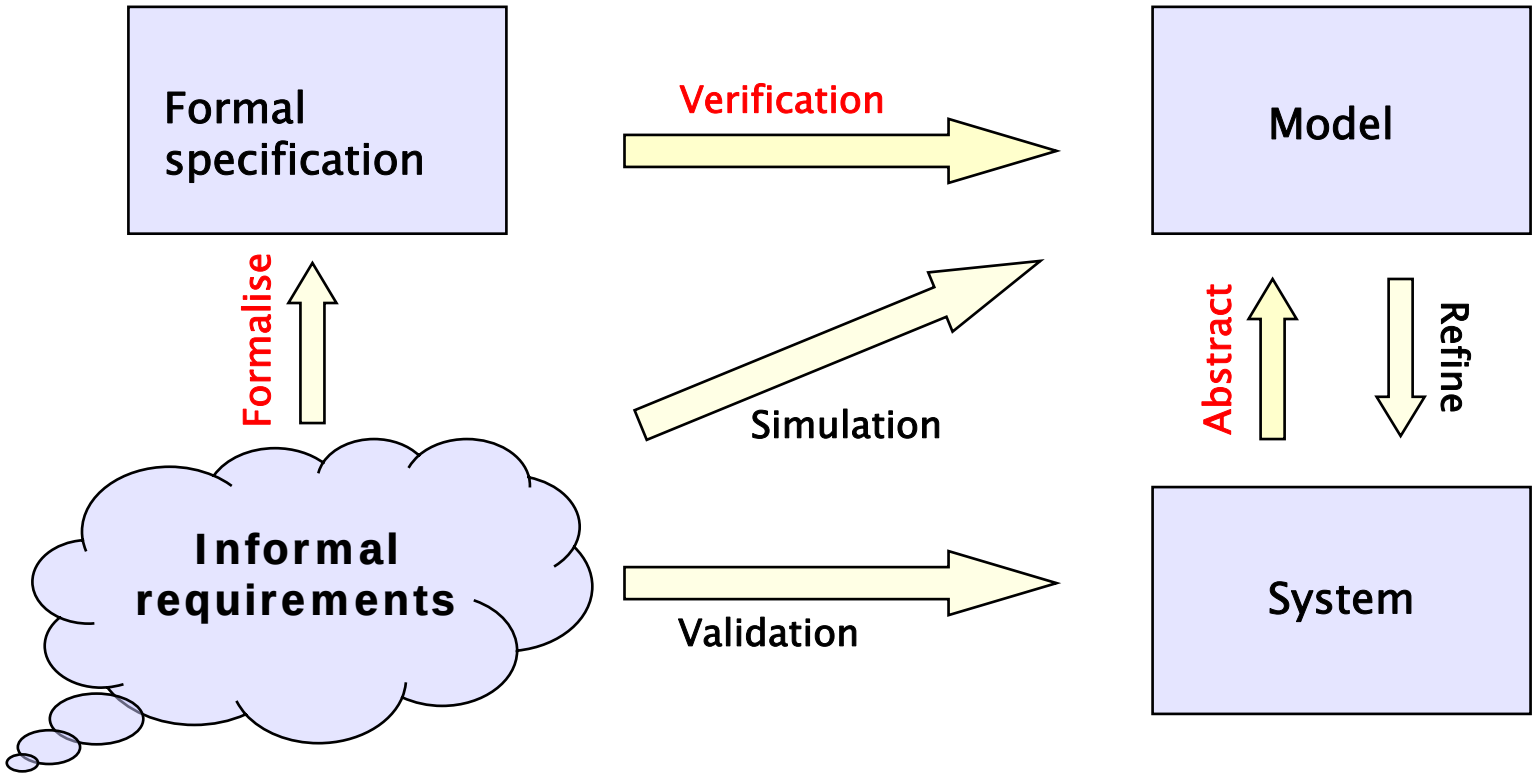
Model-based verification

- From requirements to formal specification
  - Formalise specification, derive model
  - Formally verify correctness, possibly generate code



Software engineering in future

- Verification and validation
  - Derive model, or extract from software
  - Verify correctness, validate if fit for purpose



Why must we verify?

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

"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 Wybe Dijkstra

1930–2002

To rule out errors must consider all possible executions – often not feasible mechanically!

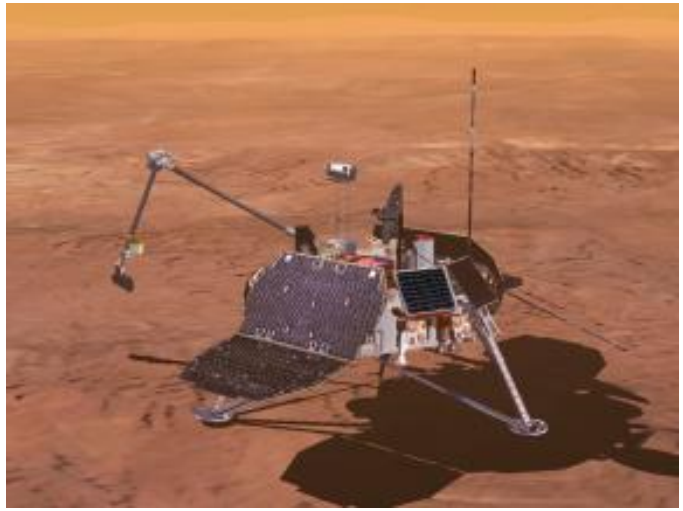
But my program works!

- 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, far greater complexity, higher expectations
  - Automotive drive-by-wire
  - Medical sensors: heart rate & blood pressure monitors
  - Intelligent buildings and spaces: WiFi hotspots, environmental sensors
- Learning from mistakes costly…

The NASA Mars space mission

Mars Climate Orbiter
Launched 11th December 1998

LOST 23rd September 1999
Conversion error from English units to metric in navigation software
Cost: $125 million



Mars Polar Lander
Launched 3rd January 1999

LOST 3rd December 1999
Engine shutdown due to spurious signals that gave false indication that spacecraft had landed

Toyota Prius

Drive-by-wire, in car network

100s of embedded components used in modern cars

2005 Toyota Prius hybrid

In May 2005, Toyota recalls about 75,000 cars.
Some Prius drivers have reported sudden stalling or stopping at highway speeds.

According to reports "the stalling problem is due to a software glitch in its sophisticated computer system."

Such problems are becoming more common: BMW 7 series, …
Cost $?

Code Red

"Code Red" Worm
Exploiting Buffer
Overflow in Microsoft IIS
Indexing Service DLL

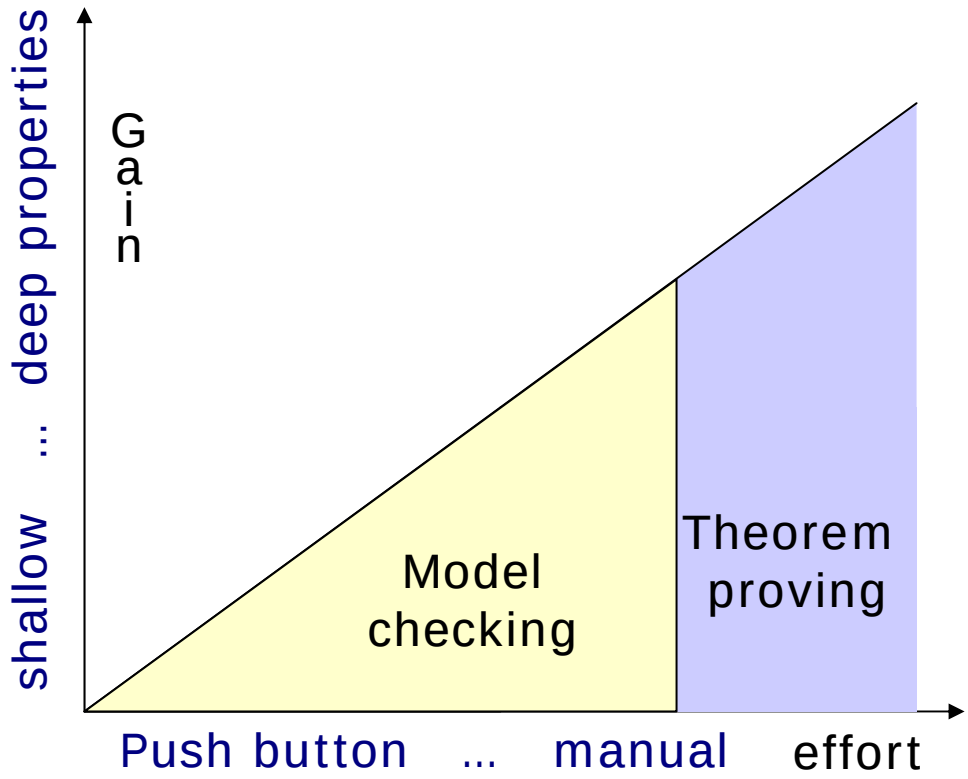
```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%uc
bd3%u7801%u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u819
0%u00c3%u0003%u8b00%u531
b%u53ff%u0078%u0000%u00=a
```

"Code Red" Footprint

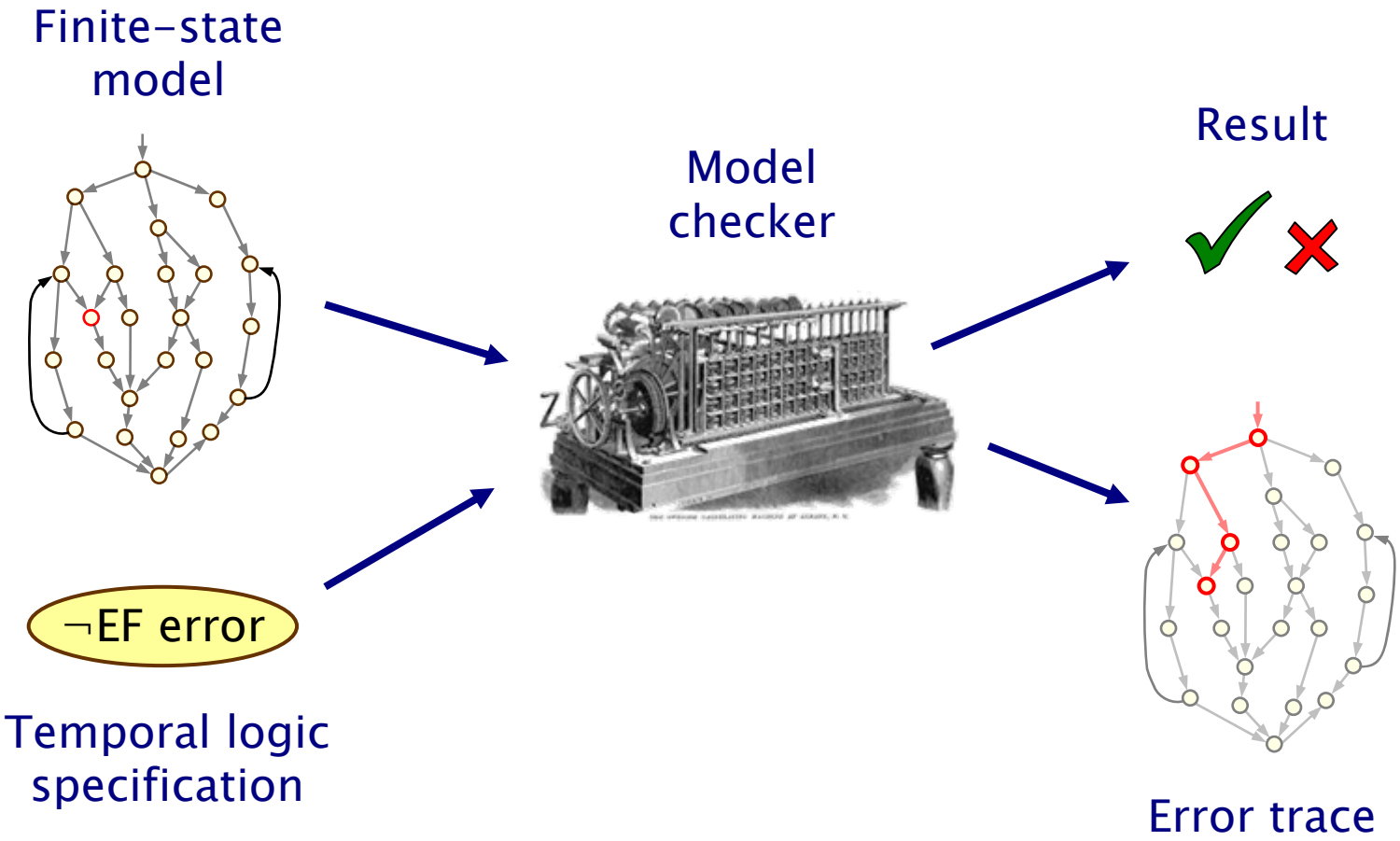
"Code Red" worm is self-replicating malicious code.
It attempts to connect to TCP port 80 on a randomly chosen host, then sends a crafted HTTP GET request to the victim, attempting to exploit a buffer overflow.
Web pages on victim machines may be defaced, suffer performance degradation, denial of service, etc.
Cost > $2 billion

Formal verification techniques

- Mathematical proof
  - manual
- Theorem proving
  - infinite-state models
  - computer-assisted
  - human interaction
- Model checking
  - finite-state models
  - fully automatic
  - exhaustive
- Static analysis
  - shallow properties
  - lightweight
  - fast



Verification via model checking



Role of model checking

- Automated techniques for the assurance of
  - safety
  - security, privacy & trust
  - performance
  - dependability
- NB, quantitative, as well as qualitative requirements:
  - how reliable is my car's Bluetooth network?
  - how efficient is my phone's power management policy?
  - is my bank's web-service secure?
- This course focuses on probabilistic model checking
  - to capture probability and resource usage
  - range of quantitative analyses

Why probability?

- Randomisation used in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding
- To model 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 .."
- In evidence-based, statistical analysis of behaviours
  - to quantify trust, anonymity, etc
- In modelling of biological processes
  - to quantify concentrations or numbers of molecules

    "the expected long-run percentage of Na molecules is … "

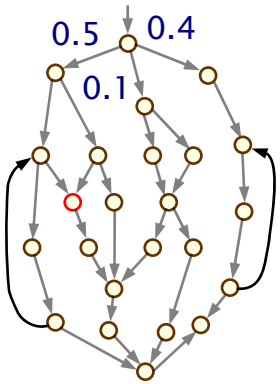
Real-world protocol examples

- Protocols featuring randomisation
  - Randomised back-off schemes
    - CSMA protocol
    - 802.11 Wireless LAN
  - Random choice of waiting time
    - IEEE 1394 Firewire root contention
    - Bluetooth, device discovery phase
  - Random choice over a set of possible addresses
    - IPv4 Zeroconf dynamic configuration (link-local addressing)
  - and more

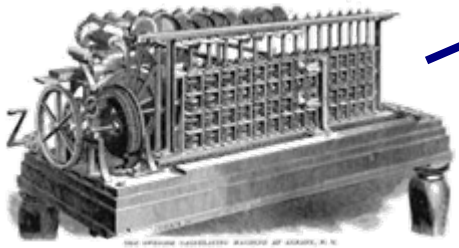
- Continuous probability distribution needed to model network traffic, node mobility, random delays…

Probabilistic model checking

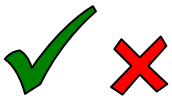
Probabilistic model
e.g. Markov chain

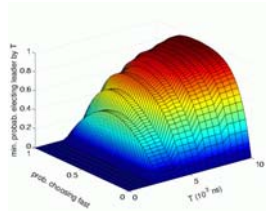


Probabilistic model checker
e.g. PRISM



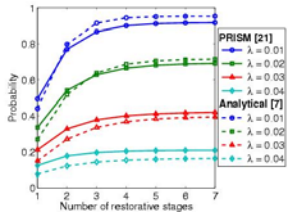
Result

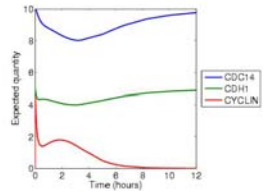




$P_{<0.01}$ [F error]

Probabilistic temporal logic specification
e.g. PCTL





Quantitative results

Probabilistic model checking inputs

- Models: variants of Markov chains
  - Discrete-Time Markov Chains (DTMCs)
  - Markov Decision Processes (MDPs)
  - Continuous-Time Markov Chains (CTMCs)
  - Probabilistic Time Automata (PTAs)
- Specifications (informally)
  - "probability of delivery within time deadline is …"
  - "expected time to message delivery is …"
  - "expected power consumption is …"
- Specifications (formally)
  - Probabilistic temporal logics (PCTL, CSL, PTCTL)
  - Probability, time, cost/rewards

Probabilistic model checking involves…

- Construction of models
  - from a high-level modelling language
  - e.g. probabilistic process algebra
- Implementation of probabilistic model checking algorithms
  - graph-theoretical algorithms, combined with
    - (probabilistic) reachability
  - numerical computation – iterative methods
    - quantitative model checking (plot values for a range of parameters)
    - typically, linear equation or linear optimisation
    - exhaustive, unlike simulation
  - also sampling-based (statistical) for approximate analysis
    - e.g. hypothesis testing based on simulation runs

Course aims

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

Course structure

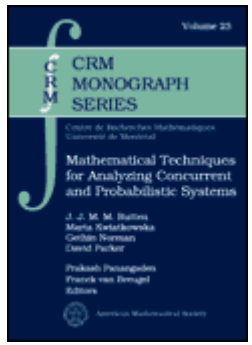
- The course covers four probabilistic models in order of increasing complexity…
  - Discrete-time Markov chains (DTMCs)
    - discrete time, discrete probabilistic behaviours only
  - Markov decision processes (MDPs)
    - as above plus, additionally, nondeterminism
  - Continuous-time Markov chains (CTMCs)
    - continuous time, continuous probabilistic behaviours
  - Probabilistic timed automata (PTAs)
    - continuous time, discrete probabilities, nondeterminism
- It also covers implementation of probabilistic model checking, in particular, symbolic techniques

The PRISM tool

- The PRISM model checker
  - supports all probabilistic models discussed here
  - (direct support for D/CTMCs, MDPs, indirect for PTAs)
  - www.prismmodelchecker.org
- The PRISM tutorial
  - provides a practical introduction to PRISM and is designed to accompany this lecture course
  - www.prismmodelchecker.org/tutorial/

Further information

- The following textbook covers much of the material from this set of lectures:
  - J. Rutten, M. Kwiatkowska, G. Norman and D. Parker
  - Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems
  - P. Panangaden and F. van Breugel (editors), CRM Monograph Series, vol. 23, AMS, 2004



- See also the PRISM web site:
  - www.prismmodelchecker.org
  - for case studies, publications and much more
- The most up-to-date version of these lectures is here:
  - www.prismmodelchecker.org/lectures/

Acknowledgements

- This set of lectures has been prepared by:
  - Marta Kwiatkowska
  - Gethin Norman
  - Dave Parker

- Various aspects of the slides and examples also appear courtesy of:
  - Yi Zhang
  - Hakan Younes
  - Ansgar Fehnker
  - Randal Bryant
  - Joost-Pieter Katoen