Model Repair for Markov Decision Processes

Marta Kwiatkowska

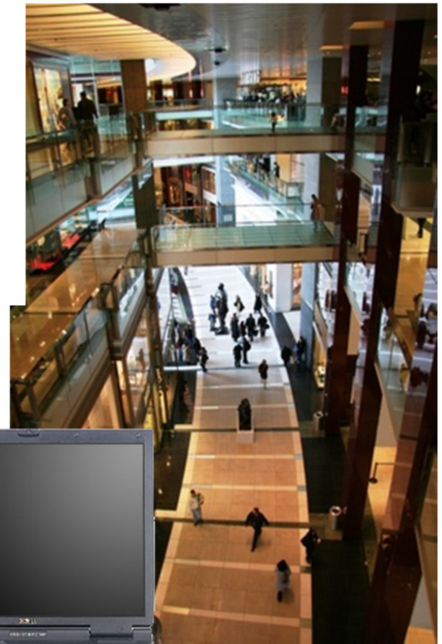
Department of Computer Science, University of Oxford

TASE 2013, Birmingham

Joint work with: T. Chen, E.M. Hahn, T. Han, H. Qu and L. Zhang

Software everywhere

- Electronic devices, ever smaller
  - Laptops, phones, sensors…
- Networking
  - Wireless & Internet everywhere
- Intelligent spaces
  - Buildings, vehicles…
- Systems
  - Adaptive
  - Context-aware
  - Self-*



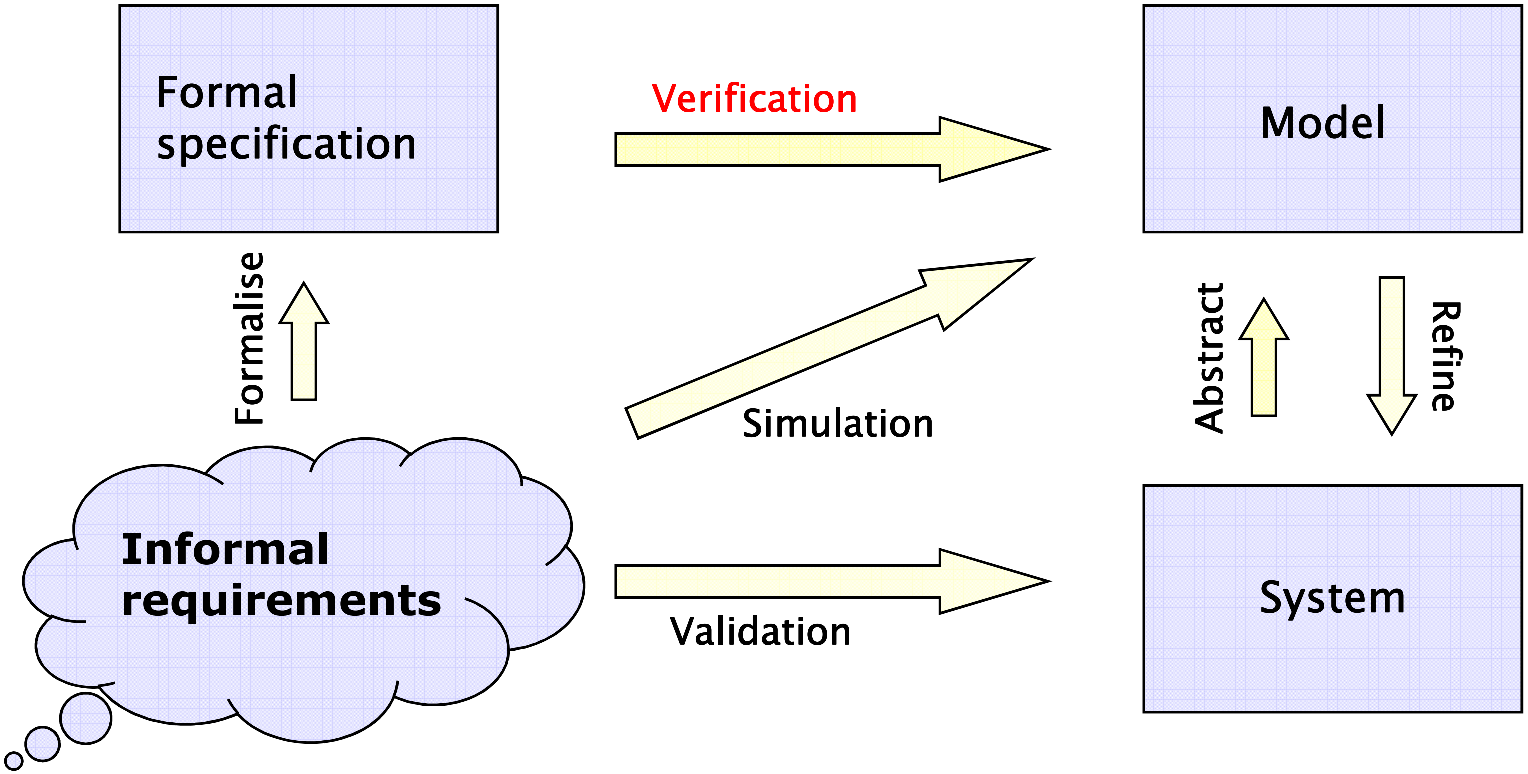
- From hardware and software, to everyware
  - Household objects do information processing
  - Software is central

Software quality assurance

- Software is a critical component of embedded systems
  - software failure costly and life endangering
- Need quality assurance methodologies
  - model-based development
  - rigorous software engineering
  - software product lines
- Use formal techniques to produce guarantees for:
  - safety, reliability, performance, resource usage, trust, …
  - (safety) "probability of failure to raise alarm is tolerably low"
  - (reliability) "the smartphone will never execute the financial transaction twice"
- Focus on automated, tool-supported methodologies
  - automated verification via model checking
  - quantitative verification

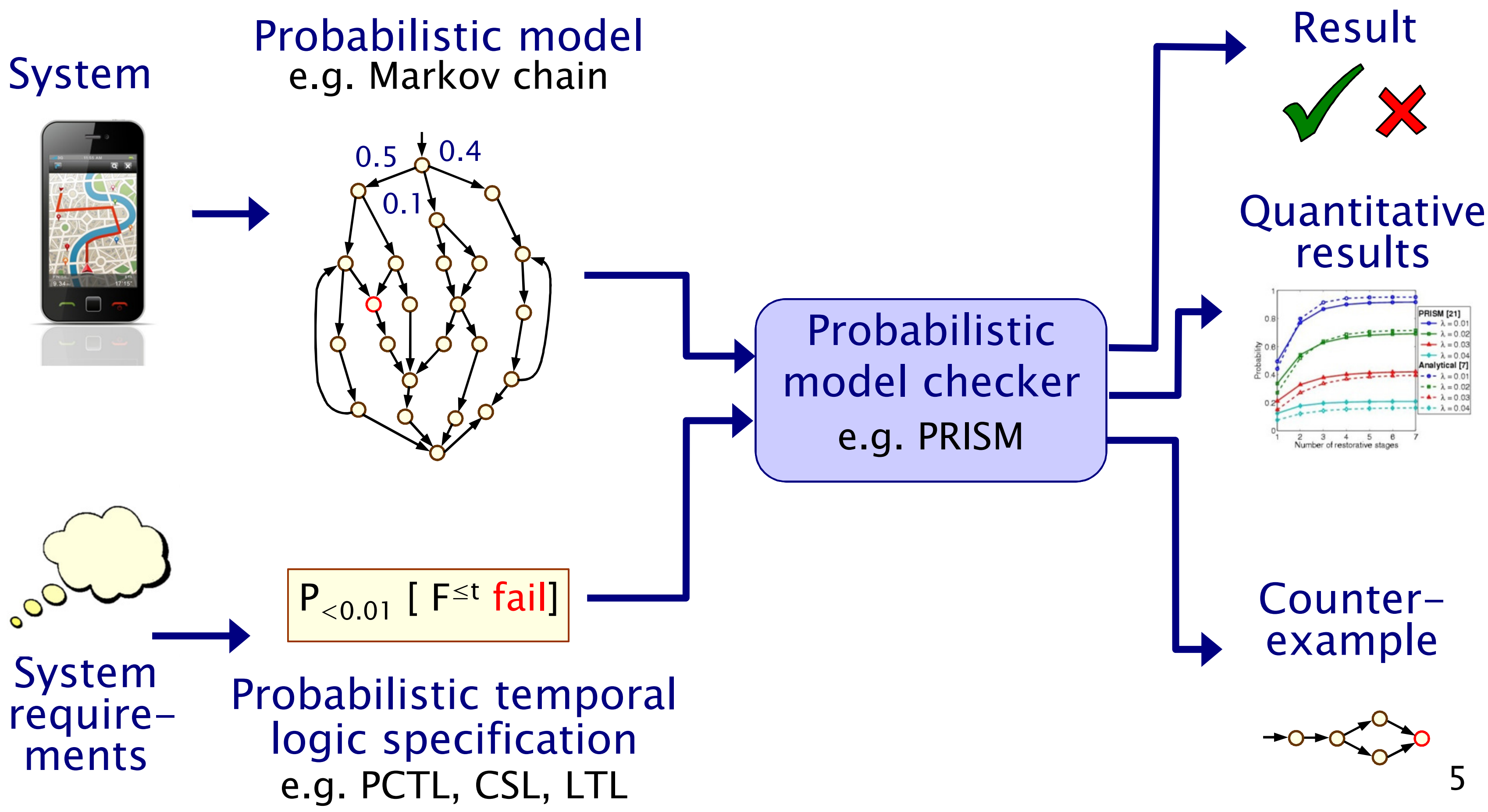
Rigorous software engineering

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



Quantitative (probabilistic) verification

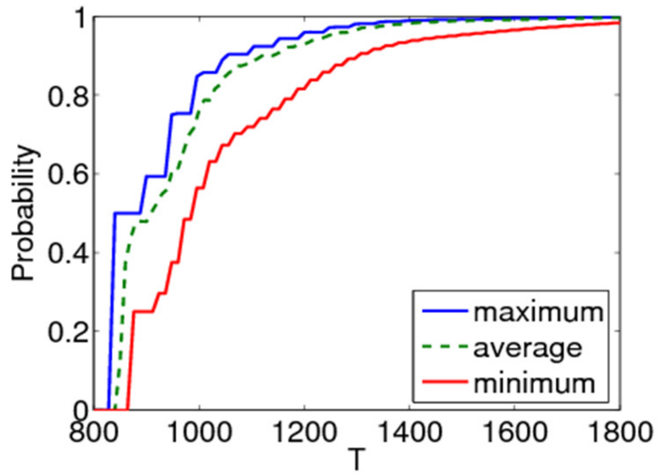
Automatic verification (aka model checking) of quantitative properties of probabilistic system models



Why quantitative verification?

- Real software/systems are quantitative:
  - Resource constraints
    - energy, buffer size, number of unsuccessful transmissions, etc
  - Randomisation, e.g. in distributed coordination algorithms
    - random delays/back-off in Bluetooth, Zigbee
  - Uncertainty, e.g. communication failures/delays
    - prevalence of wireless communication

- Analysis “quantitative” & “exhaustive”
  - strength of mathematical proof
  - best/worst-case scenarios, not possible with simulation
  - identifying trends and anomalies



Quantitative properties

- Simple properties
  - $P_{\leq 0.01}$ [F "fail"] – "the probability of a failure is at most 0.01"

- Analysing best and worst case scenarios
  - $P_{max=?}$ [$F^{\leq 10}$ "outage"] – "worst-case probability of an outage occurring within 10 seconds, for any possible scheduling of system components"
  - $P_{=?}$ [$G^{\leq 0.02}$!"deploy" {"crash"}{max}] – "the maximum probability of an airbag failing to deploy within 0.02s, from any possible crash scenario"

- Reward/cost-based properties
  - $R_{\{\text{"time"}\}=?}$ [F "end"] – "expected algorithm execution time"
  - $R_{\{\text{"energy"}\}max=?}$ [$C^{\leq 7200}$] – "worst-case expected energy consumption during the first 2 hours"

Historical perspective

- First algorithms proposed in 1980s
  - [Vardi, Courcoubetis, Yannakakis, …]
  - algorithms [Hansson, Jonsson, de Alfaro] & first implementations
- 2000: tools ETMCC (MRMC) & PRISM released
  - PRISM: efficient extensions of symbolic model checking [Kwiatkowska, Norman, Parker, …]
  - ETMCC (now MRMC): model checking for continuous-time Markov chains [Baier, Hermanns, Haverkort, Katoen, …]
- Now mature area, of industrial relevance
  - successfully used by non-experts for many application domains, but full automation and good tool support essential
    - distributed algorithms, communication protocols, security protocols, biological systems, quantum cryptography, planning…
  - genuine flaws found and corrected in real-world systems

Quantitative probabilistic verification

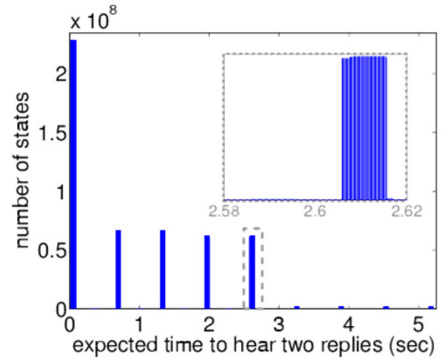
- What's involved
  - specifying, extracting and building of quantitative models
  - graph-based analysis: reachability + qualitative verification
  - numerical solution, e.g. linear equations/linear programming
  - typically computationally more expensive than the non-quantitative case
- The state of the art
  - fast/efficient techniques for a range of probabilistic models
  - feasible for models of up to 10^7 states (10^{10} with symbolic)
  - extension to probabilistic real-time systems
  - abstraction refinement (CEGAR) methods
  - probabilistic counterexample generation
  - assume-guarantee compositional verification
  - tool support exists and is widely used, e.g. PRISM, MRMC

Tool support: PRISM

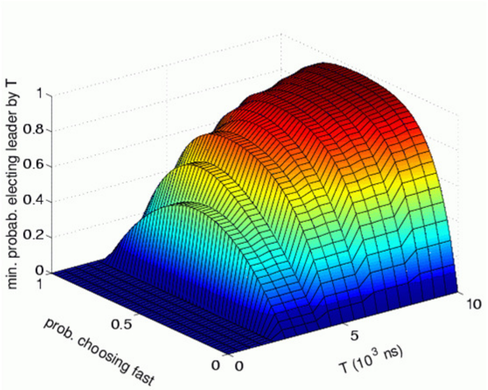
- PRISM: Probabilistic symbolic model checker
  - developed at Birmingham/Oxford University, since 1999
  - free, open source software (GPL), runs on all major OSs
- Support for:
  - models: DTMCs, CTMCs, MDPs, PTAs, SMGs, …
  - properties: PCTL/PCTL*, CSL, LTL, rPATL, costs/rewards, …
- Features:
  - simple but flexible high-level modelling language
  - user interface: editors, simulator, experiments, graph plotting
  - multiple efficient model checking engines (e.g. symbolic)
- Many import/export options, tool connections
  - MRMC, INFAMY, DSD, Petri nets, Matlab, …
- See: http://www.prismmodelchecker.org/

Quantitative verification in action

- Bluetooth device discovery protocol
  - frequency hopping, randomised delays
  - low-level model in PRISM, based on detailed Bluetooth reference documentation
  - numerical solution of 32 Markov chains, each approximately 3 billion states
  - identified worst-case time to hear one message



- FireWire root contention
  - wired protocol, uses randomisation
  - model checking using PRISM
  - optimum probability of leader election by time T for various coin biases
  - demonstrated that a biased coin can improve performance



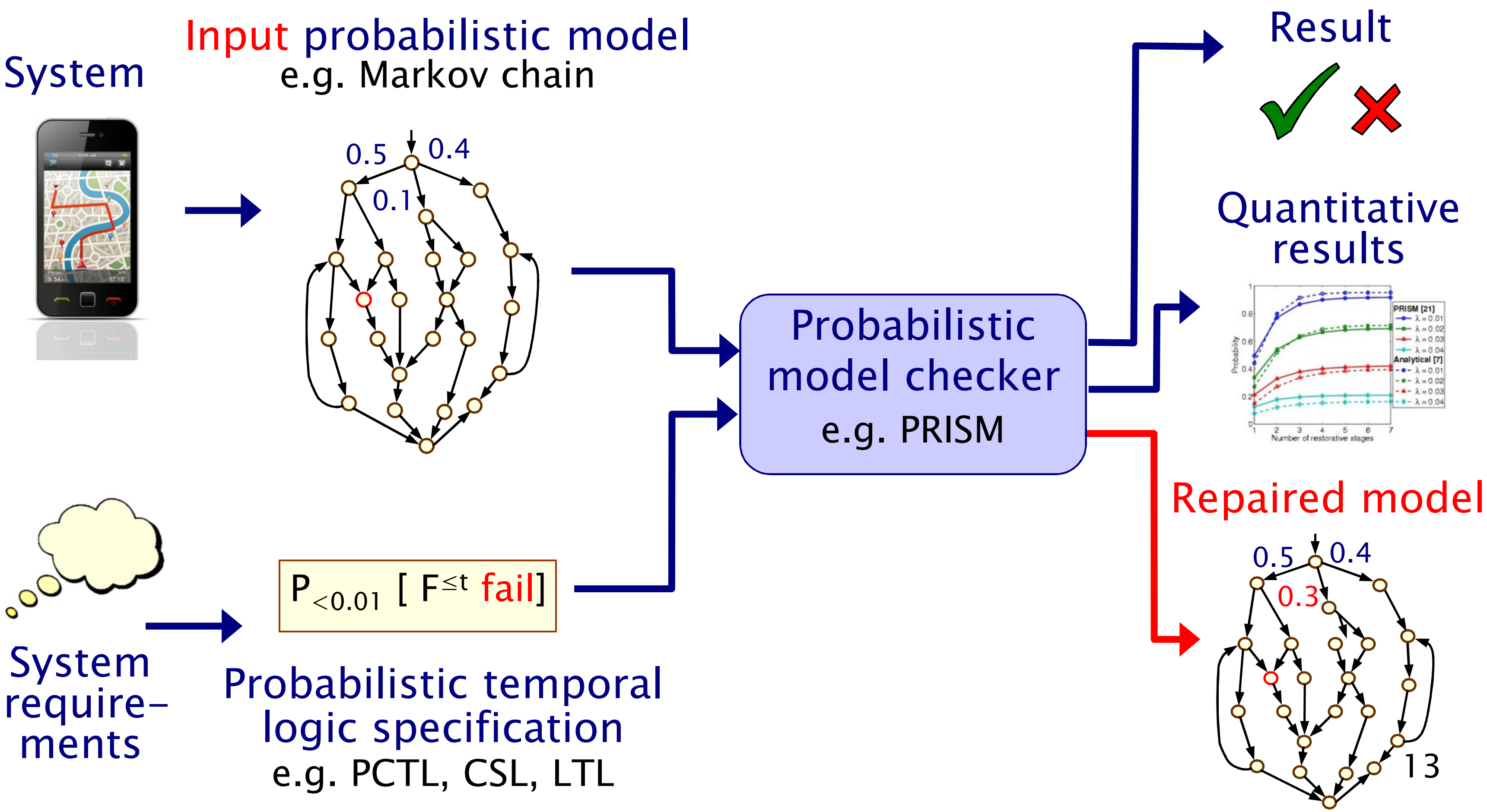
This lecture…

- What to do if quantitative verification fails?

- Majority of research to date has focused on verification
  - scalability and performance of algorithms
  - extending expressiveness of models and logics
  - real-world case studies
- Some work to date on counterexamples [Han&Katoen 2009, Aljazzar&Leue 2009]
  - need to capture two types of branching
  - but difficult to represent them compactly
- In this lecture, we focus on model repair
  - can we fix the model to guarantee that a quantitative property is satisfied?
  - adjust parameters, potentially for use at runtime

Quantitative (probabilistic) verification

Automatic verification (aka model checking) of quantitative properties of probabilistic system models



Overview

- Model repair
  - problem statement
  - parametric probabilistic models
  - property specifications: probability/expectation
- Region-based method
  - constraint-based approximate solution
- Sampling-based methods
  - randomised search through the parameter space
  - Markov chain Monte Carlo, Cross-Entropy and Particle Swarm
- Case study: network virus

Probabilistic models

- Discrete-time Markov chains (DTMCs)
  - discrete states + probability
  - for: randomisation, component failures, unreliable media

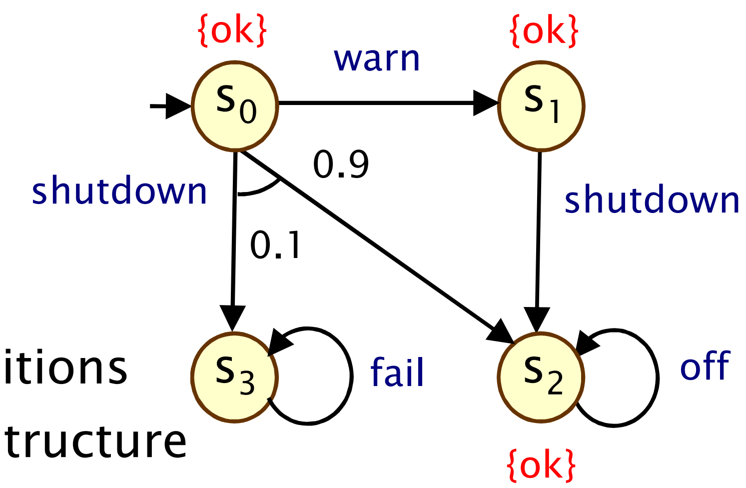
- Markov decision processes (MDPs) ← this talk
  - discrete states + probability + nondeterminism
  - for: concurrency, control, under-specification, abstraction

- Stochastic multi-player games
- Continuous-time Markov chains (CTMCs)
- Probabilistic timed automata (PTAs)
- Labelled Markov processes (LMPs)
  - and many other variants…

Markov decision processes (MDPs)

- Useful for modelling e.g. distributed protocols with failure or randomisation

- An MDP is a tuple $M = (S, s_0, Act, P, L, r)$:
  - S is the state space
  - $s_0 \in S$ is the initial state
  - Act is finite set of actions
  - $P: S \times Act \times S \rightarrow [0,1]$ is the probability matrix
  - L is labelling with atomic propositions
  - $R: S \times Act \rightarrow Real_{\geq 0}$ is a reward structure



- such that
  - each row of P sums up to 0 or 1
  - for every state s, at least one a is enabled in s

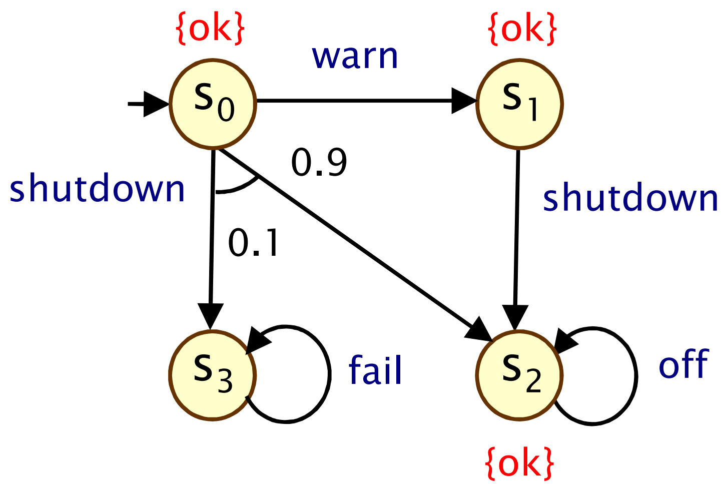
Probabilistic model checking for MDPs

- To reason formally about MDPs, we use adversaries
  - an adversary σ resolves nondeterminism in a MDP M
  - also called “scheduler”, “strategy”, “policy”, …
  - makes a (possibly randomised) choice, based on history
  - induces probability measure Pr_M^σ over (infinite) paths

- Property specifications: probabilistic and expected reward
  - specify probabilistic property $P_{\geq p}[\phi]$ in PCTL, ϕ path property
  - $Pr_M^\sigma(\phi)$ gives probability of ϕ under adversary σ
  - best-/worst-case analysis: quantify over all adversaries
  - e.g. $M \models P_{\geq p}[G \text{ “ok”}] \Leftrightarrow Pr_M^\sigma(G \text{ “ok”}) \geq p$ for all σ
  - or just compute e.g. $Pr_M^{min}(\phi) = \inf \{ Pr_M^\sigma (G \text{ “ok”}) \mid \sigma \in Adv_M \}$
  - efficient algorithms and tools exist
  - Reward properties involve computing expectations

Model repair: problem statement

- Assume we have an MDP…

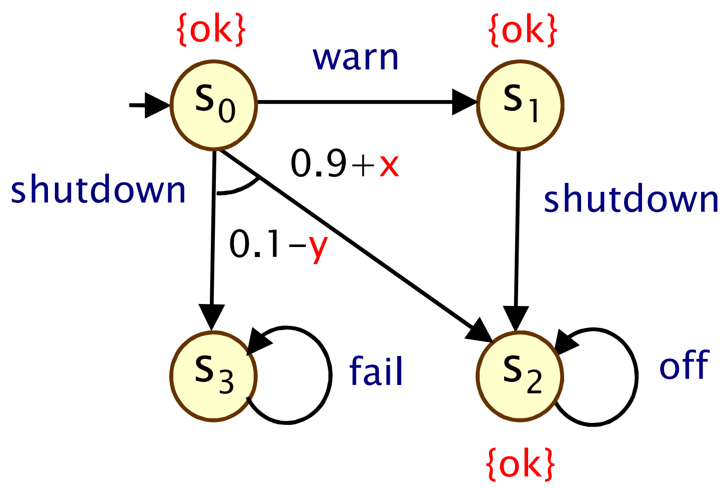


- which does not satisfy a given property, e.g.
  - $M \nvDash P_{\geq 0.99}[G \text{ “ok”}]$
- We wish to repair this model so that it does
- Solved for discrete-time Markov chains wrt reachability or expected accumulated rewards in [Bartocci et al 2011]

Main idea

- Transform to a parametric MDP
  - by adding parameters to each transition that we can modify

$M_{param}\langle x,y\rangle =$



- Find instantiations v of parameters such that
  - $M_{param}\langle v\rangle$ satisfies property, ie $M_{param}\langle v\rangle \Vdash P_{\geq 0.99}[G \text{ “ok”}]$, and
  - some objective function f(v) is minimal (repaired model is nearest wrt to some cost/distance measure)
  - e.g. $f(x,y) = x^2 + y^2$ (sum of squares)

Our contribution

- Unfortunately the methods developed for DTMCs do not transfer to MDPs
  - cannot guarantee existence of single rational function over parameters
- We extend model repair to general MDPs by approximating the solution
- Consider both probabilistic and reward properties
- Two complementary approaches implemented in PRISM
- Region-based approach
  - based on computing functions describing property depending on parameters using constraint programming
- Sampling-based optimisation
  - stochastic search through the parameter space
  - may yield a suboptimal solution but faster

Formally…

- Given
  - V set of variables, span(V) set of linear expressions over V
  - PCTL formula ϕ
  - MDP $M = (S, s_0, Act, P, L, r)$ s.t. $M \not\models \phi$
  - $Z: S \times Act \times S \rightarrow span(V)$ transition repair matrix
  - $z: S \times Act \rightarrow span(V)$ reward repair matrix
- Define parametric MDP $M' = (S, s_0, Act, P+Z, L, r+z)$

- The model repair problem for MDP M, formula ϕ and polynomial g over variables V is to find evaluation $v: V \rightarrow Real$ satisfying
  - $v \in \arg\min g<v>$ (minimise cost)
  - v is a valid evaluation (yielding a valid MDP)
  - $M'<v> \VDash \phi$

Fast model repair

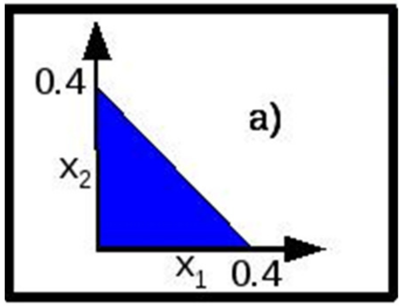
- Many practical situations demand fast parameter adaptation, typically at runtime, to guarantee some performance property, e.g.
  - self-adaptive systems
  - replacement of failed component in multiprocessor systems

- Fast model repair is defined, for b a real-valued bound, Q a penalty function, as finding an evaluation satisfying
  - $g<v>+Q<v> \leq b$ and
  - running time should be fast, trading off optimality

- The value of b is typically small to keep cost of repair sufficiently low though suboptimal
  - $b=0.0$ allowed but may result in slower repair

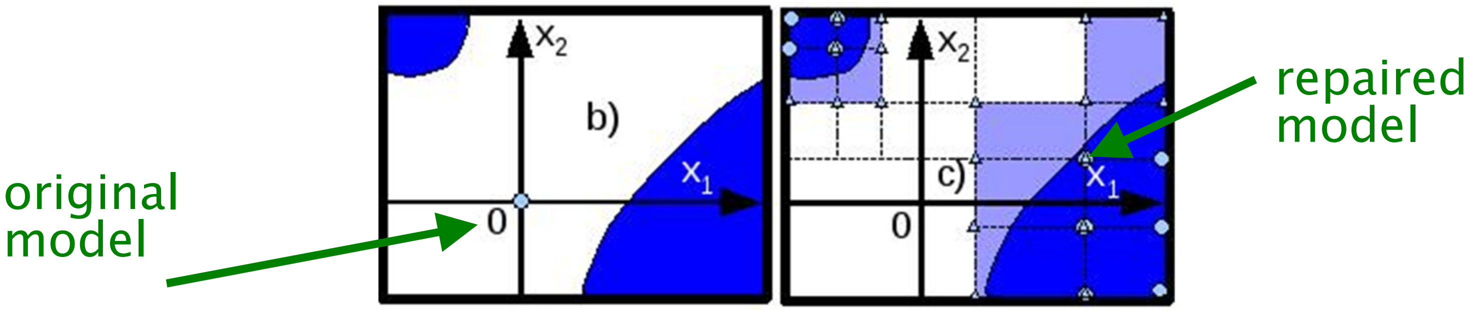
Region-based approach

- Building upon method developed earlier for parametric Markov processes in [Hahn, Han and Zhang 2011]
  - finding parameter values to guarantee satisfaction of a PCTL formula
  - assume parameter range, ie interval of values [l,u]
  - allows working with hyper-rectangles
- Does not apply to model repair…
  - need to ensure probabilities are nonnegative
  - problem if repair matrix increases two transitions while decreasing another by the same amount
  - i.e. constraints are triangles
- Obtain approximate solution...



More on region-based approach

- Encode the validity of parameter valuations into the formula, ϕ_{valid} , and derive PMDP M' as before
- Repeatedly subdivide regions into those for which the property is valid, invalid and undecided
  - point $x_1=x_2=0$ is the original (unrepaired) model
- Use constraint solving to compute approximate ϵ-solution (fraction of the parameter space left undecided)
- Can evaluate repair cost g at vertices, then take minimum of those values to obtain lower bound



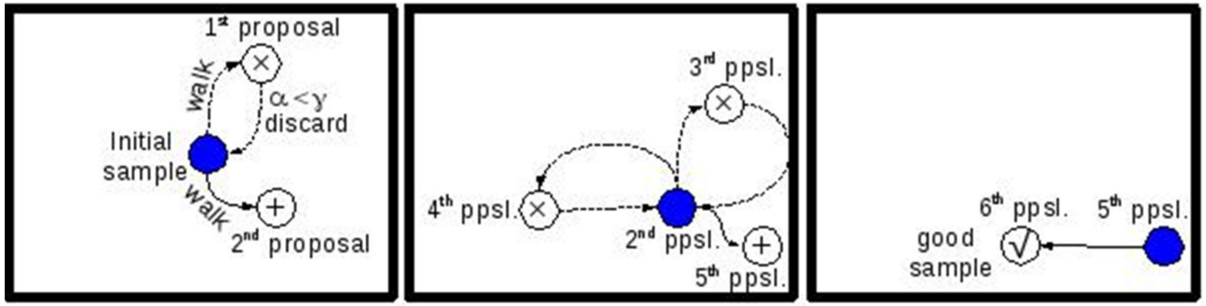
Sampling-based approach

- Three methods based on randomised search
- Work with the formulation, for bound b:
  - $g\langle v\rangle + Q\langle v\rangle \leq b$
- where
  - Q is a penalty function defined by

    $Q\langle v\rangle = 0$ if $M'\langle v\rangle \Vdash \phi$ and otherwise some value δ
  - used to guide the search towards good valuations
- Challenge: we draw samples according to an unknown probability distribution
  - $pd(v) = e^{-\beta O(v)}$
  - where O is the objective function, β weighting factor
  - so need to adapt the three methods to this scenario
  - use threshold for maximum number of samples, terminate the procedure when good sample reached

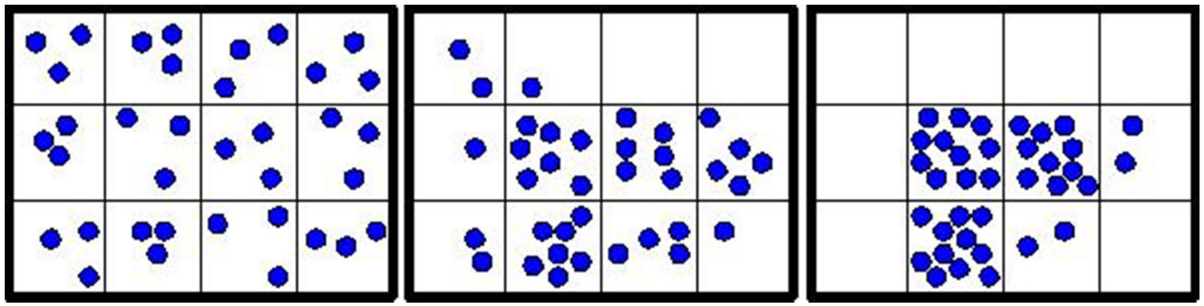
Markov chain Monte Carlo

- Use the Metropolis-Hastings algorithm
- Generates a series of samples
  - linked in a Markov chain
  - each sample correlated only with the directly preceding sample
  - in the long run, the distribution matches the desired probability distribution pd
- Performs random walk about the sample space, sometimes accepting and sometimes not



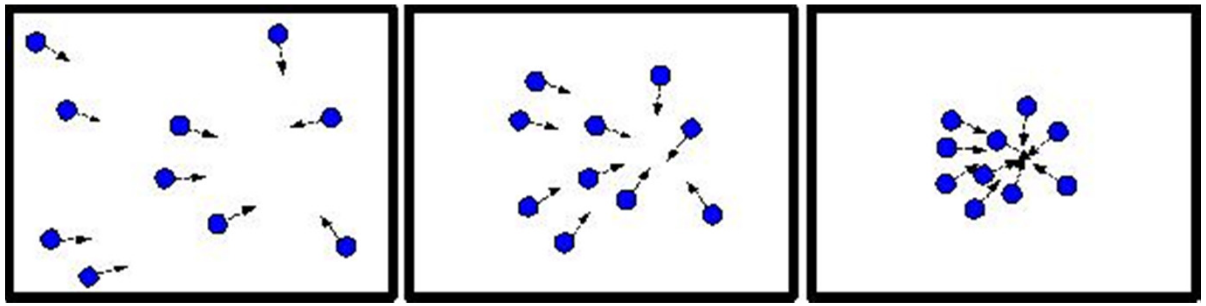
Cross-Entropy method

- Starts from a family of distributions and attempts to find a distribution which is as close as possible to pd
  - use Kullback-Leibler (KL) divergence measure
- Works as follows
  - partition the parameter space into cells, parameterised by probability that a point from cell is sampled
  - generate samples based on the candidate distribution
  - tilt the samples towards the new distribution, by minimising KL distance over samples



Particle swarm optimisation

- Based on movement of a bird flock
- Swarm of n particles
  - each with velocity, indicating where it is moving to
- Update the velocity vector by randomised combination of
  - direction to the best position of i-th particle, and
  - direction to best global particle position
- Terminate when norm of velocity smaller than ϵ

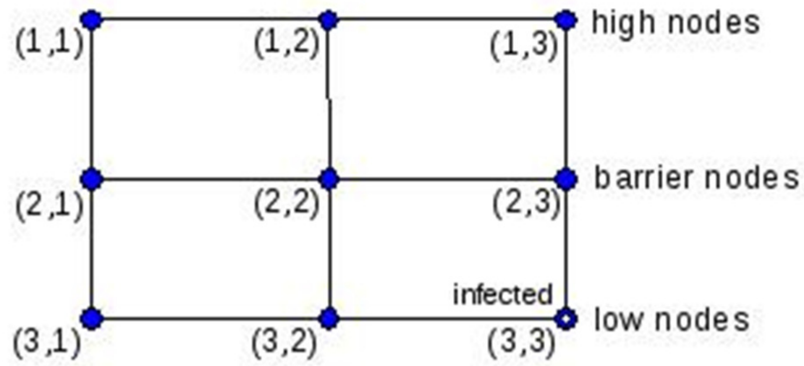


PRISM support

- Implemented both the region-based and sampling approaches in PRISM
  - ‘explicit’ engine, written in Java
  - region-based approach is a reimplementation of PARAM 2.0
  - sampling-based approaches are new implementation
  - to be included in a forthcoming release
- Input models specified as parametric PRISM models
  - parameters expressed as unevaluated constants
  - e.g. const double x;
  - repairable transition specified as 0.4 + x
  - general purpose, other types of usage
- Properties are given in PCTL, with parameter constants
  - new construct constfilter (min, x1*x2, prop)
  - filters over parameter values, rather than states

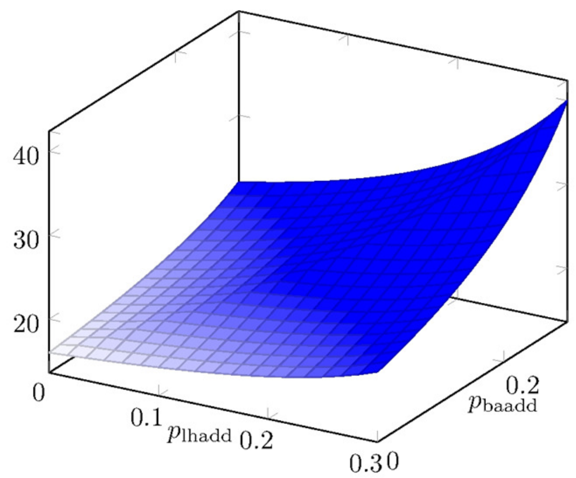
Case study: network virus

- Parametric model of a network virus
  - a grid of connected nodes
  - virus spawns/multiplies
  - once infected, virus repeatedly tries to spread to neighbouring nodes
  - there are 'high' and 'low' nodes, with barrier nodes from 'high' to 'low'
  - choice of infection by virus probabilistic
  - choice of which node to infect nondeterministic

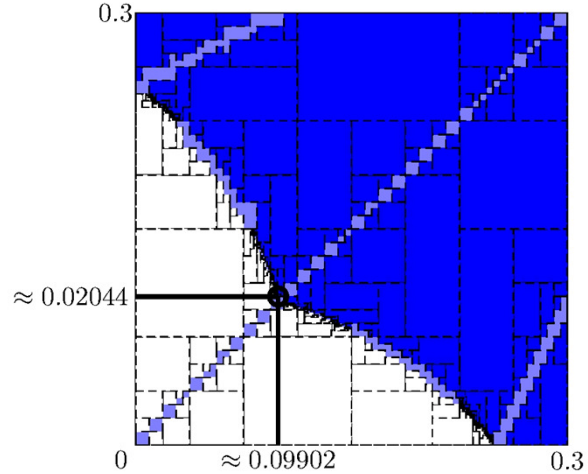


- Property specification
  - minimal expected number of attacks until infection of (1,1), starting from (N,N), is upper bounded by 20
  - probability of detection and of barrier nodes subject to repair by increasing p_{lhadd} and p_{baadd}

Case study: region-based methods



Plot of minimal expected number of attacks



Checking if minimal exp. number of attacks >= 20

Property constfilter(min,…,$R_{\{\text{“attacks”}\}>=20}$ [F “inf-11”])

Model has 809 states, ϵ = 0.05

Optimal value found in 2mins, showing repair values

Case study: sampling-based methods

- Need to work with the formulation $g\langle v\rangle+Q\langle v\rangle \leq b$
- Test two bounds, $b = 0.0$ and $b = 0.0225$
  - MCMC slower for bound $b = 0.0$, can be unstable for the larger bound
  - both CE and PSO are stable
  - PSO better performance
- Sampling methods have superior performance wrt region-based methods
  - all terminate within 20s, vs 2 mins for region-based
  - 200-500 samples
  - PSO mostly able to finish in 5s
- Hence, demonstrated practical applicability for online model repair
  - trading optimality for speed

Conclusions

- Formulated and proposed approximate solution to model repair for Markov decision processes
  - MDPs widely used to model network and security protocols, distributed systems with failure, etc
  - parametric models integrated within PRISM
  - full PCTL with the reward operator

- Demonstrated
  - sampling-based model repair feasible for runtime use
  - but scalability is still the biggest challenge

- Model repair for other probabilistic models
  - also adapted to Markov reward models, work in progress
  - incl. DTMCs and CTMCs (via discretisation)

Quantitative verification – Trends

- Being ‘younger’, generally lags behind conventional verification
  - much smaller model capacity
  - compositional reasoning in infancy
  - automation of model extraction/adaptation very limited
- Tool usage on the increase, in academic/industrial contexts
  - real-time verification/synthesis in embedded systems
  - probabilistic verification in security, reliability, performance
- Shift towards greater automation
  - specification mining, model extraction, synthesis, verification, …
- But many challenges remain!

Future directions

- Many challenges remain
  - computational runtime steering, away from danger states, in addition to online model repair
  - effective model abstraction/reduction techniques
  - scalability of monolithic/runtime verification
  - approximate methods
- More challenges not covered in this lecture
  - correct-by-construction model synthesis from specifications
  - controller synthesis
  - more expressive models and logics
  - code generation
  - new application domains, …
- and more…

Acknowledgements

- My collaborators in this work
- Project funding
  - ERC, EPSRC LSCITS
  - Oxford Martin School, Institute for the Future of Computing

- See also
  - PRISM www.prismmodelchecker.org
  - VERIWARE www.veriware.org