Modelling and verification of probabilistic systems

Marta Kwiatkowska

School of Computer Science

THE UNIVERSITY OF BIRMINGHAM

www.cs.bham.ac.uk/~mzk

www.cs.bham.ac.uk/~dxp/prism

Liverpool, 1st March 2005

Overview

- Motivation
- Probabilistic model checking
  - The models
  - Specification languages
  - What does it involve?
  - The PRISM model checker
- Case studies
  - Molecular reactions
  - IPv4 Zeroconf dynamic configuration protocol
  - Bluetooth device discovery
- Challenges for future

The future: ubiquitous computing

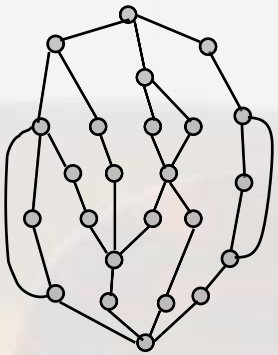
Mobile, wearable, wireless devices (WiFi, Bluetooth)
Ad hoc, dynamic, ubiquitous computing environment
Security, privacy, anonymity protection on the Internet
Self-configurable - no need for men/women in white coats!
Fast, responsive, power efficient, …

Probability helps

- In distributed co-ordination algorithms
  - As a symmetry breaker
    - "leader election is eventually resolved with probability 1"
  - In gossip-based routing and multicasting
    - "the message will be delivered to all nodes with high probability"
- When modelling uncertainty in the environment
  - To quantify failures, express soft deadlines, QoS
    - "the chance of shutdown is at most 0.1%"
    - "the probability of a frame delivered within 5ms is at least 0.91"
  - To quantify environmental factors in decision support
    - "the expected cost of reaching the goal is 100"
- When analysing system performance
  - To quantify arrivals, service, etc, characteristics
    - "in the long run, mean waiting time in a lift queue is 30 sec"

Verification via model checking…

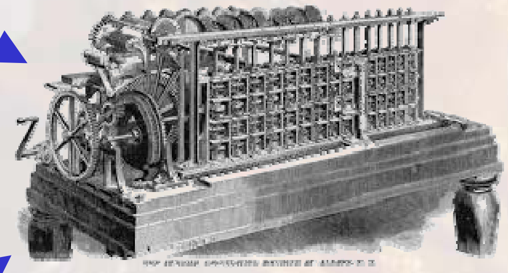
or falsification?



The model

$\boxed{send \rightarrow \Diamond deliver}$

Temporal logic specification



Model Checker

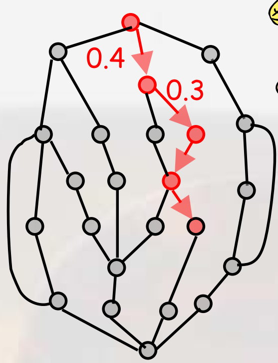
or

Error trace

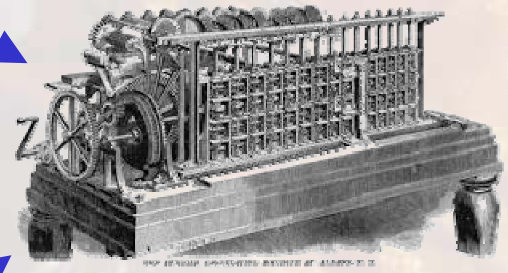
Line 5: …
Line 21: …
Line 15: …
…
Line 27: …
Line 45: …

Probabilistic model checking…

in a nutshell



Probabilistic model



Probabilistic Model Checker

or

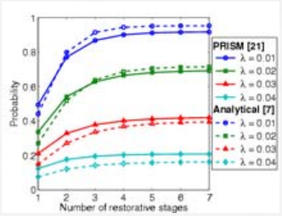
or

The probability

send $\rightarrow \mathbf{P}_{\geq 0.9}(\Diamond \text{deliver})$

Probabilistic temporal logic specification

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
…
State 12: 0
State 13: 0.1245



Probability elsewhere

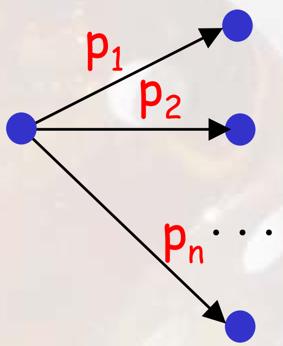
- In performance modelling
  - Pioneered by Erlang, in telecommunications, ca 1910
  - Models: typically continuous time Markov chains
  - Emphasis on steady-state and transient probabilities

- In stochastic planning
  - Cf Bellman equations, ca 1950s
  - Models: Markov decision processes
  - Emphasis on finding optimum policies

- Our focus, probabilistic model checking
  - Distinctive, on automated verification for probabilistic systems
  - Temporal logic specifications, automata-theoretic techniques
  - Shared models
  - Exchanging techniques with the other two areas

Probabilistic models: discrete time

- Labelled transition systems
  - Discrete time steps
  - Labelling with atomic propositions



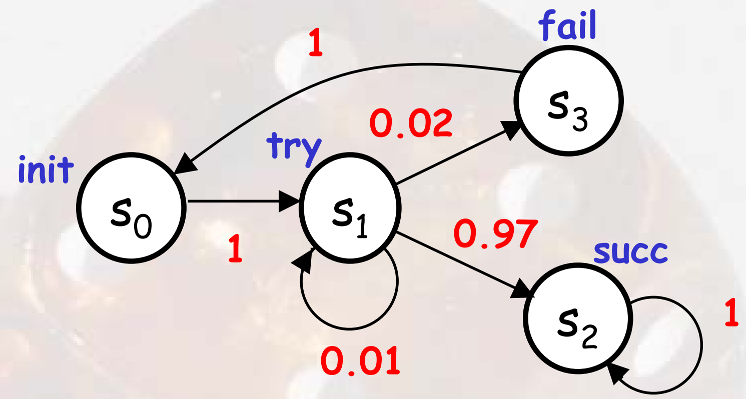
- Probabilistic transitions
  - Move to state with given probability
  - Represented as discrete probability distribution

$\sum_i p_i = 1$

- Model types
  - Discrete time Markov chains (DTMCs): probabilistic choice only
  - Markov decision processes (MDPs): probabilistic choice and nondeterminism

Discrete-Time Markov Chains (DTMCs)

- Features:
  - Only probabilistic choice in each state



- Formally, (S, s_0, P, L):
  - S finite set of states
  - s_0 initial state
  - P: S £ S ! [0,1] probability matrix, s.t. $\sum_{s'} P(s,s') = 1$, all s
  - L: S ! 2^{AP} atomic propositions

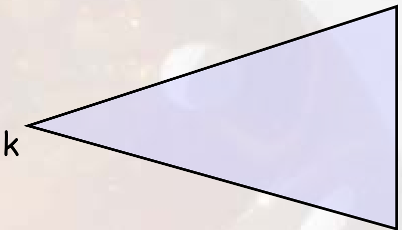
- Unfold into infinite paths $s_0 s_1 s_2 s_3 s_4 \ldots$ s.t. $P(s_i, s_{i+1}) > 0$, all i

- Probability for finite paths, multiply along path

  e.g. $s_0\, s_1\, s_1\, s_2$ is 1 ¢ 0.01 ¢ 0.97 = 0.0097

Probability space

- Intuitively:
  - Sample space = infinite paths $Path_s$ from s
  - Event = set of paths
  - Basic event = cone

$ss_1s_2 \ldots s_k$ 

- Formally, $(Path_s, \Omega, Pr)$
  - For finite path $\omega = ss_1 \ldots s_n$, define probability

$$P(\omega) = \begin{cases} 1 \text{ if } \omega \text{ has length one} \\ P(s,s_1) ¢ \ldots ¢ P(s_{n-1},s_n) \text{ otherwise} \end{cases}$$

  - Take Ω least σ-algebra containing cones

$$C(\omega) = \{ \pi \ 2 \ Path_s \mid \omega \text{ is prefix of } \pi \}$$

  - Define $Pr(C(\omega)) = P(\omega)$, all ω
  - Pr extends uniquely to measure on $Path_s$

The logic PCTL: syntax

- Probabilistic Computation Tree Logic [HJ94,BdA95,BK98]
  - For DTMCs/MDPs
  - New probabilistic operator, e.g. send $\rightarrow$ $\mathbf{P}_{\geq 0.9}(\Diamond \text{deliver})$
    "whenever a message is sent, the probability that it is eventually delivered is at least 0.9"

- The syntax of state and path formulas of PCTL is:

$$\phi ::= \text{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid \mathbf{P}_{\sim p}(\alpha)$$
$$\alpha ::= \mathrm{X}\,\phi \mid \phi\ \mathrm{U}\ \phi$$

where $p \in [0,1]$ is a probability bound and $\sim\ \in \{ <, >, \dots \}$

- Subsumes the qualitative variants [Var85,CY95] $\mathbf{P}_{=1}(\alpha)$, $\mathbf{P}_{>0}(\alpha)$

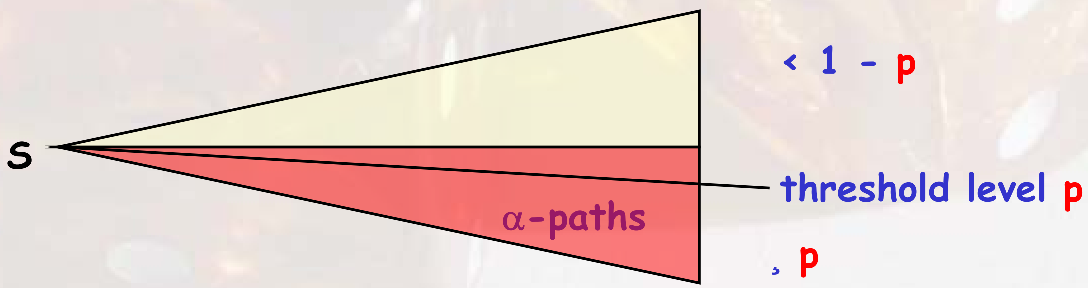
- Extension with cost/rewards and expectation operator $\mathbf{E}_{\sim c}(\phi)$

The logic PCTL: semantics

- Semantics is parameterised by a class of adversaries Adv
  - "under any scheduling, the probability bound is true at state s"
  - reasoning about worst-case/best-case scenario

- The probabilistic operator is a quantitative analogue of 8, 9

s $^2{}_{Adv}$ $\mathbf{P}_{» p}(\alpha)$, Pr^A { π 2 $Path^A{}_s$ j π $^2{}_{Adv}$ α } » p
for all A 2 Adv



PCTL semantics: summary

- Semantics of state formulas:

$s \ 2_{Adv} \ a \quad , \quad a \ 2 \ L(s)$

$s \ 2_{Adv} \ :\phi \quad , \quad s \ 2_{Adv} \ \phi$

$s \ 2_{Adv} \ \phi_1 \text{ Æ } \phi_2 \quad , \quad s \ 2_{Adv} \ \phi_1$ and $s \ 2_{Adv} \ \phi_2$

- Semantics of path formulas:

$\pi \ 2_{Adv} \ X \ \phi \quad , \quad \pi = s_0 \cdots$ and $s_1 \ 2_{Adv} \ \phi$

$\pi \ 2_{Adv} \ \phi_1 \ U \ \phi_2 \quad , \quad \pi = s_0 \cdots$ and $9 \ k$ s.t.
$s_k \ 2_{Adv} \ \phi_2$ and $8 \ j < k \ . \ s_j \ 2_{Adv} \ \phi_1$

- The probabilistic operator:

$s \ 2_{Adv} \ \mathbf{P}_{» p}(\alpha) \quad , \quad Pr^A \{ \pi \ 2 \ Path^A_s \ j \ \pi \ 2_{Adv} \ \alpha \} » p$
for all $A \ 2$ Adv

The logic PCTL: model checking

- By induction on structure of formula, as for CTL

- For the probabilistic operator and Until, solve
  - recursive linear equation for DTMCs
  - linear optimisation problem (form of Bellman equation) for MDPs
  - typically iterative solution methods

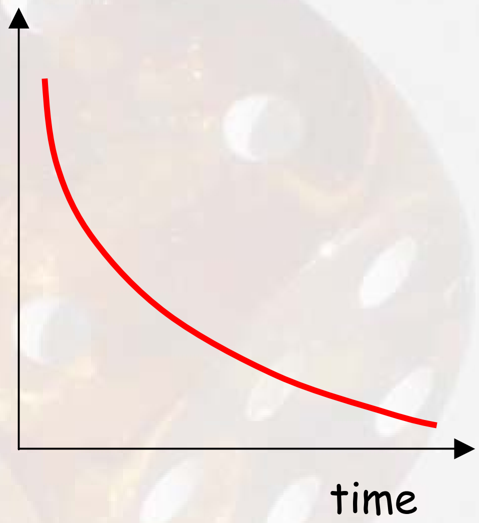
- Need to combine
  - conventional graph traversal
  - numerical linear algebra and linear optimisation (value iteration)

- Qualitative properties (probability 1, 0) proceed by graph traversal [Var85,dAKNP97]

Probabilistic models: continuous

- Assumptions on time and probability
  - Continuous passage of time
  - Continuous randomly distributed delays
  - Continuous space

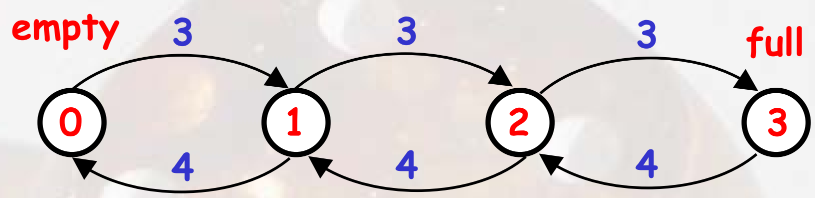
- Model types
  - Continuous time Markov chains (CTMCs): exponentially distributed delays, discrete space, no nondeterminism
  - Probabilistic Timed Automata (PTAs): dense time, (usually) discrete probability, admit nondeterminism
  - (not considered) Labelled Markov Processes (LMPs): continuous space/time, no nondeterminism



$$\int_0^{+\infty} f(x)dx = 1$$

Continuous Time Markov Chains (CTMCs)

- Features:
  - Discrete states and real time
  - Exponentially distributed random delays



- Formally:
  - Set of states S plus rates R(s,s') > 0 of moving from s to s'
  - Probability of moving from s to s' by time t > 0 is $1 - e^{-R(s,s')¢\ t}$
  - Transition rate matrix S £ S ! $R_{¸0}$

- Unfold into infinite paths $s_0 t_0 s_1 t_1 s_2 t_2 s_3 \ldots$
  - $prob_s$ (s'), probability of being in s' in the long-run, starting in s
  - $prob_s$ (s',t), probability of being in s' at time instant t

- But: no nondeterminism

The logic CSL: syntax

- Continuous Stochastic Logic [ASSB96,BKH99]
  - For CTMCs, based on PCTL, for example
    - $P_{< 0.85}(\}^{<15}$ full), probability operator
      "the probability of queue becoming full within 15 secs is < 0.85"
    - $S_{< 0.01}$(down), steady-state operator
      "in the long run, the probability the system is down is less than 1%"

- The syntax of state and path formulas of CSL is:

$$\phi ::= \text{true} \mid a \mid \phi \text{ Æ } \phi \mid :\phi \mid \mathbf{S}_{» p}(\phi) \mid \mathbf{P}_{» p}(\alpha)$$
$$\alpha ::= X\, \phi \mid \phi\, U^{\cdot t}\, \phi \mid \phi\, U\, \phi$$

where p 2 [0,1] is a probability bound, $t \ 2\ R_{¸0}$ and » 2 { <, >, … }

- Extension with time intervals for until, cost/rewards and expectation operator $\mathbf{E}_{» c}(\phi)$

CSL semantics

- Semantics of bounded until:

  π ² ϕ_1 $U^{\cdot\, t}$ ϕ_2 , iff ϕ_2 satisfied at time instant t along $\pi = s_0 \cdots$ and ϕ_1 satisfied at all preceding time instants

- The added operators:

  s ² $\mathbf{S}_{»\, p}(\phi)$, $\Sigma_{s' \text{ ² } \phi}$ prob_s (s') » p
  where prob_s (s') is prob. of being in s' in the long-run, having started in s

  s ² $\mathbf{P}_{»\, p}(\alpha)$, Pr { π 2 Path_s j π ² α } » p
  where Pr is probability measure on paths as for PCTL

- Semantics of remaining formulas as for PCTL

The logic CSL: model checking

- By induction on structure of formula, as for PCTL except for
  - $\mathbf{S}_{» p}(\phi)$ and $\mathbf{P}_{» p}(\phi_1\ U^{· t}\ \phi_2)$
- The steady-state operator
  - Requires computation of steady-state probabilities
  - Reduces to graph traversal and (iterative) solution of linear equation system
- The time-bounded until
  - Reduces to transient analysis
  - Transform CTMC by removing all outgoing transitions from states satisfying ϕ_2 or $:\phi_1$
  - Then $Pr\ \{\ \pi\ 2\ Path_s\ j\ \pi\ ²\ \phi\ U^{·\ t}\ \phi\ \} = \Sigma_{s'\ ²\ \phi 2}\ prob_s\ (s',t)$
  - Computed by using uniformisation
  - More efficient and stable, iterative computation

The PRISM project

- Approach
  - Based on symbolic, BDD-based techniques
  - Multi-Terminal BDDs, first algorithm [ICALP'97]
  - Hybrid combination of symbolic and explicit vector representation, efficient for CTMCs
- History
  - First public release September 2001, ~7 years development
  - Substantial improvements to functionality, efficiency and model size capability (> 10^{10} for CTMCs, higher for other models)
- Funding
  - EPSRC, DTI, QinetiQ
  - Current ongoing projects on compositionality, mobility extension and parallelisation

The PRISM tool: overview

- Functionality
  - Implements temporal logic probabilistic model checking
  - Construction of models: discrete and continuous Markov chains (DTMCs/CTMCs), and Markov decision processes (MDPs)
  - Modelling language: probabilistic guarded commands
    - Corresponds to sublanguage of stochastic pi-calculus
  - Probabilistic temporal logics: PCTL and CSL
  - Extension with costs/rewards, expectation operator

- Underlying computation combines graph-theoretical algorithms
  - Reachability, qualitative model checking, BDD-based

  with numerical computation – iterative methods
  - Linear equation system solution - Jacobi, Gauss-Seidel, ...
  - Uniformisation (CTMCs)
  - Dynamic programming (MDPs)
  - Explicit and symbolic (MTBDDs, etc.)

PRISM real-world case studies

- MDPs/DTMCs
  - Bluetooth device discovery [ISOLA'04]
  - Crowds anonymity protocol (by Shmatikov) [JSC 2003]
  - Randomised consensus [CAV'01,FORTE'02]
  - NAND multiplexing for nanotechnology (with Shukla) [VLSI'04]
- CTMCs
  - Molecular reactions (based on Shapiro)
  - Eukaryotic cell cycle control (based on Lecca & Priami)
  - Dependability of embedded controller [INCOM'04]
- PTAs
  - IPv4 Zeroconf dynamic configuration [FORMATS'03]
  - Root contention in IEEE 1394 FireWire [FAC 2003, STTT 2004]
  - IEEE 802.11 (WiFi) Wireless LAN MAC protocol [PROBMIV'02]

PRISM technicalities

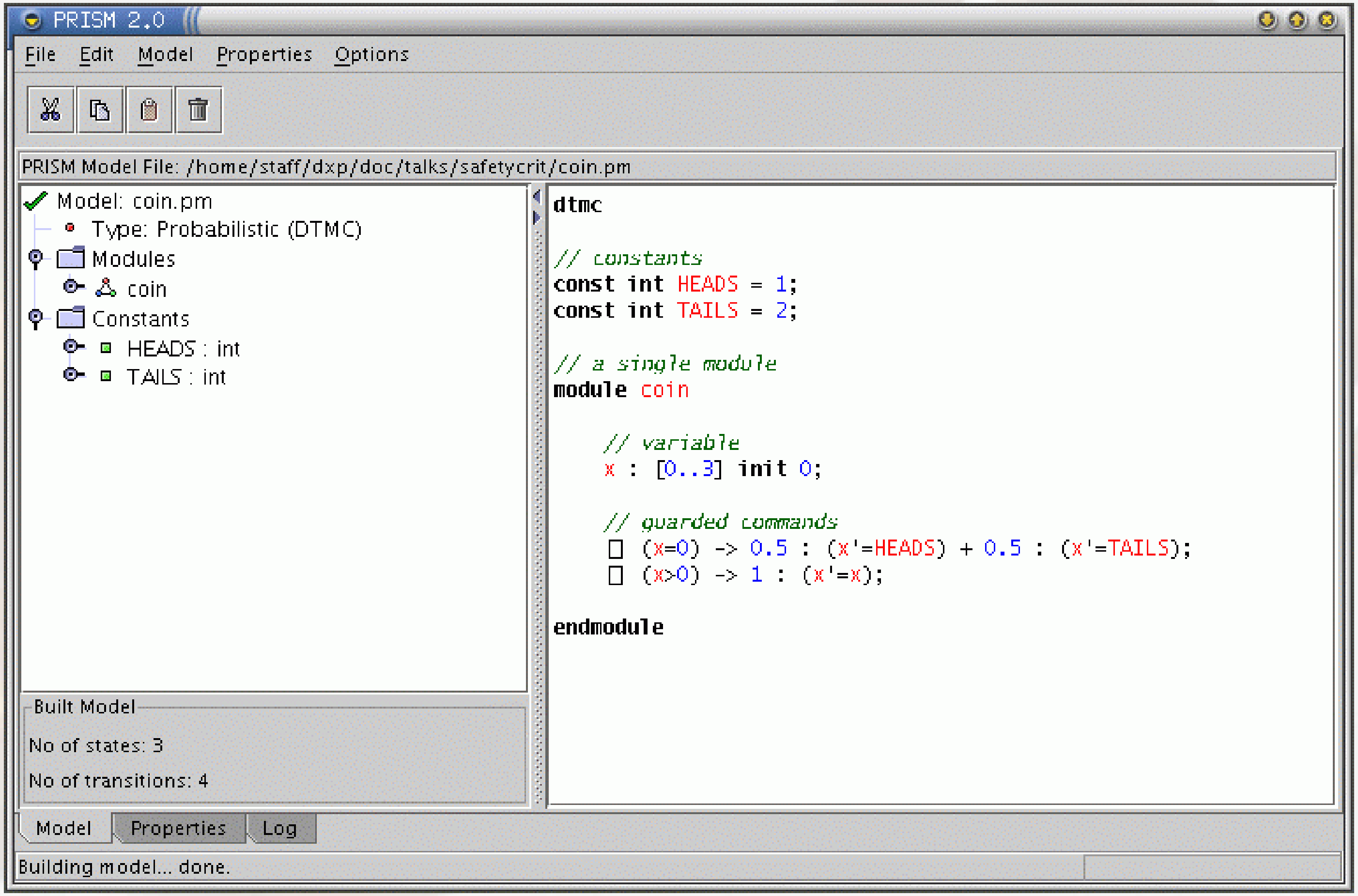
- Properties in logic CSL
  - P=? [true U A], probability of A eventually occurring?
  - P=? [true U<=T A], probability of A occurring by time T?
  - S=? [A], probability that A is true in the long-run?

- Augment states and transitions with real-valued rewards
  - Instantaneous rewards, e.g. "concentration of reactant"
  - Cumulative rewards, state- and transition-based, e.g. "power consumed", "messages lost"

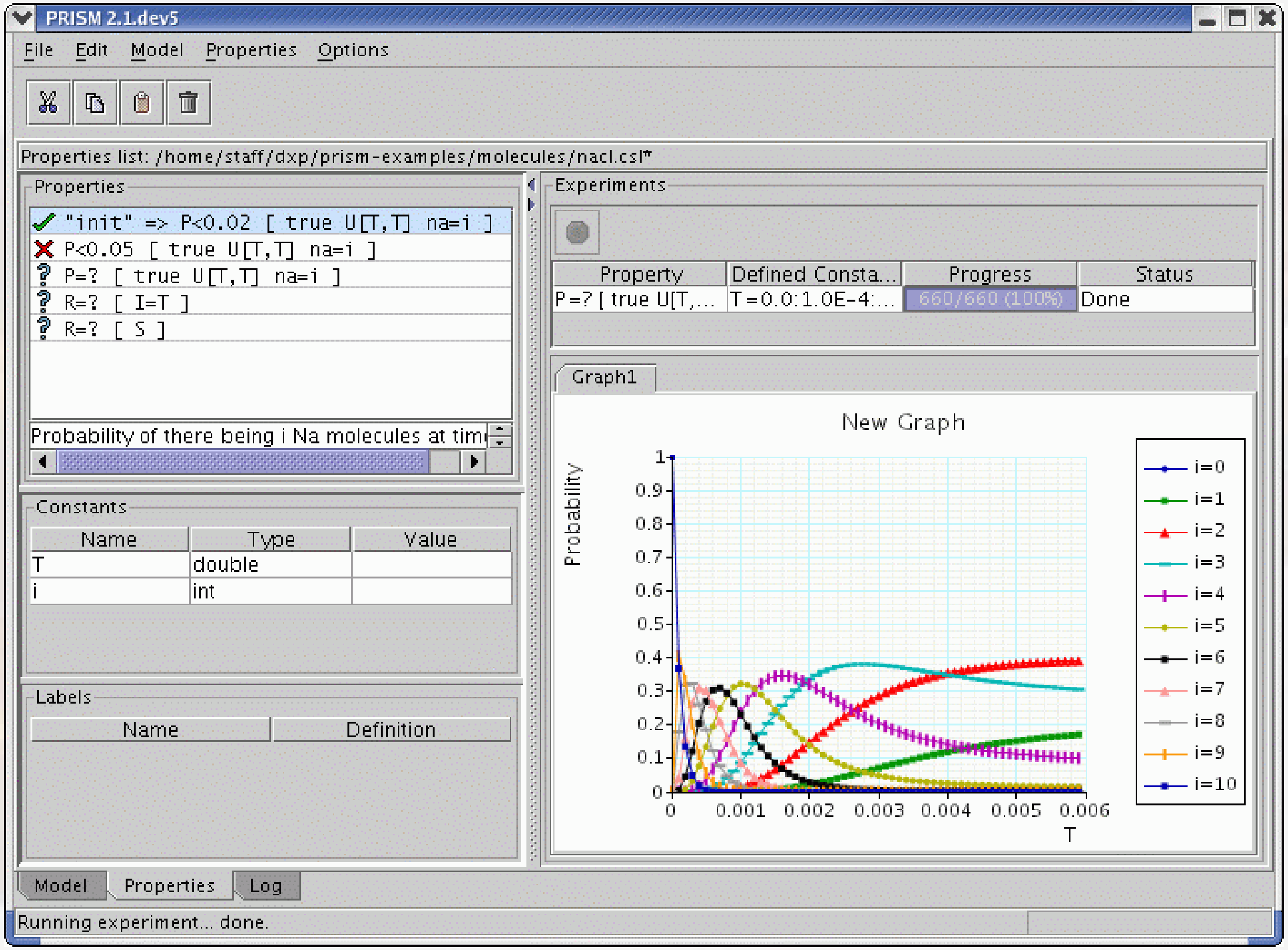
- Support for "experiments"
  - e.g. P=? [true U<=T error] for N=1..5,T=1..100

- GUI implementation
  - integrated editor for PRISM language
  - automatic graph plotting

Screenshot: Text editor

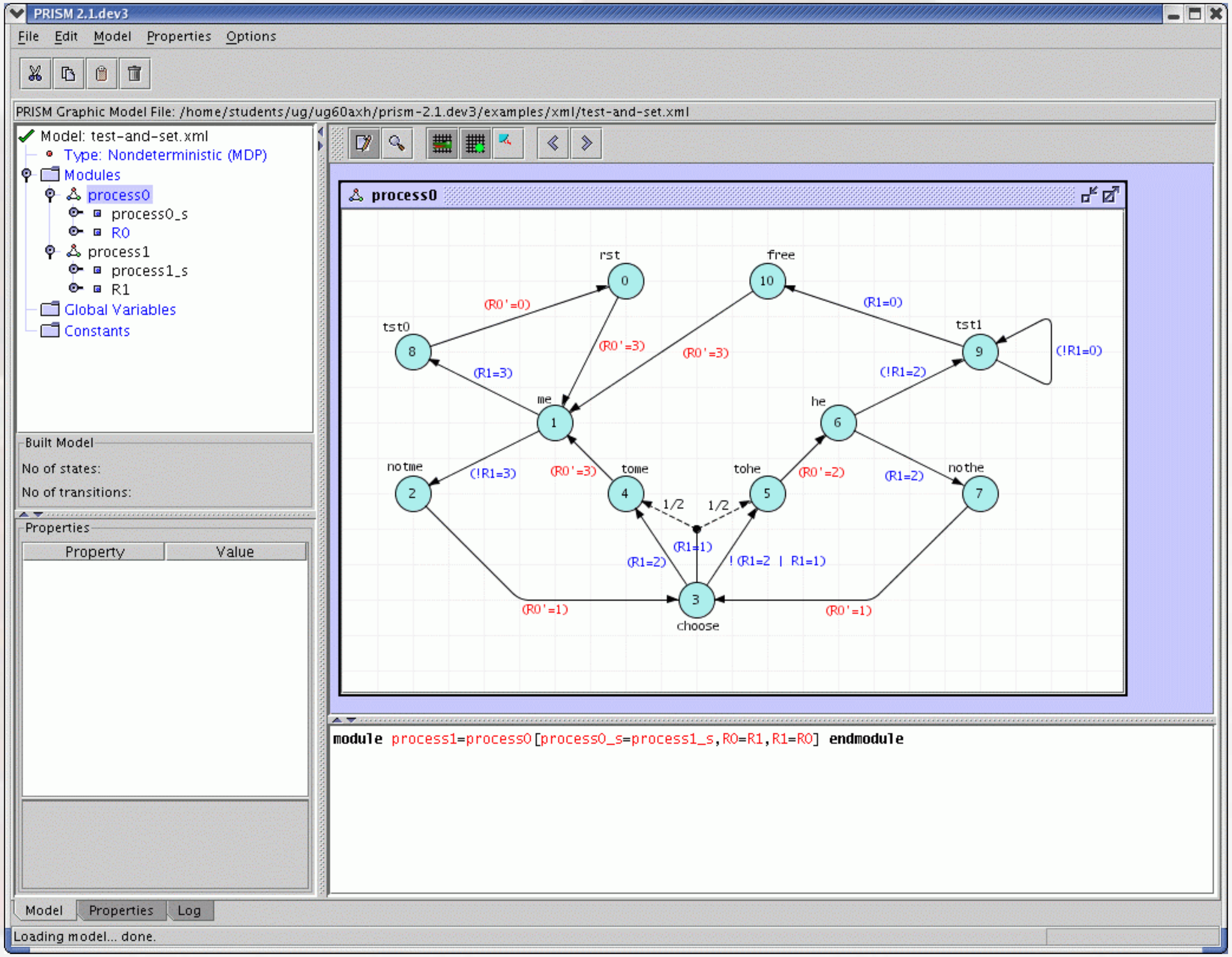


Screenshot: Graphs



Ongoing developments

- Graphical modelling language



Case Study: Molecular Reactions

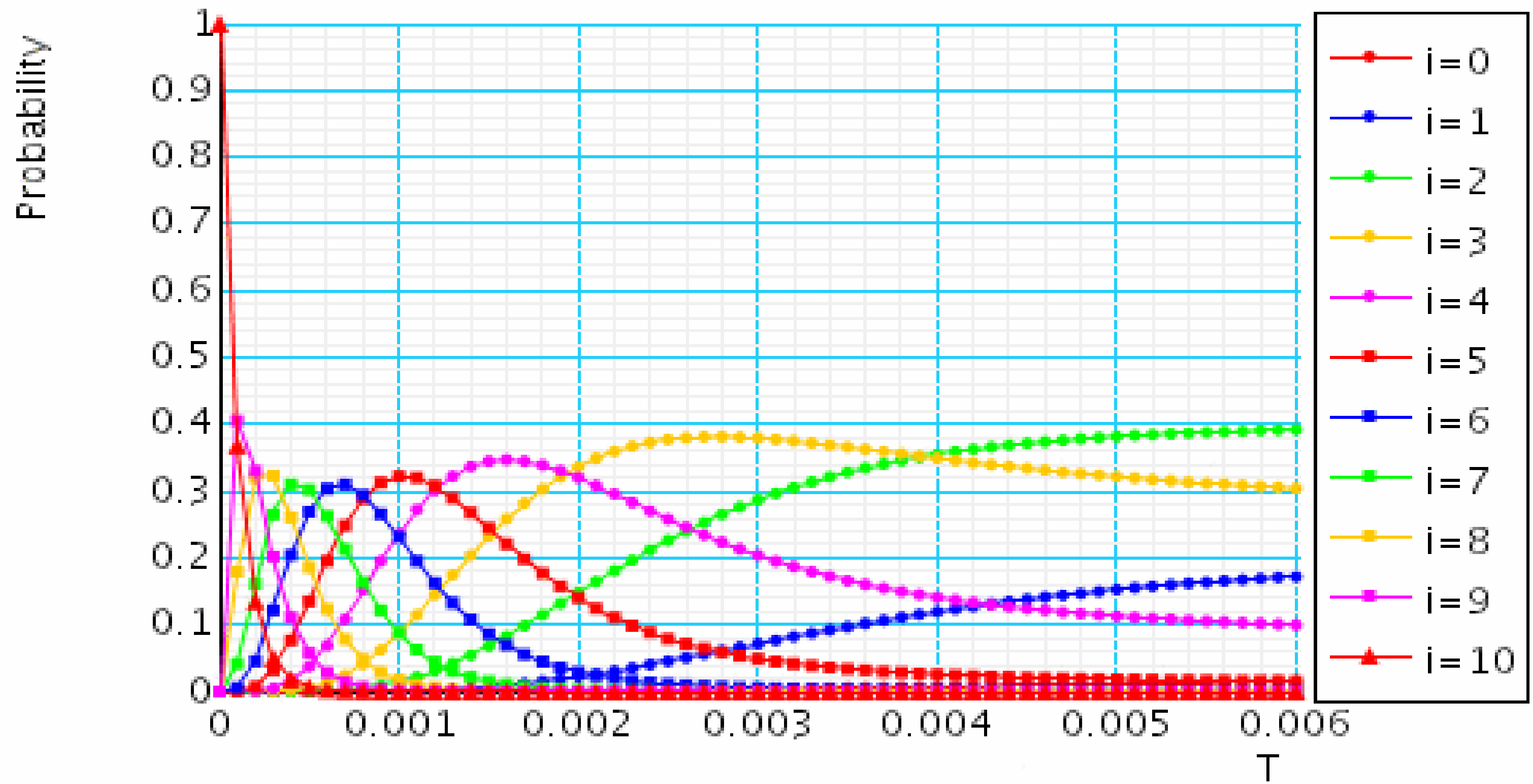
- Time until a reaction occurs is given by an exponential distribution [Gillespie 1977]
  - model reactions using continuous time Markov chains

- Rate of reaction determined by:
  - base rate (empirically determined constant)
  - concentration of reactants (number of each type of molecule that takes part in the reaction)

- This case study: **$Na + Cl \leftrightarrow Na^+ + Cl^-$**
  - forward base rate 100
  - backwards base rate 10
  - initially N1 Na molecules and N2 Cl molecules

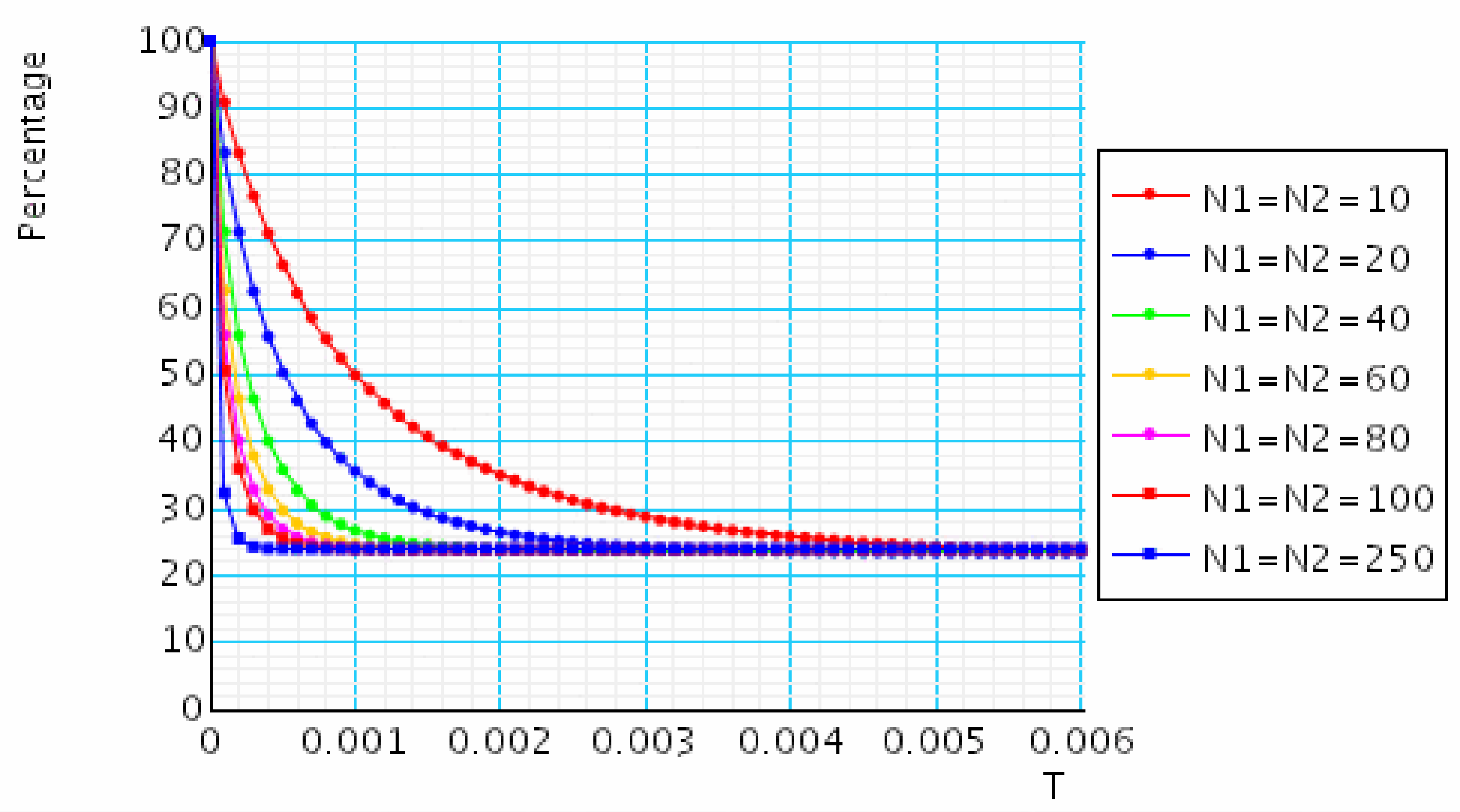
Results: Molecular Reactions

- $P_{=?}$ (true $U^{[T,T]}$ Na=i) 'probability i Na molecules at time T'



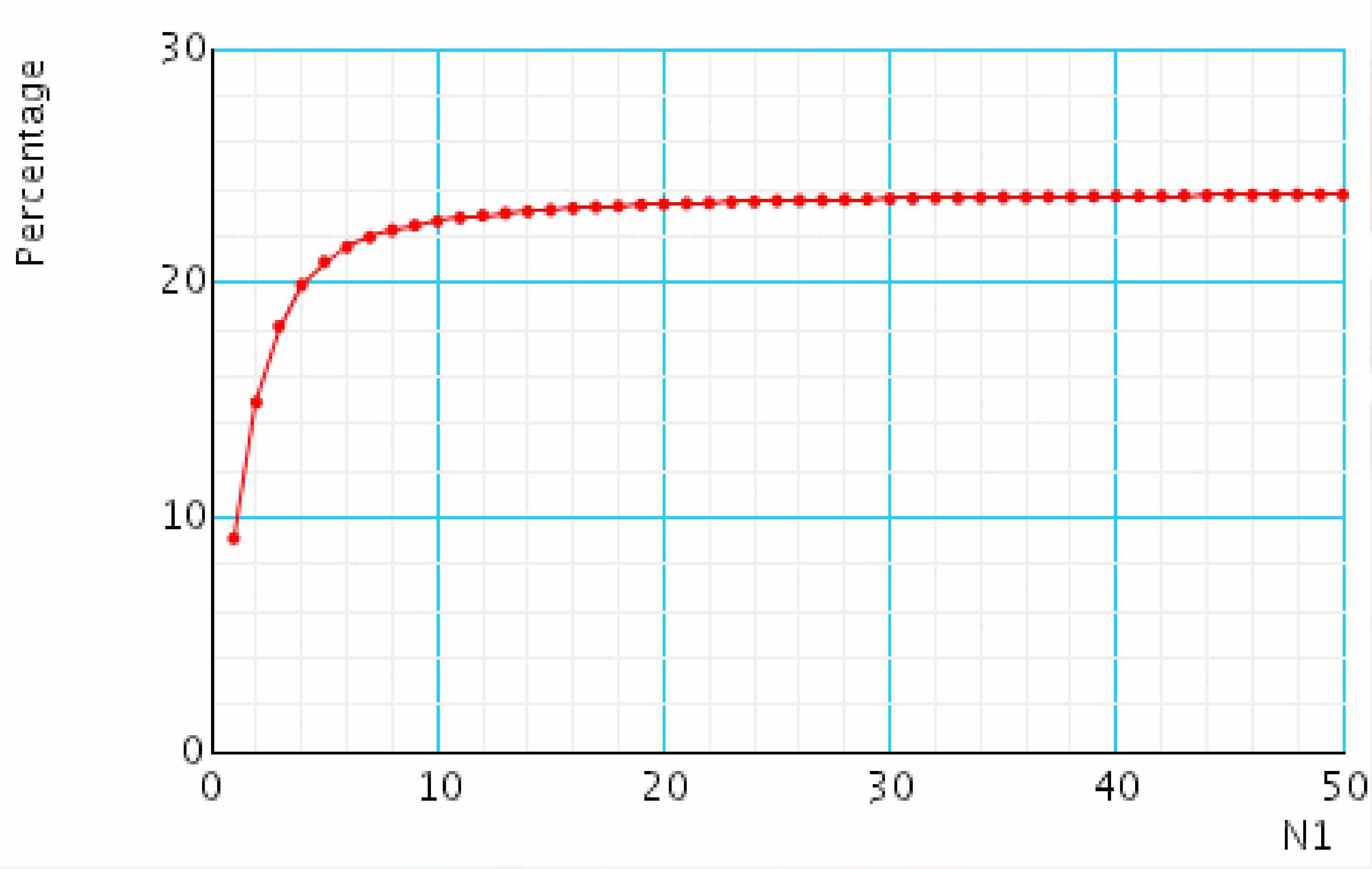
Results: Molecular Reactions

- $R_{=?}$ (I=T) 'expected percentage of Na molecules at time T'



Results: Molecular Reactions

- $\mathbf{R}_{=?}$ (S) 'expected percentage of Na molecules in the long run'



Case study: IPv4 Zeroconf protocol

- IPv4 ZeroConf protocol [Cheshire,Adoba,Guttman'02]
  - New IETF standard for dynamic network self-configuration
  - Link-local (no routers within the interface)
  - No need for an active DHCP server
  - Aimed at home networks, wireless ad-hoc networks, hand-held devices
  - "Plug and play"

- Self-configuration
  - Performs assignment of IP addresses
  - Symmetric, distributed protocol
  - Uses random choice and timing delays

IPv4 Zeroconf Standard

- Select an IP address out of 65024 at random
- Send a probe querying if address in use, and listen for 2 seconds
  - If positive reply received, restart
  - Otherwise, continue sending probes and listening (2 seconds)
- If K probes sent with no reply, start using the IP number
  - Send 2 packets, at 2 second intervals, asserting IP address is being used
  - If a conflicting assertion received, either:
    - defend (send another asserting packet)
    - defer (stop using the IP address and restart)

Will it work?

- Possible problem…
  - IP number chosen may be already in use, but:
    - Probes or replies may get lost or delayed (host too busy)
- Issues:
  - Self-configuration delays may become unacceptable
    - Would you wait 8 seconds to self-configure your PDA?
  - No justification for parameters
    - for example K=4 in the standard
- Case studies:
  - DTMC and Markov reward models, analytical [BvdSHV03,AK03]
  - TA model using UPPAAL [ZV02]
  - PTA model with digital clocks using PRISM [KNS03]

The IPv4 Zeroconf protocol model

- Modelled using Probabilistic Timed Automata (with digital clocks)

- Parallel composition of two PTAs:
  - one (joining) host, modelled in detail
  - environment (communication medium + other hosts)

- Variables:
  - K (number of probes sent before the IP address is used)
  - the probability of message loss
  - the number of other hosts already in the network

Expected costs

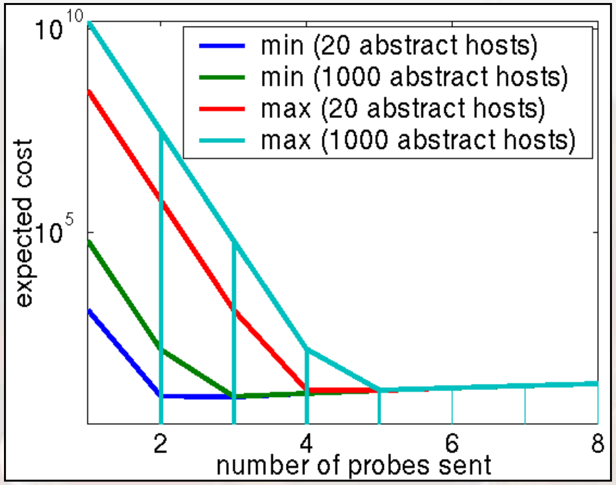
- Compute minimum/maximum expected cost accumulated before obtaining a valid IP address?
  - Implement algorithms of [de Alfaro97] (stochastic shortest path problems for finite-state MDPs)

- Costs:
  - Time should be costly: the host should obtain a valid IP address as soon as possible
  - Using an IP address that is already in use should be very costly: minimise probability of error

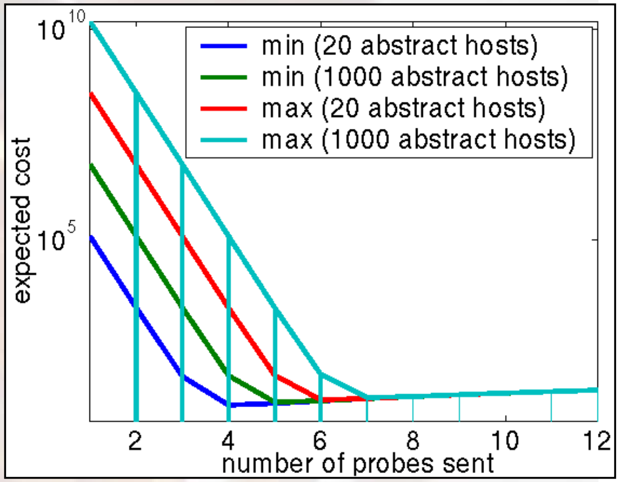
- Cost pair: (r,e)
  - $r=1$ (t time units elapsing corresponds to a cost of t)
  - $e=10^{12}$ for the event corresponding to using an address which is already in use
  - $e=0$ for all other events

Results for IPv4 Zeroconf

Prob. of message loss = 0.001



Prob. of message loss = 0.01

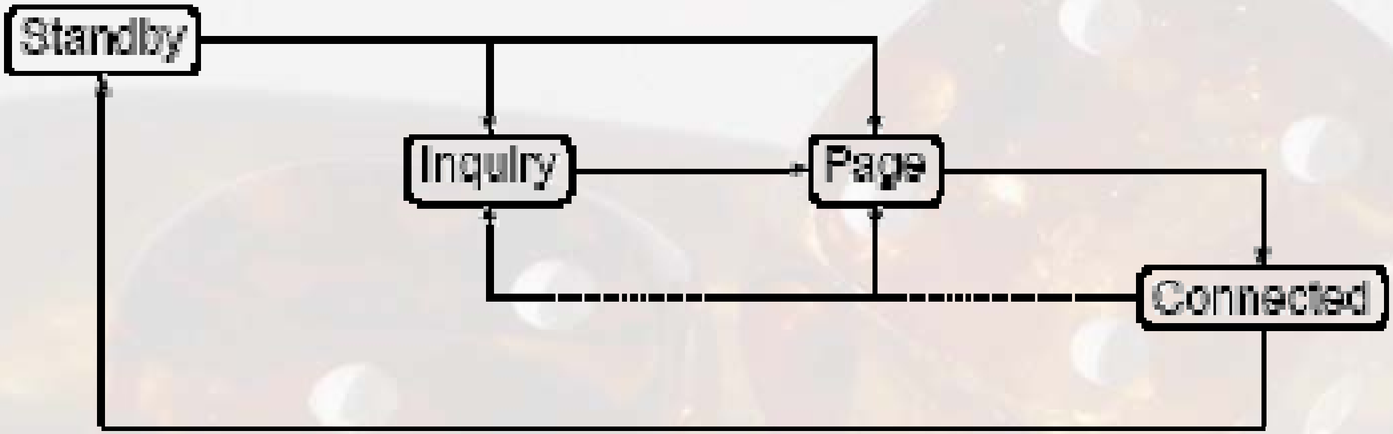


- Sending a high number of probes increases the cost
  - increases delay before a fresh IP address can be used
- Sending a low number of probes increases the cost
  - increases probability of using an IP address already in use
- Similar results to the simpler model of [BvdSHV03]

Case Study: Bluetooth protocol

- Short-range low-power wireless protocol
  - Personal Area Networks (PANs)
  - Open standard, versions 1.1 and 1.2
  - Widely available in phones, PDAs, laptops, …
- Uses frequency hopping scheme
  - To avoid interference (uses unregulated 2.4GHz band)
  - Pseudo-random frequency selection over 32 of 79 frequencies
  - Inquirer hops faster
  - Must synchronise hopping frequencies
- Network formation
  - Piconets (1 master, up to 7 slaves)
  - Self-configuring: devices discover themselves
  - Master-slave roles

States of a Bluetooth device



- Master looks for device, slave listens for master
- Standby: default operational state
- Inquiry: device discovery
- Page: establishes connection
- Connected: device ready to communicate in a piconet

Why focus on device discovery?

- Performance of device discovery crucial
  - No communication before initialisation
  - First mandatory step: device discovery
- Device discovery
  - Exchanges information about slave clock times, which can be used in later stages
  - Has considerably higher power consumption
  - Determines the speed of piconet formation

Frequency hopping

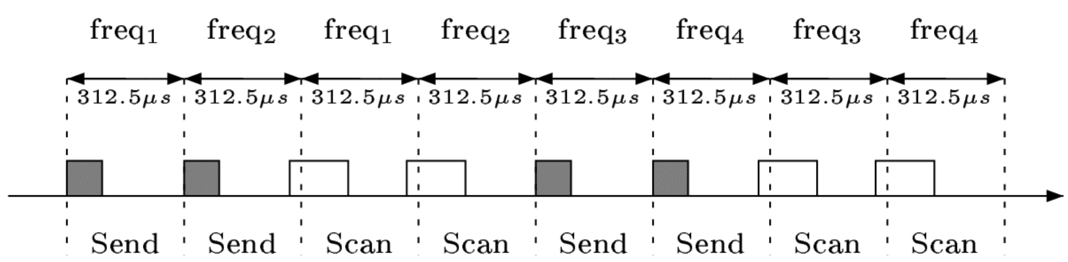


Fig. 1. Timing of the inquiring device's behaviour

- Clock CLK, 28 bit free-running, ticks every 312.5μs
- Inquiring device (master) broadcasts inquiry packets on two consecutive frequencies, then listens on the same two (plus margin)
- Potential slaves want to be discovered, scan for messages
- Frequency sequence determined by formula, dependent on bits of clock CLK (k defined on next slide):

$$freq = [CLK_{16\text{-}12}+k+ (CLK_{4\text{-}2,0}-CLK_{16\text{-}12}) \bmod 16] \bmod 32$$

Frequency hopping sequence

$freq = [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \bmod 16] \bmod 32$

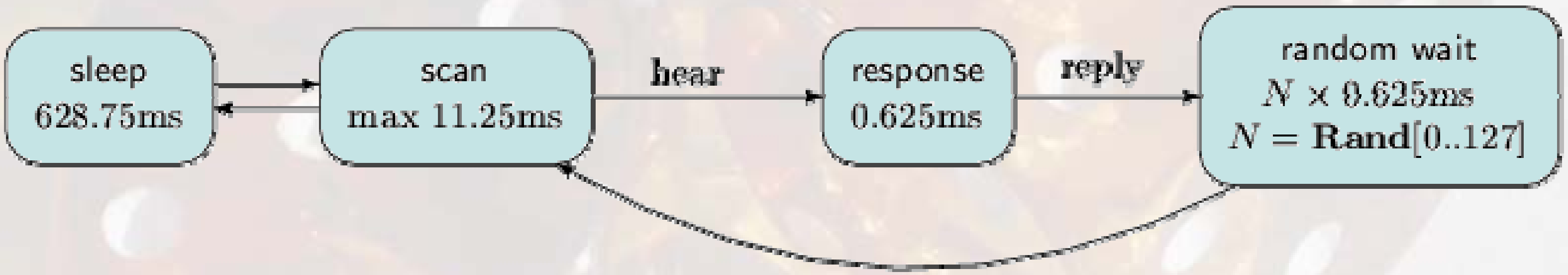
- Two trains (=lines)
- k is offset that determines which train
- Swaps between trains every 2.56 sec
- Each line repeated 128 times

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	16

Sending and receiving in Bluetooth

- Sender: broadcasts inquiry packets, sending according to the frequency hopping sequence, then listens, and repeats

- Receiver: follows the frequency hopping sequence, own clock



- Listens continuously on one frequency
- If hears message sent by the sender, then replies on the same frequency
- Random wait to avoid collision if two receivers hear on same frequency

Bluetooth modelling

- Very complex interaction
  - Genuine randomness, probabilistic modelling essential
  - Devices make contact only if listen on the right frequency at the right time!
  - Sleep/scan periods unbreakable, much longer than listening
  - Cannot scale constants (approximate results)
  - Cannot omit subactivities, otherwise oversimplification
- Huge model, even for one sender and one receiver!
  - Initial configurations dependent on 28 bit clock
  - Cannot fix start state of receiver, clock value could be arbitrary
  - 17,179,869,184 possible initial states
- But is a realistic future ubiquitous computing scenario!

What about other approaches?

- Indeed, others have tried…
  - network simulation tools (BlueHoc)
  - analytical approaches

- But
  - simulations obtain averaged results, in contrast to best/worst case analysis performed here
  - analytical approaches require simplifications to the model
  - it is easy to make incorrect probabilistic assumptions, as we can demonstrate

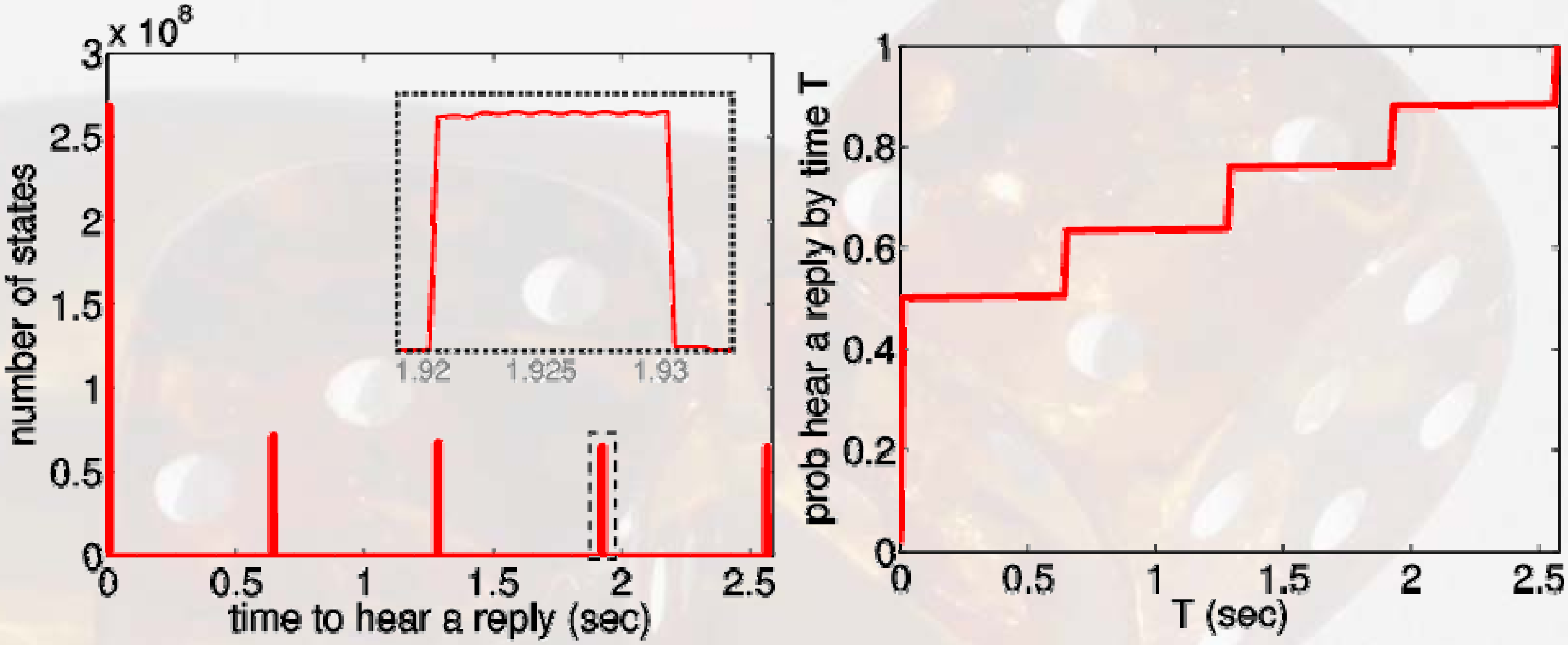
- There is a case for all types of analyses, or their combinations…

Lessons learnt…

- Must optimise/reduce model
  - Assume negligible clock drift
  - Discrete time, obtain a DTMC
  - Manual abstractions, combine transitions, etc
  - Divide into 32 separate cases
  - Success (exhaustive analysis) with one/two replies

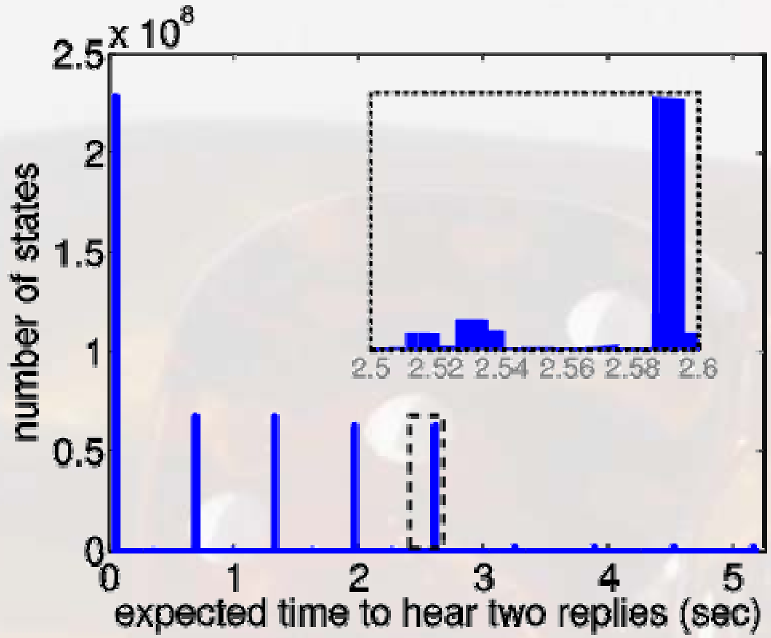
- Observations
  - Work with realistic constants, as in the standard
  - Analyse v1.2 and 1.1, confirm 1.1 slower
  - Show best/worst case values, can pinpoint scenarios which give rise to them
  - Also obtain power consumption analysis

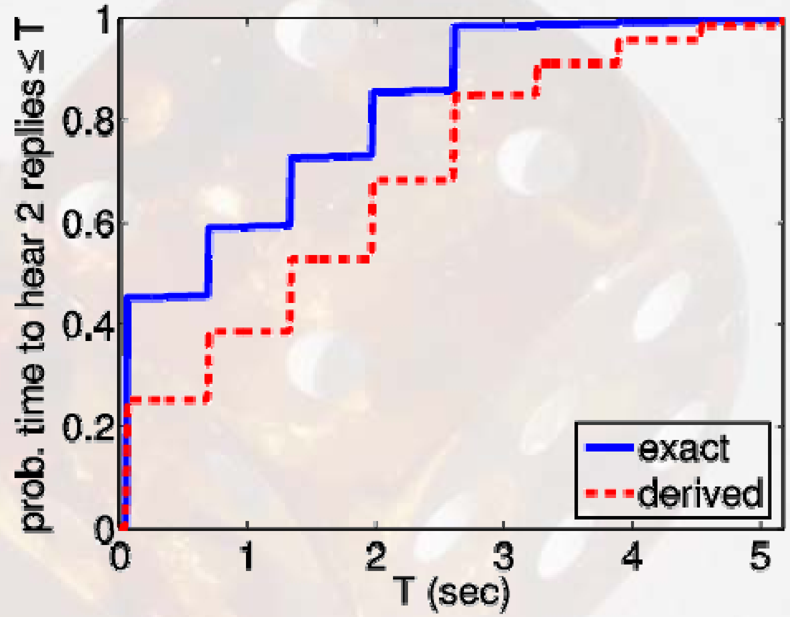
Time to hear 1 reply



- Max time to hear is 2.5716sec, in 921,600 possible initial states, (Min 635μs)
- Cumulative: assume uniform distribution on states when receiver first starts to listen

Bluetooth: verification vs simulation





Huge probabilistic model, 17,179,869,184 possible initial states.
Unlike simulation, model checking is exhaustive.
The exact curve is obtained by model checking.
Derived plot incorrectly assumes independence of events.

Successes so far

- Fully automatic, no expert knowledge needed for
  - Probabilistic reachability and temporal logic properties
  - Expected time/cost
- Tangible results!
  - 6 cases of "unusual behaviour" found, over 30 case studies
  - Greater level of detail, may expose obscure dependencies
- PRISM tool robust
  - Simple model description language
  - Broad class of models
  - Large, realistic models often possible
  - Flexible property language
  - Choice of engines

But…

- Models monolithic and finite-state only
  - Emphasis on efficiency
  - No decomposition, abstraction
  - No data reduction

- State-space explosion has not gone away…
  - Heuristics for MTBDDs/BDDs sometimes fail
  - Parallelise? Disk-based?

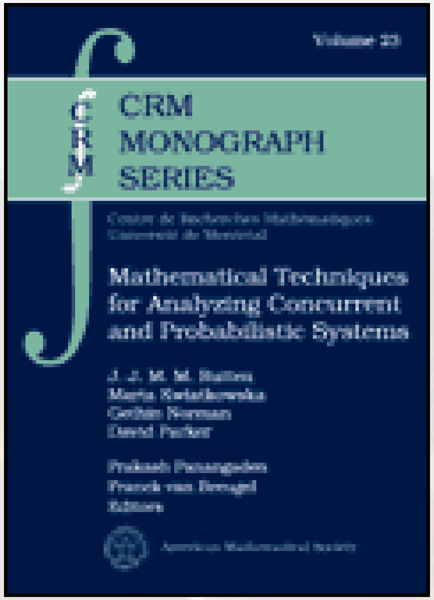
- Limited expressiveness
  - Only PCTL plus extensions (LTL in progress)
  - Only exponential distributions
  - No direct support for PTAs (work in progress, [FORMATS'04])
  - No continuous space models
  - No mobility

Challenges for future

- Exploiting structure
  - Abstraction, data/equivalence quotient, (de)compositionality...
  - Parametric probabilistic verification?
- Proof assistant for probabilistic verification?
- Approximation methods?
- Efficient methods for continuous models
  - Continuous PTAs? Continuous time MDPs? LMPs?
- More expressive specifications
  - Probabilistic LTL/PCTL*/mu-calculus?
- Real software, not models!

- More applications
  - Quantum cryptographic protocols
  - Mobile ad hoc network protocols

For more information…



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
March 2004

www.cs.bham.ac.uk/~dxp/prism/

- Case studies, statistics, group publications
- Download, version 2.1 (2000 downloads)
- Unix/Linux, Windows, Apple platforms
- Publications by others and courses that feature PRISM…

PRISM collaborators worldwide