Modelling and verification of probabilistic systems

Marta Kwiatkowska
School of Computer Science

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

Lucent, 10th November 2004
Overview

- **Motivation**

- **Probabilistic model checking**
  - The models
  - Specification languages
  - What does it involve?
  - The PRISM model checker

- **Case studies**
  - Self-stabilisation
  - Dynamic power management
  - IPv4 Zeroconf dynamic configuration protocol
  - Root contention in IEEE 1394 FireWire

- **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, ...

Correct design a challenge for formal methods?
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

Model Checker

send → ◊ deliver

Temporal logic specification

Also refinement checking, equivalence checking, ...

Line 5: ...
Line 21: ...
Line 15: ...
...
Line 27: ...
Line 45: ...
Probabilistic model checking...

in a nutshell

Probabilistic model

Probabilistic Model Checker

The probability

send $\rightarrow P_{0.9}(\Diamond \text{deliver})$

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

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

\[ \sum_{i} p_i = 1 \]
Theory timeline: discrete models

Qualitative (with probability 1 or 0)
- 1983 Hart-Sharir-Pnueli
- 1985 Vardi
- 1988 Courcoubetis-Yannakakis

Quantitative (with arbitrary probability)
- 1991 Larsen-Skou (probab. bisimulation)
- 1994 Hansson-Jonsson (DTMC model checking)
- 1995 Bianco-de Alfaro (MDP model checking)
- 1995 Segala-Lynch (probab. simulation)
- 1997 Huth-Kwiatkowska [LICS] (probab. mu-calculus)
- 1997 Baier et al (DTMC model checking)
- 1998 Baier-Kwiatkowska (MDPs + fairness)
- 1999 Kwiatkowska-Norman-Segala-Sproston (PTAs)
- 2001 Kwiatkowska-Norman-Sproston (infinite state)
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
  - \(P: S \rightarrow S \rightarrow [0,1]\) probability matrix, s.t. \(\sum_{s'} P(s, s') = 1\), all \(s\)
  - \(L: S \rightarrow 2^{AP}\) 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 \(1 \cdot 0.01 \cdot 0.97 = 0.0097\)
**Probability space**

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

- **Formally, (Path_s, Ω, Pr)**
  - For finite path ω = ss_1...s_n, define probability
    \[
    P(ω) = \begin{cases} 
    1 & \text{if } ω \text{ has length one} \\
    P(s, s_1) \land ... \land P(s_{n-1}, s_n) & \text{otherwise}
    \end{cases}
    \]
  - Take Ω least σ-algebra containing cones
    \[
    C(ω) = \{ π : \text{2 Path_s} \mid ω \text{ is prefix of } π \}
    \]
  - Define \( Pr(C(ω)) = P(ω) \), all ω
  - \( Pr \) extends uniquely to measure on Path_s
Markov Decision Processes (MDPs)

- **Features:**
  - Nondeterministic choice
  - Parallel composition of DTMCs

- Formally, \((S, s_0, \text{Steps}, L)\):
  - \(S\) finite set of states
  - \(s_0\) initial state
  - \(\text{Steps}\) maps states \(s\) to sets of probability distributions \(\mu\) over \(S\)
  - \(L: S \rightarrow 2^{AP}\) atomic propositions

- Unfold into infinite paths \(s_0 \mu_0 s_1 \mu_1 s_2 \mu_2 s_3 \ldots\) s.t. \(\mu_i(s_i, s_{i+1}) > 0\), all \(i\)

- Probability space induced on \(\text{Paths}_s\) by adversary (policy) \(A\) mapping finite path \(s_0 \mu_0 s_1 \mu_1 \ldots s_n\) to a distribution from state \(s_n\)
The logic PCTL: syntax

- **Probabilistic Computation Tree Logic** [HJ94,BdA95,BK98]
  - For DTMCs/MDPs
  - New probabilistic operator, e.g. \( send \rightarrow P_{0.9}(\Diamond \text{deliver}) \)
    “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:

\[
\phi ::= \text{true} \mid a \mid \phi \land \phi \mid :\phi \mid P_{p}(\alpha)
\]
\[
\alpha ::= X \phi \mid \phi U \phi
\]

where \( p \in [0,1] \) is a probability bound and \( \in \{<, >, \ldots \} \)

- Subsumes the qualitative variants [Var85,CY95] \( P_{\geq 1}(\alpha) \), \( P_{> 0}(\alpha) \)
- Extension with cost/rewards and expectation operator \( E_{c}(\phi) \)
Semantics is parameterised by a class of adversaries $\text{Adv}$
- “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

$$s \overset{2}{\text{Adv}} P \overset{p(\alpha)}{\to} \Pr^A \{ \pi \text{2 Path}^A_s \mid \pi \overset{2}{\text{Adv}} \alpha \} \to p$$
for all $A \in \text{Adv}$
PCTL semantics: summary

• Semantics of state formulas:
  \[ s^{2_{Adv}} a \] , \[ a \ 2 \ L(s) \]
  \[ s^{2_{Adv}} \phi \] , \[ s^{2_{Adv}} \phi \]
  \[ s^{2_{Adv}} \phi_1 \lor \phi_2 \] , \[ s^{2_{Adv}} \phi_1 \text{ and } s^{2_{Adv}} \phi_2 \]

• Semantics of path formulas:
  \[ \pi^{2_{Adv}} X \phi \] , \[ \pi = s_0 \ldots \text{ and } s_1^{2_{Adv}} \phi \]
  \[ \pi^{2_{Adv}} \phi_1 \lor \phi_2 \] , \[ \pi = s_0 \ldots \text{ and } 9 \ k \ \text{s.t.} \]
  \[ s_k^{2_{Adv}} \phi_2 \text{ and } 8 \ j < k \ . s_j^{2_{Adv}} \phi_1 \]

• The probabilistic operator:
  \[ s^{2_{Adv}} P \succ_p (\alpha) \] , \[ Pr^A \{ \pi \ 2 \ Path^A s_j \ pi^{2_{Adv}} \alpha \} \succ_p \]
  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]
PCTL model checking for DTMCs

- By induction on structure of formula
- For the probabilistic operator
  \[- \text{Sat}( P \cdot P(X \phi) ) , \{ s \in S | \sum_{s'} \text{Sat}(\phi) P(s,s') \geq p \} \]
  \[- \text{Sat}( P \cdot P(\phi_1 U \phi_2) ) , \{ s \in S | x_s \geq p \} \]

where $x_s$, $s \in S$, are obtained from the recursive linear equation

$$x_s = \begin{cases} 
0 & \text{if } s \in S^{no} \\
1 & \text{if } s \in S^{yes} \\
\sum_{s'} P(s,s') x_{s'} & \text{if } s \in S^{no} \{ S^{yes} \} 
\end{cases}$$

and

- $S^{yes}$ - states that satisfy $\phi_1 U \phi_2$ with probability exactly 1
- $S^{no}$ - states that satisfy $\phi_1 U \phi_2$ with probability exactly 0
PCTL model checking for DTMCs

- For the remaining formulas standard:

\[
\begin{align*}
    \text{Sat}(a) &= L(a) \\
    \text{Sat}(\phi) &= S\setminus\text{Sat}(\phi) \\
    \text{Sat}(\phi_1 \land \phi_2) &= \text{Sat}(\phi_1) \setminus \text{Sat}(\phi_2)
\end{align*}
\]

- \text{Syes}, \text{Sno} can be precomputed by graph traversal [Var85] (or BDD fixed point computation)

- Need to combine
  - Conventional graph-theoretic traversal
  - Numerical linear algebra
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

\[ \int_{s_0}^{s_1} f(x) \, dx = 1 \]
Theory timeline: continuous models

**Continuous distributions**
- **1991** Alur-Courcoubetis-Dill (GSMPs)
- **1996** Aziz-Sanwal-Singhal-Brayton (logic CSL)
- **1998** de Alfaro (long-run average)
- **1999** Baier, Katoen, Hermanns (CTMC model checking)
- **2000** Baier, Haverkort, Hermanns, Katoen (uniformis.)
- **2000** Kwiatkowska-Norman-Segala-Sproston (cont. PTAs)

**Continuous space, approximation**
- **1997** Blute-Desharnais-Edalat-Panangaden [LICS] (bisim. LMPs)
- **1998** Desharnais-Edalat-Panangaden (logic LMPs)
- **1999** Desharnais-Gupta-Jagadeesan-Panangaden [CONCUR] (metric)
- **2000** Desharnais-Gupta-Jagadeesan-Panangaden [LICS] (approx. LMPs)
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 $S \cdot R \cdot S^T > 0$

- **Unfold into infinite paths** $s_0 \uparrow s_1 \uparrow s_2 \uparrow s_3 \ldots$
  - $\text{prob}_s (s')$, probability of being in $s'$ in the long-run, starting in $s$
  - $\text{prob}_s (s',t)$, 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
    - \( P_{<0.85}(<15 \text{ full}) \), probability operator
      "the probability of queue becoming full within 15 secs is < 0.85"
    - \( S_{<0.01} \text{(down)} \), 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:

\[
\phi ::= \text{true} \mid a \mid \phi \land \phi \mid :\phi \mid S_{\geq p}(\phi) \mid P_{\geq p}(\alpha)
\]

\[
\alpha ::= X \phi \mid \phi U^+ \phi \mid \phi U \phi
\]

where \( p \in [0,1] \) is a probability bound, \( t \in \mathbb{R}_{\geq 0} \) and \( \geq \{ <, >, \ldots \} \)

- Extension with time intervals for until, cost/rewards and expectation operator \( E_{\geq c}(\phi) \)
CSL semantics

• Semantics of bounded until:
  \[ \pi^2 \phi_1 U^t \phi_2 \]
  iff \( \phi_2 \) satisfied at time instant \( t \) along \( \pi = s_0 \cdots \) and \( \phi_1 \) satisfied at all preceding time instants

• The added operators:
  \[ s^2 S_p(\phi) \]
  \[ s^2 P_p(\alpha) \]
  \[ \Sigma_{s'} \phi \text{ prob}_{s}(s') \gg p \]
  \[ \Pr \{ \pi^2 \text{ Path}_s j \pi^2 \alpha \} \gg p \]
  where \( \text{prob}_{s}(s') \) is prob. of being in \( s' \) in the long-run, having started in \( s \)
  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
  - $S_p(\phi)$ and $P_p(\phi_1 \cup^t \phi_2)$

• 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 $\phi_2$ or $\phi_1$
  - Then $Pr \{ \pi \phi \text{Paths}_s | \pi \phi U^t \phi \} = \sum_{s',t} \phi_2 \text{prob}_s(s',t)$
  - Computed by using uniformisation
  - More efficient and stable, iterative computation
Probabilistic model checking in practice

- **Model construction**: probability matrices
  - **Enumerative**
    - Manipulation of *individual* states
    - Size of state space main limitation
  - **Symbolic**
    - Manipulation of *sets* of states
    - Compact representation possible in case of regularity

- **Temporal logic** model checking: currently limited to
  - discrete probability/space models
  - CTMCs
  - Simulation admits more general distributions

- **Probabilistic Symbolic Model Checker** PRISM
The PRISM tool: overview

• Functionality
  - Direct support for models: DTMCs, MDPs and CTMCs
  - Extension with costs/rewards, expectation operator
  - PTAs with digital clocks by manual translation
  - Connection from KRONOS to PRISM for PTAs
  - Experimental implementation using DBMs/DDDs for PTAs

• Input languages
  - System description
    • probabilistic extension of reactive modules [Alur and Henzinger]
  - Probabilistic temporal logics: PCTL and CSL

• Implementation
  - Symbolic model construction (MTBDDs), uses CUDD [Somenzi]
  - Three numerical computation engines
  - Written in Java and C++
The PRISM tool: implementation

- **Numerical engines**
  - **Symbolic**, MTBDD based
    - Fast construction, reachability analysis
    - Very large models if regularity
  - **Enumerative**, sparse-matrix based
    - Generally fast numerical computation
    - Model size up to millions
  - **Hybrid**
    - Speed comparable to sparse matrices for numerical calculations
    - Limited by size of vector

- **Experimental results**
  - Several large scale examples: $10^{10} - 10^{30}$ states
  - No engine wins overall
  - See [www.cs.bham.ac.uk/~dxp/prism](http://www.cs.bham.ac.uk/~dxp/prism)
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]
  - Self-stabilising protocols

- **CTMCs**
  - Dynamic Power Management (with Shukla and Gupta) [HLDVT’02]
  - 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 Modelling Language

- **Simple, state-based** language for DTMCs/CTMCs/MDPs
  - based on Reactive Modules [Alur/Henzinger]
- **Basic components:**
  - modules (system components, parallel composition)
  - variables (finite-state, typed)
  - guarded commands (probabilistic, action-labelled)

\[
\begin{align*}
\text{[send]} & (s=2) \rightarrow p_{\text{loss}} : (s'=3) & (\text{lost}'=\text{lost}+1) + (1-p_{\text{loss}}) : (s'=4);
\end{align*}
\]
PRISM Modelling Language...

- **Other features:**
  - Synchronisation on action labellings
  - Process algebra style specifications
    - Parallel composition: $P_1 ||| P_2$, $P_1 |[a,b]| P_2$, $P_1 || P_2$
    - Action hiding/renaming: $P/\{a\}$, $P\{a<b\}$
  - Import of PEPA models
  - State-dependent probabilities/rates
  - Global variables, macros, ...
PRISM Property Specifications

- Temporal logics: **PCTL/CSL**
  - probabilistic extensions of **CTL**
- Examples:
  - \( P \geq 1 \text{ [ true U terminate ] } \)
    “the algorithm eventually terminates successfully with probability 1”
  - \( P < 0.001 \text{ [ true U} \leq 100 \text{ error ] } \)
    “the probability of the system reaching an error state within 100 time units is less than 0.001”
PRISM Property Specifications

More examples:

- **down => P>0.75 [ !fail U[1,2.5] up ]**
  “when a shutdown occurs, the probability of system recovery being completed in between 1 and 2.5 hours, without further failures occurring, is greater than 0.75”

- **S<0.01 [ num_sensors < min ]**
  “in the long-run, the probability that an inadequate number of sensors are operational is less than 0.01”
  (CSL only)
PRISM Property Specifications...

- Can write query formulae:
  - $P=? \left[ \text{true} \cup \leq 10 \text{ terminate} \right]$
  
  "What is the probability that the algorithm terminates successfully within 10 time units?"

- Can automate model checking with experiments:
  - $P=? \left[ \text{true} \cup \leq T \text{ terminate} \right]$
  
  "What is the probability that the algorithm terminates successfully within time $T$?" for $T=0,\ldots,1000$
Adding Costs/Rewards

- Augment states and transitions of model with real-valued rewards

- Instantaneous rewards
  - state-based
  - e.g. "queue size", "concentration of reactant"

- Cumulative rewards
  - state- and transition-based
  - e.g. "time taken", "power consumed", "messages lost"
Properties - Instantaneous

- $R = ? \ [ I = T ]$
  Expected reward at time instant $T$?

- $R = ? \ [ S ]$
  Expected long-run reward?
Properties - Cumulative

- \( R = ? \ [ F \ A ] \)
  Expected reward to reach \( A \)?

- \( R = ? \ [ C \leq T ] \)
  Expected reward by time \( T \)?

- \( R = ? \ [ S ] \)
  Expected long-run reward per unit time?
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: $\text{Na} + \text{Cl} \leftrightarrow \text{Na}^+ + \text{Cl}^-$
  - forward base rate 100
  - backwards base rate 10
  - initially $N_1$ Na molecules and $N_2$ Cl molecules
Results: Molecular Reactions

- $P_{\approx ?} (true \ U^{[T,T]} Na=i)$ ‘probability $i$ Na molecules at time $T$’
Results: Molecular Reactions

- $R_{\text{Na}}(I=T)$ 'expected percentage of Na molecules at time $T$'
Results: Molecular Reactions

- $R_{\pm} (S)$ ‘expected percentage of Na molecules in the long run’
Case Study: Power management

- **Power Management**
  - controls *power consumption* in battery-operated devices
  - savings in *power usage* translate to *extended battery life*
  - important for portable, mobile and handheld electronic devices

- **System level power management**
  - Manages various system devices for *power optimisation*
  - System components manufactured with several *power modes*
    - e.g. disk drive has: active, idle, standby, sleep, ...
  - Modes can be changed by the operating system through *APIs*
  - Exploits application characteristics
  - Needs to be implemented at the O/S level
Dynamic Power Management (DPM)

- **DPM** make *optimal decisions* at runtime based on:
  - Dynamically changing system state
  - Workload
  - Performance constraints

- **Stochastic optimal control strategies** for **DPM**
  - Construct a mathematical model of the system in PRISM
    - transition times modelled with exponential distributions
    - Model is CTMC or DTMC depending on time domain
  - Formulate stochastic optimisation problems
    - e.g. “optimise average energy usage while average delay below k”
  - Create stochastic strategies by solving optimisation problem
    - Exported to Maple for solution externally
  - Analyse optimal stochastic strategies directly in PRISM
DPM: The System Model

- **Service requester (SR)** (generates the service requests)
- **Service provider (SP)** (provides service to the requests)
- **Service queue (SQ)** (buffers the requests)
- **Power manager (PM)** (monitors the states of the SR, SP and SQ and issues state-transition commands to the SP)
DPM Case Study: Fujitsu Disk Drive

- **4 state** Fujitsu disk drive: busy, idle, standby and sleep
  - Modelled as CTMC
- **Policies:**
  - *Minimize:* average power consumption
  - *Constraint:* average queue size
- **Properties** checked with PRISM:
  - Average power consumption/queue size
  - Average number of lost customers
  - Expected power consumption/queue size by time $t$
  - Expected number of lost customers by time $t$
  - Probability $n$ requests lost by time $t$
  - Probability a request gets lost/served by time $t$

- See **PRISM web site** for further details
DPM Results: Fujitsu Disk Drive
DPM Results: Fujitsu Disk Drive
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
Modelling the host
Modelling the environment
Expected costs

- Compute minimum/maximum expected cost accumulated before obtaining a valid IP address?

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: 
  - $r=1$ (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

- 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]
Successes so far

- **Fully automatic, no expert knowledge needed for**
  - Probabilistic reachability and temporal logic properties
  - Expected time/cost

- **Tangible results!**
  - 5 cases of “unusual behaviour” found, over 20 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
Comparison of model checking engines

- Tandem queueing network
  - “first station becomes fully occupied within $t$ time units”

<table>
<thead>
<tr>
<th>States:</th>
<th>MTBDD (sec)</th>
<th>Sparse (sec)</th>
<th>Hybrid (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32,640</td>
<td>0.04</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>130,816</td>
<td>0.06</td>
<td>0.15</td>
<td>0.23</td>
</tr>
<tr>
<td>523,776</td>
<td>0.10</td>
<td>0.71</td>
<td>0.99</td>
</tr>
<tr>
<td>2,096,128</td>
<td>0.23</td>
<td>-</td>
<td>3.89</td>
</tr>
<tr>
<td>33,550,336</td>
<td>0.66</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

(450 MHz workstation, 500 MB memory)
Comparison of model checking engines

- Kanban manufacturing system
  - Computation of steady-state probabilities

<table>
<thead>
<tr>
<th>States:</th>
<th>MTBDD</th>
<th>Sparse</th>
<th>Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>58,400</td>
<td>41.7</td>
<td>0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>454,475</td>
<td>-</td>
<td>0.44</td>
<td>0.50</td>
</tr>
<tr>
<td>2,546,432</td>
<td>-</td>
<td>2.76</td>
<td>3.15</td>
</tr>
<tr>
<td>11,261,376</td>
<td>-</td>
<td>-</td>
<td>14.8</td>
</tr>
<tr>
<td>41,644,800</td>
<td>-</td>
<td>-</td>
<td>58.9</td>
</tr>
</tbody>
</table>

(450 MHz workstation, 1 GB memory)
• 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/](http://www.cs.bham.ac.uk/~dxp/prism/)

- Case studies, statistics, group publications
- Download, version 2.1 (900 users)
- Publications by others and courses that feature PRISM...
PRISM Contributors