Protocol analysis via probabilistic model checking

Marta Kwiatkowska
School of Computer Science

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

Stanford University, 15th Nov 2004
Overview

- Network protocols
  - Probability - why needed, challenges

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

- Case studies
  - Self-stabilising algorithms
  - Bluetooth device discovery
  - Contract signing

- 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"
Probabilistic model checking...

**in a nutshell**

- **Probabilistic model**
- **Probabilistic Model Checker**

**Send → P_{>0.9}(◊deliver)**

**Probabilistic temporal logic specification**

<table>
<thead>
<tr>
<th>State</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.6789</td>
</tr>
<tr>
<td>6</td>
<td>0.9789</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.1245</td>
</tr>
</tbody>
</table>
Verification via model checking…

or falsification?

The model

Model Checker

send \rightarrow \diamond deliver

Temporal logic specification

Also refinement checking, equivalence checking, …
Probabilistic model checking...

Probabilistic model

Probabilistic temporal logic specification

send → P_{0.9}(◊deliver)

Probabilistic model checker

in a nutshell

or

The probability

\[
\begin{align*}
\text{State 1: } & 0.1245 \\
\text{State 5: } & 0.6789 \\
\text{State 6: } & 0.9789 \\
\text{State 7: } & 1.0 \\
\text{...} & \\
\text{State 12: } & 0 \\
\text{State 13: } & 0.1245
\end{align*}
\]
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

\[ \sum_{i} p_i = 1 \]
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
  - \(P: S \times S \rightarrow [0,1]\) probability matrix, s.t. \(\sum_{s'} P(s,s') = 1\), all \(s\)
  - \(L: S \rightarrow 2^{AP}\) atomic propositions

- Unfold into infinite paths \(s_0s_1s_2s_3s_4\ldots\) s.t. \(P(s_i,s_{i+1}) > 0\), all \(i\)

- Probability for finite paths, multiply along path
  e.g. \(s_0s_1s_1s_2\) is \(1 \cdot 0.01 \cdot 0.97 = 0.0097\)
• Intuitively:
  - **Sample space** = infinite paths \( \text{Path}_s \) from \( s \)
  - **Event** = set of paths
  - **Basic event** = cone

• Formally, \((\text{Path}_s, \Omega, \Pr)\)
  - For finite path \( \omega = s_1 s_2 \ldots s_n \), define probability

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

  - Take \( \Omega \) least \( \sigma \)-algebra containing cones

    \[
    C(\omega) = \{ \pi \mid 2 \text{ Path}_s \mid \omega \text{ is prefix of } \pi \}
    \]

  - Define \( \Pr(C(\omega)) = P(\omega) \), all \( \omega \)
  - \( \Pr \) extends uniquely to measure on \( \text{Path}_s \)
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 \(S\)
  - \(L\): \(S \rightarrow 2^{\text{AP}}\) 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 → P_{0.9}(◊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:

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

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

- Subsumes the qualitative variants [Var85,CY95] \( P_{\geq1}(\alpha), P_{>0}(\alpha) \)

- Extension with cost/rewards and expectation operator \( E_{c}(\phi) \)
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 \(8, 9\)

\[
\begin{align*}
S_{Adv}^2 \mathcal{P} \gg_p \alpha, \quad & \Pr^A \{ \pi \; 2 \; \text{Path}^A_s \; j \; \pi_{Adv}^2 \alpha \} \gg_p \\
& \text{for all } A \in \text{Adv}
\end{align*}
\]
PCTL semantics: summary

- Semantics of state formulas:
  \[ s^{2}_{\text{Adv}} a, \quad a 2 L(s) \]
  \[ s^{2}_{\text{Adv}} :\phi, \quad s^{2}_{\text{Adv}} \phi \]
  \[ s^{2}_{\text{Adv}} \phi_{1} \land \phi_{2}, \quad s^{2}_{\text{Adv}} \phi_{1} \land s^{2}_{\text{Adv}} \phi_{2} \]

- Semantics of path formulas:
  \[ \pi^{2}_{\text{Adv}} X \phi, \quad \pi = s_{0} \cdots \text{ and } s_{1}^{2}_{\text{Adv}} \phi \]
  \[ \pi^{2}_{\text{Adv}} \phi_{1} \lor \phi_{2}, \quad \pi = s_{0} \cdots \text{ and } 9 \text{ k s.t. } s_{k}^{2}_{\text{Adv}} \phi_{2} \text{ and } 8 \text{ j < k . } s_{j}^{2}_{\text{Adv}} \phi_{1} \]

- The probabilistic operator:
  \[ s^{2}_{\text{Adv}} P_{p} (\alpha), \quad \text{Pr}^{A} \{ \pi 2 \text{ Path}^{A} s_{j}^{2}_{\text{Adv}} \alpha \} \rightarrow p \]
  for all \( A 2 \text{ Adv} \)
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
  - \( \text{Sat}( P » p(\mathbf{X} \phi) ) \), \( \{ s \in S | \sum_{s' \in S} \text{Sat}(\phi) P(s,s') » p \} \)
  - \( \text{Sat}( P » p(\phi_1 \mathbf{U} \phi_2) ) \), \( \{ s \in S | x_s » p \} \)

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

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

and

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

- For the remaining formulas standard:

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

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

- Need to combine
  - Conventional *graph-theoretic traversal*
  - Numerical linear algebra
PCTL model checking for MDPs

- \(S_{yes}, S_{no}\) can also be precomputed by graph traversal (BDD fixed point) [dAKNP97]

- The linear equation generalises to linear optimisation problems solvable iteratively, e.g.

\[
\text{Sat}(P_{p}(\phi_1 \cup \phi_2)) \quad \{s \in S \mid x_s, p\}
\]

\[
x_s = \begin{cases} 
  0 & \text{if } s \in S_{no} \\
  1 & \text{if } s \in S_{yes} \\
  \min_{\mu} 2 \text{ Steps}(s) \sum_{s' \in S} \mu(s') \not\in x_{s'} & \text{if } s \in S(S_{no} \mid S_{yes})
\end{cases}
\]

- Need to combine
  - Conventional graph-theoretic traversal
  - Linear optimisation (simplified value iteration)
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

\[ S_0^{+1} f(x)dx = 1 \]
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 $1 - e^{-R(s,s')c^t}$
  - Transition rate matrix $S \in \mathbb{R}_{\geq 0}$

- Unfold into infinite paths $s_0 \uparrow s_1 \uparrow s_2 \uparrow s_3 ...$
  - $\text{prob}_s(s')$, 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
• Dense real-time, $t \in \mathbb{R}_0$

• **Clocks** take values from time domain $\mathbb{R}_0$
  - Increase at the same rate as real time
  - Assume finite set $X$ of clocks, maximum const $K_{\text{max}}$
  - If $n$ clocks, $v,v' \in \mathbb{R}_0^n$ are clock valuations
  - $v + t$ is time increment, $v[X:=0]$ clock reset of all clocks in $X$

• **Zones** of $X$, for $x,y \in X$, $c \in \mathbb{N}$

\[
\zeta := x \sim c \quad \land \quad x-y \sim c \quad \land \quad \zeta \in \mathbb{E} \quad \zeta \in \mathbb{C} \quad \zeta \in \mathbb{I} : \zeta
\]

- Consider only in canonical form
- **Closed, diagonal-free** if do not feature $x < c, x > c, x-y \sim c$
- Convex, or non-convex (cf [Tripakis98])
Probabilistic Timed Automata: syntax

- **Features:**
  - Clocks, $x$, real-valued
  - Can be reset, e.g. $\{x:=0\}$
  - Invariants, e.g. $x \cdot 8$
  - Probabilistic transitions, guarded e.g. $x \cdot 4$, $x=8$

- Formally, $(\text{Loc}, s_0, \text{Inv}, \text{prob}, \text{Act}, L)$:
  - Loc finite set of locations
  - $s_0$ initial location
  - Inv maps locations $s$ to invariant clock constraints
  - prob probabilistic edge relation that yields the probability of moving from $s$ to $s'$ if enabled at $s$, resetting specified clocks
  - Act action labelling of transitions $\mu$ (probability distribution)
  - L: $S \rightarrow 2^{AP}$ atomic propositions
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) [JCS 2004]
  - Randomised consensus [CAV’01]
  - Randomised Byzantine Agreement [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]
Case Study: Self-Stabilization

- Self-stabilizing protocol for a network of processes
  - starts from possibly illegal start state
  - returns to a legal (stable) state
    - without any outside intervention
    - within some finite number of steps

- Network: synchronous or asynchronous ring of \( N \) processes
  - Illegal states: more than one process is privileged (has a token)
  - Stable states: exactly one process is privileged (has a token)
  - Properties
    - From any state, a stable state is reached with probability 1
    - Expected time to reach a stable state
Herman's self-stabilising protocol

- Synchronous ring of \( N \) (\( N \) odd) processes (DTMC)
  - Each process has a local boolean variable \( x_i \)
  - Token in place \( i \) if \( x_i = x_{i+1} \)
  - Basic step of process \( i \):
    - if \( x_i = x_{i+1} \) make a uniform random choice as to the next value of \( x_i \)
    - otherwise set \( x_i \) to the current value of \( x_{i+1} \)
  - In the PRISM language:

    ```
    module process1
      x1 : bool;
      [step] x1=x2  -> 0.5 : x1'=0 + 0.5 : x1'=1;
      [step] !(x1=x2)  -> x1'=x2;
    endmodule
    
    module process2 = process1 [x1=x2, x2=x3] endmodule
    ...
    ...
    module processN = process1 [x1=xN, x2=x1] endmodule
    ```
Results: Herman’s protocol

- $P_{\leq 1}(\text{stable})$: min probability of reaching a stable state is 1
- $E_{\geq 2}(\text{stable})$: max expected time (number of steps) to reach a stable state, assuming the probability is 1, is:

![Bar chart showing expected time for different numbers of processes.](chart.png)
Israeli-Jalfon’s self-stabilising protocol

- Asynchronous ring of $N$ processes (MDP)
- Each process has a local boolean variable $q_i$
  - token in place $i$ if $q_i$=true
  - process is active if and only if has a token
  - Basic step of (active) process: uniform random choice as to whether to move the token to the left or right

- In the PRISM language:

```
global q1 : [0..1]; ... global qN : [0..1];
module process1
    s1 : bool; // dummy variable
    [] (q1=1) -> 0.5 : (q1'=0) & (qN'=1) + 0.5 : (q1'=0) & (q2'=1);
endmodule

module process2 = process1 [s1=s2, q1=q2, q2=q3, qN=q1] endmodule
    : 
module processN = process1 [s1=sN, q1=qN, q2=q1, qN=qN-1] endmodule
```
Results: Israeli-Jalfon’s protocol

- $P_1(\text{stable})$: min probability of reaching a stable state is 1
- $E_2(\text{stable})$: max expected time (number of steps) to reach a stable state, assuming the probability is 1, is:

![Graph showing expected time vs. number of processes]
Beauquier, Gradinariu and Johnen's self-stabilising protocol

- Asynchronous ring of \( N \) (\( N \) odd) processes (MDP)
  - Each process has two boolean variables: \( d_i \) and \( p_i \) where:
    - if \( d_i = d_{i-1} \) process \( i \) is said to have a deterministic token
    - if \( p_i = p_{i-1} \) process \( i \) is said to have a probabilistic token
    - stable states are those where there is only one probabilistic token
    - process is active if and only if has a deterministic token
  - Basic step of (active) process \( i \):
    - negate \( d_i \) and if \( p_i = p_{i-1} \), then set \( p_i \) uniformly at random
  - In the PRISM language:

```plaintext
module process1
  d1 : bool; p1 : bool;
  [] d1=d3 & p1=p3 -> 0.5 : (d1'=!d1) & (p1'=p1) + 0.5 : (d1'=!d1) & (p1'=!p1);
  [] d1=d3 & !p1=p3 -> (d1'=!d1);
endmodule

module process2 = process1 [d1=d2, d2=d3, p1=p2, p2=p3] endmodule
  : ...
module processN = process1 [d1=dN, d2=d1, p1=pN, p2=p1] endmodule
```
Results: Beauquier, Gradinariu and Johnen’s protocol

- $P_{\hat{1}}(\hat{\text{stable}})$: min probability of reaching a stable state is 1
- $E_{\hat{2}}(\hat{\text{stable}})$: max expected time (number of steps) to reach a stable state, assuming the probability is 1, is:
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

- **Clock** CLK, 28 bit free-running, ticks every 312.5 μs
- **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):

  \[
  \text{freq} = [\text{CLK}_{16-12} + k + (\text{CLK}_{4-2,0} - \text{CLK}_{16-12}) \mod 16] \mod 32
  \]
Frequency hopping sequence

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

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

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

- **Receiver**: follows the frequency hopping sequence, own clock.
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
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
Time to hear 2 replies

- Max time to hear is 5.177 sec (16,565 slots), in 444 possible initial states
- Cumulative (derived): assumes time to reply to 2nd message is independent of time to reply to 1st (incorrect, compare with exact curve obtained from model checking)
Case Study: Contract Signing

• Two parties want to agree on a contract
• Each will sign if the other will sign
  - Cannot trust other party in the protocol
  - There may be a trusted third party (judge), but it should only be used if something goes wrong

• **Contract signing with pen and paper**
  - Sit down and write signatures *simultaneously*

• **Contract signing on the Internet**
  - Challenge: how to exchange commitments on an asynchronous network?
Contract Signing

Partial secret exchange protocol of Even, Goldreich and Lempel (1985) for two parties (A and B)

- A (B) holds secrets $a_1, \ldots, a_{2n}$ ($b_1, \ldots, b_{2n}$)
  - Secret is a binary string of length $l$
  - Secrets partitioned into pairs:
    - $\{(a_i, a_{n+i}) | i=1,\ldots,n\}$ and $\{(b_i, b_{n+i}) | i=1,\ldots,n\}$
  - A (B) committed if B (A) knows one of A's (B's) pairs

- Uses **1-out-of-2 oblivious transfer protocol**: $\text{OT}(S,R,x,y)$
  - $S$ sends $x$ and $y$ to $R$
  - $R$ receives $x$ with probability $\frac{1}{2}$ otherwise receives $y$
  - $S$ does not know which one $R$ receives
  - if $S$ cheats then $R$ can detect this with probability $\frac{1}{2}$
Contract Signing

(step 1)
for i=1,…,n
    OT(A,B, a_i, a_{n+i})
    OT(B,A b_i, b_{n+i})
end

(step 2)
for i=1,…,l (l is the bit length of the secrets)
    for j=1,…,2n
        A transmits bit i of secret a_j to B
    end
    for j=1,…,2n
        B transmits bit i of secret b_j to A
    end
end
Results: Contract Signing

- Discovered a **weakness** in the protocol when party B is allowed to act maliciously by quitting the protocol early
  - this behaviour not considered in the original analysis

- **PRISM** analysis shows:
  - if B stops participating in the protocol as soon as he/she has obtained at least one of A pairs, then, with **probability 1**, at this point:
    - B possesses a pair of A’s secrets
    - A does not have complete knowledge of any pair of B’s secrets

- Protocol is therefore not fair under this attack:
  - B has a distinct advantage over A
The protocol is unfair because in step 2: A sends a bit for each of its secret before B does.

Can we make this protocol fair by changing the message sequence scheme?

Since the protocol is asynchronous the best we can hope for is with probability $\frac{1}{2}$ B (or A) gains this advantage.

We consider 3 possible alternate message sequence schemes...
Contract Signing: EGL2

(step1)

... 

(step2)

for i=1,...,l
  for j=1,...,n A transmits bit i of secret a_j to B
  for j=1,...,n B transmits bit i of secret b_j to A
end

for i=1,...,l
  for j=n+1,...,2n A transmits bit i of secret a_j to B
  for j=n+1,...,2n B transmits bit i of secret b_j to A
end
Contract Signing: EGL3

(step1)

...  

(step2)

for $i=1,\ldots,l$ for $j=1,\ldots,n$

A transmits bit $i$ of secret $a_j$ to B  
B transmits bit $i$ of secret $b_j$ to A  

end

for $i=1,\ldots,l$ for $j=n+1,\ldots,2n$

A transmits bit $i$ of secret $a_j$ to B  
B transmits bit $i$ of secret $b_j$ to A  

end
(step 1)

...  

(step 2)

for $i=1,\ldots,l$

$A$ transmits bit $i$ of secret $a_1$ to $B$

for $j=1,\ldots,n$  $B$ transmits bit $i$ of secret $b_j$ to $A$

for $j=2,\ldots,n$  $A$ transmits bit $i$ of secret $a_j$ to $B$

end

for $i=1,\ldots,l$

$A$ transmits bit $i$ of secret $a_{n+1}$ to $B$

for $j=n+1,\ldots,2n$  $B$ transmits bit $i$ of secret $b_j$ to $A$

for $j=n+2,\ldots,2n$  $A$ transmits bit $i$ of secret $a_j$ to $B$

end
Results: Contract Signing

- Probability the other party gains knowledge first
  - The chance that the protocol is unfair

A solid line
B dashed line
Results: Contract Signing

- Expected bits a party requires to know a pair once the other knows a pair
  - quantifies how unfair the protocol is
Results: Contract Signing

- Expected messages a party must receive to know a pair once the other knows a pair
  - measures the influence the other party has on the fairness, since it can try and delay these messages

\[
\begin{align*}
\text{A solid line} & \quad \text{B dashed line}
\end{align*}
\]
Results: Contract Signing

- Expected messages that need to be sent for a party to know a pair once the other party knows a pair
  - measures the duration of unfairness

A solid line
B dashed line
Results: Contract Signing

• Results show EGL4 is the 'fairest' protocol
• Except for duration of fairness measure:
  Expected messages that need to be sent for a party to know a pair once the other party knows a pair
  - this value is larger for B than for A
  - and, in fact, as n increases, this measure:
    • increases for B
    • decreases for A

• Solution: if a party sends a sequence of bits in a row (without the other party sending messages in between), require that the party send these bits as a single message
Results: Contract Signing

- Expected messages that need to be sent for a party to know a pair once the other party knows a pair
  - measures the duration of unfairness

A solid line
B dashed line
Related projects

• **FORWARD (this case study, see ISOLA’04)**
  - Performance modelling of MAC layer of Bluetooth
  - Security analysis of Bluetooth

• **Modelling and verification of mobile ad hoc network protocols**
  - Modelling language with mobility and randomisation
  - Model checking algorithms & techniques
  - Tool development & implementation
  - Modelling timing properties of AODV

• **Focus on properties**
  - “probability of delivery within time deadline is ...”
  - “expected time to device discovery is ...”
  - “expected power consumption is ...”
Challenges for future

- Exploiting structure
  - Abstraction, data reduction, compositionality…
  - Parametric probabilistic verification?
- Proof assistant for probabilistic verification
- Extension for mobility
- Extension for hybrid systems
- Simulation, statistical testing [Younes]
- Approximation methods
- Continuous PTAs
  - Efficient model checking methods?
- More expressive specifications
  - Probabilistic LTL/PCTL*/mu-calculus?
- Real software, not models!
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 2.0 (approx. 1000 users)
- Publications by others and courses that feature PRISM...
PRISM Contributors