Automated Verification Techniques for Probabilistic Systems

Vojtěch Forejt
Marta Kwiatkowska
Gethin Norman
Dave Parker

SFM-11:CONNECT Summer School, Bertinoro, June 2011

EU-FP7: CONNECT  LSCITS/PSS  VERIWARE

THE ROYAL SOCIETY
Overview

- **Lecture 1 (9am–11am)**
  - Introduction to Modelling and Quantitative Verification
  - Marta Kwiatkowska

- **Invited lecture: Christel Baier**
  - Component and Connector Modelling Formalisms

- **Lecture 2 (2.30pm–4pm)**
  - Quantitative Compositional Verification
  - Dave Parker

- **Lab session (4.30pm–6pm)**
  - Modelling and Compositional Verification of Probabilistic Component–Based Systems using PRISM
  - Dave Parker

- [http://www.prismmodelchecker.org/courses/sfm11connect/](http://www.prismmodelchecker.org/courses/sfm11connect/)
Part 1

Introduction
Quantitative verification

• **Formal verification**
  – is the application of *rigorous*, mathematics–based techniques to establish the correctness of computerised systems

• **Quantitative verification**
  – applies *formal verification* techniques to the modelling and analysing of non–functional aspects of system behaviour (e.g. probability, time, cost, …)

• **Probabilistic model checking**
  – is a an *automated quantitative verification* technique for systems that exhibit *probabilistic* behaviour
Why formal verification?

- Errors in computerised systems can be costly...

  - Pentium chip (1994)
    Bug found in FPU. Intel (eventually) offers to replace faulty chips. Estimated loss: $475m

  - Ariane 5 (1996)
    Self-destructs 37secs into maiden launch. Cause: uncaught overflow exception.

  - Toyota Prius (2010)
    Software “glitch” found in anti-lock braking system. 185,000 cars recalled.

- Why verify?
  - “Testing can only show the presence of errors, not their absence.” [Edsger Dijkstra]
Model checking

System

Finite-state model

Temporal logic specification

Model checker

e.g. SMV, Spin

Result

Counter-example

System requirements

¬EF fail
Why probability?

• Some systems are inherently probabilistic…

• Randomisation, e.g. in distributed coordination algorithms
  – as a symmetry breaker, in gossip routing to reduce flooding

• Examples: real–world protocols featuring randomisation:
  – Randomised back–off schemes
    • CSMA protocol, 802.11 Wireless LAN
  – Random choice of waiting time
    • IEEE1394 Firewire (root contention), Bluetooth (device discovery)
  – Random choice over a set of possible addresses
    • IPv4 Zeroconf dynamic configuration (link–local addressing)
  – Randomised algorithms for anonymity, contract signing, …
Why probability?

- Some systems are inherently probabilistic...
- **Randomisation**, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding
- **To model uncertainty and performance**
  - to quantify rate of failures, express Quality of Service

**Examples:**
- computer networks, embedded systems
- power management policies
- nano-scale circuitry: reliability through defect-tolerance
Why probability?

• Some systems are inherently probabilistic…

• Randomisation, e.g. in distributed coordination algorithms
  – as a symmetry breaker, in gossip routing to reduce flooding

• To model uncertainty and performance
  – to quantify rate of failures, express Quality of Service

• To model biological processes
  – reactions occurring between large numbers of molecules are naturally modelled in a stochastic fashion
We are not just interested in correctness

We want to be able to quantify non-functional properties:
- security, privacy, trust, anonymity, fairness
- safety, reliability, performance, dependability
- resource usage, e.g. battery life
- and much more...

Quantitative, as well as qualitative requirements:
- how reliable is the disaster service provider network?
- how efficient is my phone’s power management policy?
- is my bank’s web-service secure?
- what is the expected long-run percentage of protein X?
Probabilistic model checking

System

Probabilistic model
e.g. Markov chain

0.5
0.4
0.1

Probabilistic model checker
e.g. PRISM

Result

Quantitative results

Counter-example

System requirements

Probabilistic temporal logic specification
e.g. PCTL, CSL, LTL

\[ P_{<0.1} [ F \text{ fail} ] \]
CONNECTed probabilistic systems

- Many of the probabilistic systems that we want to verify are naturally decomposed into sub-systems
  - communication protocols, power management systems, ...

- Need modelling formalisms to capture this behaviour
  - Markov decision processes (probabilistic automata)
  - combine probabilistic and nondeterministic behaviour
  - analysis non-trivial – need automated techniques and tools

- Component-based systems
  - offer opportunities to exploit their structure
  - compositional probabilistic verification: assume-guarantee
  - more generally, quantitative properties
## Probabilistic models

<table>
<thead>
<tr>
<th>Time</th>
<th>Fully probabilistic</th>
<th>Nondeterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete time</strong></td>
<td>Discrete-time Markov chains (DTMCs)</td>
<td>Markov decision processes (MDPs) (probabilistic automata)</td>
</tr>
<tr>
<td><strong>Continuous time</strong></td>
<td>Continuous-time Markov chains (CTMCs)</td>
<td>CTMDPs/IMCs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Probabilistic timed automata (PTAs)</td>
</tr>
</tbody>
</table>
Overview

• Lectures 1 and 2:
  – 1 – Introduction
  – 2 – Discrete-time Markov chains
  – 3 – Markov decision processes
  – 4 – Compositional probabilistic verification

• Course materials available here:
  – http://www.prismmodelchecker.org/courses/sfm11connect/
  – lecture slides, reference list, tutorial chapter, lab session
Part 2

Discrete–time Markov chains
Overview (Part 2)

- Discrete–time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- Other properties: LTL, costs and rewards
- Case study: Bluetooth device discovery
Discrete–time Markov chains

- **Discrete–time Markov chains (DTMCs)**
  - state–transition systems augmented with probabilities

- **States**
  - discrete set of states representing possible configurations of the system being modelled

- **Transitions**
  - transitions between states occur in discrete time–steps

- **Probabilities**
  - probability of making transitions between states is given by discrete probability distributions
Discrete-time Markov chains

- Formally, a DTMC D is a tuple $(S, s_{\text{init}}, P, L)$ where:
  - $S$ is a finite set of states ("state space")
  - $s_{\text{init}} \in S$ is the initial state
  - $P : S \times S \rightarrow [0,1]$ is the transition probability matrix
    where $\sum_{s' \in S} P(s,s') = 1$ for all $s \in S$
  - $L : S \rightarrow 2^{\text{AP}}$ is function labelling states with atomic propositions

- Note: no deadlock states
  - i.e. every state has at least one outgoing transition
  - can add self loops to represent final/terminating states
DTMCs: An alternative definition

• Alternative definition: a DTMC is:
  – a family of random variables \( \{ X(k) \mid k=0,1,2,... \} \)
  – \( X(k) \) are observations at discrete time-steps
  – i.e. \( X(k) \) is the state of the system at time-step \( k \)

• Memorylessness (Markov property)
  – \( \Pr( X(k)=s_k \mid X(k-1)=s_{k-1}, \ldots, X(0)=s_0 ) \)
    = \( \Pr( X(k)=s_k \mid X(k-1)=s_{k-1} ) \)

• We consider homogenous DTMCs
  – transition probabilities are independent of time
  – \( P(s_{k-1},s_k) = \Pr( X(k)=s_k \mid X(k-1)=s_{k-1} ) \)
Paths and probabilities

• **A (finite or infinite) path through a DTMC**
  – is a sequence of states $s_0 s_1 s_2 s_3 \ldots$ such that $P(s_i, s_{i+1}) > 0 \ \forall i$
  – represents an execution (i.e. one possible behaviour) of the system which the DTMC is modelling

• **To reason (quantitatively) about this system**
  – need to define a probability space over paths

• **Intuitively:**
  – sample space: $\text{Path}(s) = \text{set of all infinite paths from a state } s$
  – events: sets of infinite paths from $s$
  – basic events: cylinder sets (or “cones”)
  – cylinder set $C(\omega)$, for a finite path $\omega$
    = set of infinite paths with the common finite prefix $\omega$
  – for example: $C(ss_1s_2)$
Probability spaces

- Let \( \Omega \) be an arbitrary non-empty set
- A \( \sigma \)-algebra (or \( \sigma \)-field) on \( \Omega \) is a family \( \Sigma \) of subsets of \( \Omega \) closed under complementation and countable union, i.e.:
  - if \( A \in \Sigma \), the complement \( \Omega \setminus A \) is in \( \Sigma \)
  - if \( A_i \in \Sigma \) for \( i \in \mathbb{N} \), the union \( \bigcup_i A_i \) is in \( \Sigma \)
  - the empty set \( \emptyset \) is in \( \Sigma \)
- **Theorem:** For any family \( F \) of subsets of \( \Omega \), there exists a unique smallest \( \sigma \)-algebra on \( \Omega \) containing \( F \)
- **Probability space** \((\Omega, \Sigma, \Pr)\)
  - \( \Omega \) is the sample space
  - \( \Sigma \) is the set of events: \( \sigma \)-algebra on \( \Omega \)
  - \( \Pr : \Sigma \to [0,1] \) is the probability measure:
    \[ \Pr(\Omega) = 1 \text{ and } \Pr(\bigcup_i A_i) = \sum_i \Pr(A_i) \text{ for countable disjoint } A_i \]
Probability space over paths

• **Sample space** $\Omega = \text{Path}(s)$
  set of infinite paths with initial state $s$

• **Event set** $\Sigma_{\text{Path}(s)}$
  – the **cylinder set** $C(\omega) = \{ \omega' \in \text{Path}(s) \mid \omega \text{ is prefix of } \omega' \}$
  – $\Sigma_{\text{Path}(s)}$ is the least $\sigma$–algebra on $\text{Path}(s)$ containing $C(\omega)$ for all finite paths $\omega$ starting in $s$

• **Probability measure** $Pr_s$
  – define probability $P_s(\omega)$ for finite path $\omega = ss_1...s_n$ as:
    - $P_s(\omega) = 1$ if $\omega$ has length one (i.e. $\omega = s$)
    - $P_s(\omega) = P(s,s_1) \cdot ... \cdot P(s_{n-1},s_n)$ otherwise
  – define $Pr_s(C(\omega)) = P_s(\omega)$ for all finite paths $\omega$
  – $Pr_s$ extends **uniquely** to a probability measure $Pr_s : \Sigma_{\text{Path}(s)} \to [0,1]$

• **See [KSK76]** for further details
• Paths where sending fails the first time
  - $\omega = s_0s_1s_2$
  - $C(\omega) = \text{all paths starting } s_0s_1s_2...$
  - $P_{s_0}(\omega) = P(s_0,s_1) \cdot P(s_1,s_2)$
    \[= 1 \cdot 0.01 = 0.01\]
  - $Pr_{s_0}(C(\omega)) = P_{s_0}(\omega) = 0.01$

• Paths which are eventually successful and with no failures
  - $C(s_0s_1s_3) \cup C(s_0s_1s_1s_3) \cup C(s_0s_1s_1s_1s_3) \cup ...$
  - $Pr_{s_0}(C(s_0s_1s_3) \cup C(s_0s_1s_1s_3) \cup C(s_0s_1s_1s_1s_3) \cup ...)$
    \[= P_{s_0}(s_0s_1s_3) + P_{s_0}(s_0s_1s_1s_3) + P_{s_0}(s_0s_1s_1s_1s_3) + ...
    = 1 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.01 \cdot 0.98 + ...\]
    \[= 0.9898989898...
    = 98/99\]
Overview (Part 2)

• Discrete–time Markov chains (DTMCs)

• PCTL: A temporal logic for DTMCs

• PCTL model checking

• Other properties: LTL, costs and rewards

• Case study: Bluetooth device discovery
PCTL

• Temporal logic for describing properties of DTMCs
  – PCTL = Probabilistic Computation Tree Logic [HJ94]
  – essentially the same as the logic pCTL of [ASB+95]

• Extension of (non–probabilistic) temporal logic CTL
  – key addition is probabilistic operator $P$
  – quantitative extension of CTL’s A and E operators

• Example
  – $\text{send} \rightarrow P_{\geq 0.95} [\text{true U}^{\leq 10} \text{ deliver} ]$
  – “if a message is sent, then the probability of it being delivered within 10 steps is at least 0.95”
PCTL syntax

- **PCTL syntax:**
  
  \[ \phi ::= \text{true} \mid a \mid \phi \land \phi \mid \neg \phi \mid P_{\sim p} [ \psi ] \]  

  (state formulas)

  \[ \psi ::= X \phi \mid \phi U^{\leq k} \phi \mid \phi U \phi \]  

  (path formulas)

  - where \( a \) is an atomic proposition, used to identify states of interest, \( p \in [0,1] \) is a probability, \( \sim \in \{<,\geq\} \), \( k \in \mathbb{N} \)

- **A PCTL formula is always a state formula**
  - path formulas only occur inside the P operator
PCTL semantics for DTMCs

- PCTL formulas interpreted over states of a DTMC
  - \( s \models \phi \) denotes \( \phi \) is “true in state \( s \)” or “satisfied in state \( s \)”

- Semantics of (non-probabilistic) state formulas:
  - for a state \( s \) of the DTMC \( (S,s_{\text{init}},P,L) \):
    - \( s \models a \) \iff \( a \in L(s) \)
    - \( s \models \phi_1 \land \phi_2 \) \iff \( s \models \phi_1 \) and \( s \models \phi_2 \)
    - \( s \models \neg \phi \) \iff \( s \models \phi \) is false

- Examples
  - \( s_3 \models \text{succ} \)
  - \( s_1 \models \text{try} \land \neg \text{fail} \)
PCTL semantics for DTMCs

• **Semantics of path formulas:**
  - for a path $\omega = s_0s_1s_2...$ in the DTMC:
  - $\omega \models X \phi \Leftrightarrow s_1 \models \phi$
  - $\omega \models \phi_1 U \leq_k \phi_2 \Leftrightarrow \exists i \leq k$ such that $s_i \models \phi_2$ and $\forall j < i, s_j \models \phi_1$
  - $\omega \models \phi_1 U \phi_2 \Leftrightarrow \exists k \geq 0$ such that $\omega \models \phi_1 U \leq_k \phi_2$

• **Some examples of satisfying paths:**
  - $X$ succ
    - $\{\text{try}\} \{\text{succ}\} \{\text{succ}\} \{\text{succ}\}$
    ![Diagram](image1)
  - $\neg$fail U succ
    - $\{\text{try}\} \{\text{try}\} \{\text{succ}\} \{\text{succ}\}$
    ![Diagram](image2)
• **Semantics of the probabilistic operator** $P$
  
  – informal definition: $s \models P_{\sim p} [\psi]$ means that “the probability, from state $s$, that $\psi$ is true for an outgoing path satisfies $\sim p$”
  
  – example: $s \models P_{<0.25} [X \text{ fail }] \iff \text{“the probability of atomic proposition fail being true in the next state of outgoing paths from } s \text{ is less than 0.25”}$
  
  – formally: $s \models P_{\sim p} [\psi] \iff \text{Prob}(s, \psi) \sim p$
  
  – where: $\text{Prob}(s, \psi) = \Pr_s \{ \omega \in \text{Path}(s) \mid \omega \models \psi \}$
  
  – (sets of paths satisfying $\psi$ are always measurable [Var85])

\[
\begin{array}{c}
\text{\textbf{s}} \\
\text{\textbf{-\psi}} \\
\text{\textbf{Prob}(s, \psi) \sim p ?}}
\end{array}
\]
More PCTL…

- **Usual temporal logic equivalences:**
  - false $\equiv \neg \text{true}$ (false)
  - $\phi_1 \lor \phi_2 \equiv \neg (\neg \phi_1 \land \neg \phi_2)$ (disjunction)
  - $\phi_1 \rightarrow \phi_2 \equiv \neg \phi_1 \lor \phi_2$ (implication)
  - $\text{F } \phi \equiv \Diamond \phi \equiv \text{true} \lor \phi$ (eventually, “future”)
  - $\text{G } \phi \equiv \Box \phi \equiv \neg (\text{F } \neg \phi)$ (always, “globally”)
  - bounded variants: $\text{F}^{\leq k} \phi$, $\text{G}^{\leq k} \phi$

- **Negation and probabilities**
  - e.g. $\neg P_{>p} [ \phi_1 \lor \phi_2 ] \equiv P_{\leq p} [\phi_1 \lor \phi_2 ]$
  - e.g. $P_{>p} [ \text{G } \phi ] \equiv P_{<1-p} [ \text{F } \neg \phi ]$
Qualitative vs. quantitative properties

- The P operator of PCTL can be seen as a quantitative analogue of the CTL operators A (for all) and E (there exists).

- A PCTL property $P_{\sim p} [ \psi ]$ is...
  - qualitative when $p$ is either 0 or 1
  - quantitative when $p$ is in the range (0,1)

- $P_{>0} [ F \phi ]$ is identical to $EF \phi$
  - there exists a finite path to a $\phi$-state

- $P_{\geq 1} [ F \phi ]$ is (similar to but) weaker than $AF \phi$
  - e.g. $AF \text{"tails"}$ (CTL) $\neq P_{\geq 1} [ F \text{"tails"} ]$ (PCTL)
Quantitative properties

- Consider a PCTL formula $P_{\sim p} (\psi)$
  - if the probability is unknown, how to choose the bound $p$?
- When the outermost operator of a PTCL formula is $P$
  - we allow the form $P=? (\psi)$
  - “what is the probability that path formula $\psi$ is true?”
- Model checking is no harder: compute the values anyway
- Useful to spot patterns, trends

- Example
  - $P=? (F \text{ err/total}>0.1)$
  - “what is the probability that 10% of the NAND gate outputs are erroneous?”
Some real PCTL examples

- **NAND multiplexing system**
  - \( P_{=?} \ [ F \text{ err/total} \geq 0.1 ] \)
  - “what is the probability that 10% of the NAND gate outputs are erroneous?”

- **Bluetooth wireless communication protocol**
  - \( P_{=?} \ [ F \leq t \ \text{reply\_count}=k ] \)
  - “what is the probability that the sender has received \( k \) acknowledgements within \( t \) clock-ticks?”

- **Security: EGL contract signing protocol**
  - \( P_{=?} \ [ F (\text{pairs\_a}=0 \ & \ \text{pairs\_b}>0) ] \)
  - “what is the probability that the party B gains an unfair advantage during the execution of the protocol?”
Overview (Part 2)

• Discrete–time Markov chains (DTMCs)

• PCTL: A temporal logic for DTMCs

• PCTL model checking

• Other properties: LTL, costs and rewards

• Case study: Bluetooth device discovery
PCTL model checking for DTMCs

- **Algorithm for PCTL model checking** [CY88,HJ94,CY95]
  - inputs: DTMC $D=(S,s_{\text{init}},P,L)$, PCTL formula $\phi$
  - output: $\text{Sat}(\phi) = \{ s \in S \mid s \models \phi \} = \text{set of states satisfying } \phi$

- **What does it mean for a DTMC $D$ to satisfy a formula $\phi$?**
  - sometimes, want to check that $s \models \phi \ \forall s \in S$, i.e. $\text{Sat}(\phi) = S$
  - sometimes, just want to know if $s_{\text{init}} \models \phi$, i.e. if $s_{\text{init}} \in \text{Sat}(\phi)$

- **Sometimes, focus on quantitative results**
  - e.g. compute result of $P=\? [ F \ \text{error} ]$
  - e.g. compute result of $P=\? [ F^{\leq k} \ \text{error} ]$ for $0 \leq k \leq 100$
PCTL model checking for DTMCs

- **Basic algorithm proceeds by induction on parse tree of $\phi$**
  - example: $\phi = (\neg \text{fail} \land \text{try}) \rightarrow P_{>0.95} [ \neg \text{fail} U \text{succ} ]$

- **For the non-probabilistic operators:**
  - $\text{Sat}(\text{true}) = S$
  - $\text{Sat}(a) = \{ s \in S \mid a \in L(s) \}$
  - $\text{Sat}(\neg \phi) = S \setminus \text{Sat}(\phi)$
  - $\text{Sat}(\phi_1 \land \phi_2) = \text{Sat}(\phi_1) \cap \text{Sat}(\phi_2)$

- **For the $P_{>p} [ \psi ]$ operator**
  - need to compute the probabilities $\text{Prob}(s, \psi)$ for all states $s \in S$
  - focus here on “until” case: $\psi = \phi_1 U \phi_2$
PCTL until for DTMCs

• Computation of probabilities Prob(s, φ₁ U φ₂) for all s ∈ S
• First, identify all states where the probability is 1 or 0
  – \( S^{yes} = \text{Sat}(P_{\geq 1} [ \phi_1 U \phi_2 ]) \)
  – \( S^{no} = \text{Sat}(P_{\leq 0} [ \phi_1 U \phi_2 ]) \)
• Then solve linear equation system for remaining states

• We refer to the first phase as "precomputation"
  – two algorithms: Prob0 (for \( S^{no} \)) and Prob1 (for \( S^{yes} \))
  – algorithms work on underlying graph (probabilities irrelevant)
• Important for several reasons
  – reduces the set of states for which probabilities must be computed numerically (which is more expensive)
  – gives exact results for the states in \( S^{yes} \) and \( S^{no} \) (no round-off)
  – for \( P_{\sim p}[\cdot] \) where \( p \) is 0 or 1, no further computation required
PCTL until – Linear equations

- Probabilities $\text{Prob}(s, \phi_1 U \phi_2)$ can now be obtained as the unique solution of the following set of linear equations:

$$
\text{Prob}(s, \phi_1 U \phi_2) = \begin{cases} 
1 & \text{if } s \in S^{yes} \\
0 & \text{if } s \in S^{no} \\
\sum_{s' \in S} P(s,s') \cdot \text{Prob}(s', \phi_1 U \phi_2) & \text{otherwise}
\end{cases}
$$

- can be reduced to a system in $|S^?|$ unknowns instead of $|S|$ where $S^? = S \setminus (S^{yes} \cup S^{no})$

- This can be solved with (a variety of) standard techniques
  - direct methods, e.g. Gaussian elimination
  - iterative methods, e.g. Jacobi, Gauss–Seidel, … (preferred in practice due to scalability)
PCTL until – Example

- Example: $P_{>0.8}[\neg a \cup b]$
Example: $P_{>0.8} [\neg a \cup b ]$

$S_{no} = \text{Sat}(P_{\leq 0} [\neg a \cup b ])$

$S_{yes} = \text{Sat}(P_{\geq 1} [\neg a \cup b ])$
Example: $P_{>0.8}[\neg a \cup b]$

Let $x_s = \text{Prob}(s, \neg a \cup b)$

Solve:

$x_4 = x_5 = 1$
$x_1 = x_3 = 0$
$x_0 = 0.1x_1 + 0.9x_2 = 0.8$
$x_2 = 0.1x_2 + 0.1x_3 + 0.3x_5 + 0.5x_4 = \frac{8}{9}$

$\text{Prob}(\neg a \cup b) = x = [0.8, 0, \frac{8}{9}, 0, 1, 1]$

$\text{Sat}(P_{\geq 1}[\neg a \cup b])$

$\text{Sat}(P_{\leq 0}[\neg a \cup b])$

$\text{Sat}(P_{>0.8}[\neg a \cup b]) = \{ s_2, s_4, s_5 \}$
PCTL model checking – Summary

• **Computation of set Sat(Φ) for DTMC D and PCTL formula Φ**
  – recursive descent of parse tree
  – combination of graph algorithms, numerical computation

• **Probabilistic operator P:**
  – $X \Phi$: one matrix–vector multiplication, $O(|S|^2)$
  – $\Phi_1 U \leq^k \Phi_2$: $k$ matrix–vector multiplications, $O(k|S|^2)$
  – $\Phi_1 U \Phi_2$: linear equation system, at most $|S|$ variables, $O(|S|^3)$

• **Complexity:**
  – linear in $|\Phi|$ and polynomial in $|S|$
Overview (Part 2)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- Other properties: LTL, costs and rewards
- Case study: Bluetooth device discovery
Limitations of PCTL

- PCTL, although useful in practice, has limited expressivity
  - essentially: probability of reaching states in X, passing only through states in Y (and within k time-steps)

- More expressive logics can be used, for example:
  - LTL [Pnu77] – (non-probabilistic) linear-time temporal logic
  - PCTL* [ASB+95,BdA95] – which subsumes both PCTL and LTL
  - both allow path operators to be combined
  - (in PCTL, $P \sim p [...]$ always contains a single temporal operator)

- Another direction: extend DTMCs with costs and rewards…
LTL – Linear temporal logic

- **LTL syntax (path formulae only)**
  - $\psi ::= \text{true} \mid a \mid \psi \land \psi \mid \neg \psi \mid X \psi \mid \psi U \psi$
  - where $a \in \text{AP}$ is an atomic proposition
  - usual equivalences hold: $F \phi \equiv \text{true} U \phi$, $G \phi \equiv \neg (F \neg \phi)$
  - evaluated over paths of a model

- **Examples**
  - $(F \text{tmp\_fail}_1) \land (F \text{tmp\_fail}_2)$
  - “both servers suffer temporary failures at some point”
  - $GF \text{ ready}$
  - “the server always eventually returns to a ready–state”
  - $FG \text{ error}$
  - “an irrecoverable error occurs”
  - $G (\text{req} \rightarrow X \text{ ack})$
  - “requests are always immediately acknowledged”
LTL for DTMCs

- Same idea as PCTL: probabilities of sets of path formulae
  - for a state $s$ of a DTMC and an LTL formula $\psi$:
    - $\text{Prob}(s, \psi) = \Pr_s \{ \omega \in \text{Path}(s) \mid \omega \models \psi \}$
  - all such path sets are measurable [Var85]

- A (probabilistic) LTL specification often comprises an LTL (path) formula and a probability bound
  - e.g. $P_{\geq 1}[\text{GF ready}]$ – “with probability 1, the server always eventually returns to a ready-state”
  - e.g. $P_{<0.01}[\text{FG error}]$ – “with probability at most 0.01, an irrecoverable error occurs”

- PCTL* subsumes both LTL and PCTL
  - e.g. $P_{>0.5}[\text{GF crit}_1] \land P_{>0.5}[\text{GF crit}_2]$
Fundamental property of DTMCs

- **Strongly connected component (SCC)**
  - maximally strongly connected set of states

- **Bottom strongly connected component (BSCC)**
  - SCC $T$ from which no state outside $T$ is reachable from $T$

- **Fundamental property of DTMCs:**
  - “with probability 1, a BSCC will be reached and all of its states visited infinitely often”

  - Formally:
    - $\Pr_s \{ \omega \in \text{Path}(s) \mid \exists i \geq 0, \exists \text{BSCC } T \text{ such that } \forall j \geq i, \omega(i) \in T \text{ and } \forall s' \in T \omega(k) = s' \text{ for infinitely many } k \} = 1$
LTL model checking for DTMCs

- Steps for model checking LTL property $\psi$ on DTMC $D$
  - i.e. computing $\text{Prob}^D(s, \psi)$

1. Build a deterministic Rabin automaton (DRA) $A$ for $\psi$
  - i.e. a DRA $A$ over alphabet $2^{AP}$ accepting $\psi$-satisfying traces

2. Build the “product” DTMC $D \otimes A$
  - records state of $A$ for path through $D$ so far

3. Identify states $T_{acc}$ in “accepting” BSCCs of $D \otimes A$
  - i.e. those that meet the acceptance condition of $A$

4. Compute probability of reaching $T_{acc}$ in $D \otimes A$
  - which gives $\text{Prob}^D(s, \psi)$, as required
Example: LTL for DTMCs

DTMC $D$

DRA $A_\psi$ for $\psi = G \neg b \land GF a$

Product DTMC $D \otimes A_\psi$

Prob$^D(s, \psi)$

$= \text{Prob}^{D \otimes A_\psi}(F T_1)$

$= 3/4.$
We augment DTMCs with rewards (or, conversely, costs)
- real-valued quantities assigned to states and/or transitions
- these can have a wide range of possible interpretations

Some examples:
- elapsed time, power consumption, size of message queue, number of messages successfully delivered, net profit, ...

Costs? or rewards?
- mathematically, no distinction between rewards and costs
- when interpreted, we assume that it is desirable to minimise costs and to maximise rewards
- we will consistently use the terminology “rewards” regardless
Reward-based properties

- **Properties of DTMCs augmented with rewards**
  - allow a wide range of quantitative measures of the system
  - basic notion: expected value of rewards
  - formal property specifications will be in an extension of PCTL

- **More precisely, we use two distinct classes of property**...

  - **Instantaneous properties**
    - the expected value of the reward at some time point

  - **Cumulative properties**
    - the expected cumulated reward over some period
DTMC reward structures

• For a DTMC \((S, s_{\text{init}}, P, L)\), a reward structure is a pair \((\rho, \iota)\):
  - \(\rho : S \rightarrow \mathbb{R}_{\geq 0}\) is the state reward function (vector)
  - \(\iota : S \times S \rightarrow \mathbb{R}_{\geq 0}\) is the transition reward function (matrix)

• Example (for use with instantaneous properties)
  - “size of message queue”: \(\rho\) maps each state to the number of jobs in the queue in that state, \(\iota\) is not used

• Examples (for use with cumulative properties)
  - “time–steps”: \(\rho\) returns 1 for all states and \(\iota\) is zero (equivalently, \(\rho\) is zero and \(\iota\) returns 1 for all transitions)
  - “number of messages lost”: \(\rho\) is zero and \(\iota\) maps transitions corresponding to a message loss to 1
  - “power consumption”: \(\rho\) is defined as the per–time–step energy consumption in each state and \(\iota\) as the energy cost of each transition
PCTL and rewards

- Extend PCTL to incorporate reward-based properties
  - add an R operator, which is similar to the existing P operator

\[
\phi ::= \ldots \mid P_{\sim p}[\psi] \mid R_{\sim r}[I=k] \mid R_{\sim r}[C\leq k] \mid R_{\sim r}[F\phi]
\]

- where \( r \in \mathbb{R}_{\geq 0}, \sim \in \{<,>,\leq,\geq\}, \) \( k \in \mathbb{N} \)

- \( R_{\sim r}[\cdot] \) means “the expected value of \( \cdot \) satisfies \( \sim r \)”
Types of reward formulas

- **Instantaneous**: $R_{\sim r}[ I=^k ]$
  - “the expected value of the state reward at time-step $k$ is $\sim r$”
  - e.g. “the expected queue size after exactly 90 seconds”

- **Cumulative**: $R_{\sim r}[ C\leq^k ]$
  - “the expected reward cumulated up to time-step $k$ is $\sim r$”
  - e.g. “the expected power consumption over one hour”

- **Reachability**: $R_{\sim r}[ F \phi ]$
  - “the expected reward cumulated before reaching a state satisfying $\phi$ is $\sim r$”
  - e.g. “the expected time for the algorithm to terminate”
Reward formula semantics

- **Formal semantics of the three reward operators**
  - based on random variables over (infinite) paths

- **Recall:**
  - \( s \models P_{\sim p} [ \psi ] \iff Pr_s \{ \omega \in \text{Path}(s) \mid \omega \models \psi \} \sim p \)

- **For a state \( s \) in the DTMC:**
  - \( s \models R_{\sim r} [ I=\leq k ] \iff \text{Exp}(s, X_{I=\leq k}) \sim r \)
  - \( s \models R_{\sim r} [ C=\leq k ] \iff \text{Exp}(s, X_{C=\leq k}) \sim r \)
  - \( s \models R_{\sim r} [ F \Phi ] \iff \text{Exp}(s, X_{F\Phi}) \sim r \)

where: \( \text{Exp}(s, X) \) denotes the **expectation** of the random variable \( X : \text{Path}(s) \to \mathbb{R}_{\geq 0} \) with respect to the probability measure \( Pr_s \).
Reward formula semantics

- Definition of random variables:
  - for an infinite path $\omega = s_0s_1s_2...$

  $$X_{I-k}(\omega) = \rho(s_k)$$

  $$X_{C_{sk}}(\omega) = \begin{cases} 0 & \text{if } k = 0 \\ \sum_{i=0}^{k-1} \rho(s_i) + \iota(s_i, s_{i+1}) & \text{otherwise} \end{cases}$$

  $$X_{F_{\phi}}(\omega) = \begin{cases} 0 & \text{if } s_0 \in \text{Sat}(\phi) \\ \infty & \text{if } s_i \notin \text{Sat}(\phi) \text{ for all } i \geq 0 \\ \sum_{i=0}^{k_{\phi}-1} \rho(s_i) + \iota(s_i, s_{i+1}) & \text{otherwise} \end{cases}$$

  - where $k_{\phi} = \min\{ j \mid s_j \models \phi \}$
Model checking reward properties

- **Instantaneous**: $R_{\sim r}[I^{=k}]$
- **Cumulative**: $R_{\sim r}[C^{\leq t}]$
  - variant of the method for computing bounded until probabilities
  - solution of recursive equations

- **Reachability**: $R_{\sim r}[F \phi]$
  - similar to computing until probabilities
  - precomputation phase (identify infinite reward states)
  - then reduces to solving a system of linear equation

- For more details, see e.g. [KNP07a]
Overview (Part 2)

• Discrete–time Markov chains (DTMCs)

• PCTL: A temporal logic for DTMCs

• PCTL model checking

• Other properties: LTL, costs and rewards

• Case study: Bluetooth device discovery
The PRISM tool

- **PRISM**: Probabilistic symbolic model checker
  - developed at Birmingham/Oxford University, since 1999
  - free, open source (GPL), runs on all major OSs

- **Support for:**
  - discrete-/continuous-time Markov chains (D/CTMCs)
  - Markov decision processes (MDPs)
  - probabilistic timed automata (PTAs)
  - PCTL, CSL, LTL, PCTL*, costs/rewards, ...

- **Multiple efficient model checking engines**
  - mostly symbolic (BDDs) (up to $10^{10}$ states, $10^7$–$10^8$ on avg.)

- **Successfully applied to a wide range of case studies**
  - communication protocols, security protocols, dynamic power management, cell signalling pathways, ...

- **See:** [http://www.prismmodelchecker.org/](http://www.prismmodelchecker.org/)
Bluetooth device discovery

• **Bluetooth: short-range low-power wireless protocol**
  – widely available in phones, PDAs, laptops, ...
  – open standard, specification freely available

• **Uses frequency hopping scheme**
  – to avoid interference (uses unregulated 2.4GHz band)
  – pseudo-random selection over 32 of 79 frequencies

• **Formation of personal area networks (PANs)**
  – piconets (1 master, up to 7 slaves)
  – self-configuring: devices discover themselves

• **Device discovery**
  – mandatory first step before any communication possible
  – relatively high power consumption so performance is crucial
  – master looks for devices, slaves listens for master
Master (sender) behaviour

- 28 bit free-running clock $CLK$, ticks every 312.5µs
- Frequency hopping sequence determined by clock:
  - $freq = [CLK_{16-12} + k + (CLK_{4-2,0} - CLK_{16-12}) \mod 16] \mod 32$
  - 2 trains of 16 frequencies (determined by offset $k$), 128 times each, swap between every 2.56s
- Broadcasts “inquiry packets” on two consecutive frequencies, then listens on the same two
Slave (receiver) behaviour

- **Listens (scans) on frequencies for inquiry packets**
  - must listen on right frequency at right time
  - cycles through frequency sequence at much slower speed (every 1.28s)

- **On hearing packet, pause, send reply and then wait for a random delay before listening for subsequent packets**
  - avoid repeated collisions with other slaves

![Diagram showing the process of slave (receiver) behaviour](image-url)
Bluetooth – PRISM model

• Modelled/analysed using PRISM model checker [DKNP06]
  – model scenario with one sender and one receiver
  – synchronous (clock speed defined by Bluetooth spec)
  – model at lowest-level (one clock-tick = one transition)
  – randomised behaviour so model as a DTMC
  – use real values for delays, etc. from Bluetooth spec

• Modelling challenges
  – complex interaction between sender/receiver
  – combination of short/long time-scales – cannot scale down
  – sender/receiver not initially synchronised, so huge number of possible initial configurations (17,179,869,184)
Bluetooth – Results

• Huge DTMC – initially, model checking infeasible
  – partition into 32 scenarios, i.e. 32 separate DTMCs
  – on average, approx. $3.4 \times 10^9$ states (536,870,912 initial)
  – can be built/analysed with PRISM's MTBDD engine

• We compute:
  – $R=? \mathbf{[ F \text{ replies} \leq K \text{ "init"\{max\} ]}$
  – “worst-case expected time to hear K replies over all possible initial configurations”

• Also look at:
  – how many initial states for each possible expected time
  – cumulative distribution function (CDF) for time, assuming equal probability for each initial state
• **Worst-case expected time** = 2.5716 sec
  - in 921,600 possible initial states
  - best-case = 635 µs
• **Worst-case expected time** = 5.177 sec
  
  – in 444 possible initial states
  
  – compare actual CDF with derived version which assumes times to reply to first/second messages are independent
Bluetooth – Results

• Other results: (see [DKNP06])
  – compare versions 1.2 and 1.1 of Bluetooth, confirm 1.1 slower
  – power consumption analysis (using costs + rewards)

• Conclusions:
  – successful analysis of complex real-life model
  – detailed model, actual parameters used
  – exhaustive analysis: best/worst-case values
    · can pinpoint scenarios which give rise to them
    · not possible with simulation approaches
  – model still relatively simple
    · consider multiple receivers?
    · combine with simulation?
**Summary (Parts 1 & 2)**

- **Probabilistic model checking**
  - automated quantitative verification of stochastic systems
  - to model randomisation, failures, ...

- **Discrete–time Markov chains (DTMCs)**
  - state transition systems + discrete probabilistic choice
  - probability space over paths through a DTMC

- **Property specifications**
  - probabilistic extensions of temporal logic, e.g. PCTL, LTL
  - also: expected value of costs/rewards

- **Model checking algorithms**
  - combination of graph–based algorithms, numerical computation, automata constructions

- **Next: Markov decision processes (MDPs)**