Introduction

Probabilistic model checking
What is probabilistic model checking?

- **Probabilistic model checking…**
  - is a *formal verification* technique for modelling and analysing systems that exhibit *probabilistic* behaviour

- **Formal verification…**
  - is the application of rigorous, mathematics–based techniques to establish the correctness of computerised systems
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 37 secs 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 Dijstra]
Model checking

System

Finite-state model

Temporal logic specification

¬EF fail

System requirements

Model checker e.g. SMV, Spin

Result

Counterexample
Probabilistic model checking

System

Probabilistic model
e.g. Markov chain

Result

Quantitative results

Counter-example

System requirements

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

0.5

0.4

0.1

P_{<0.1} [ F \text{ 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
Verifying probabilistic systems

• We are not just interested in correctness

• We want to be able to quantify:
  – 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 my car’s Bluetooth 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 models

<table>
<thead>
<tr>
<th>Discrete time</th>
<th>Fully probabilistic</th>
<th>Nondeterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discrete-time Markov chains (DTMCs)</td>
<td>Markov decision processes (MDPs) (probabilistic automata)</td>
<td></td>
</tr>
<tr>
<td>Continuous time</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>
Course overview

• 5 lectures: Mon–Fri, 11am–12.30pm

  – Introduction
  – 1 – Discrete time Markov chains
  – 2 – Markov decision processes
  – 3 – Continuous–time Markov chains
  – 4 – Probabilistic model checking in practice
  – 5 – Probabilistic timed automata

• Course materials available here:
  – lecture slides, reference list
Part 1
Discrete-time Markov chains
Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
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
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
- $P : S \times S \to [0, 1]$ is the transition probability matrix where $\sum_{s' \in S} P(s, s') = 1$ for all $s \in S$
- $L : S \to 2^{\text{Ap}}$ 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 \( \{ X(k) \mid k=0,1,2,\ldots \} \)
  - \( X(k) \) 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 $s_0s_1s_2s_3\ldots$ 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:
    - $\Pr(\Omega) = 1$ and $\Pr(\bigcup_i A_i) = \sum_i \Pr(A_i)$ 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. $\omega = s$)
    - $P_s(\omega) = P(s,s_1) \cdot \ldots \cdot P(s_{n-1},s_n)$ 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
Probability space – Example

• Paths where sending fails the first time
  - \( \omega = s_0s_1s_2 \)
  - \( C(\omega) = \) all paths starting \( s_0s_1s_2 \ldots \)
  - \( P_{s_0}(\omega) = P(s_0,s_1) \cdot P(s_1,s_2) \)
    \[ = 1 \cdot 0.01 = 0.01 \]
  - \( Pr_{s_0}(C(\omega)) = P_{s_0}(\omega) = 0.01 \)

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

- Discrete-time Markov chains (DTMCs)
- **PCTL: A temporal logic for DTMCs**
- PCTL model checking
- LTL model checking
- Costs and rewards
• **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**
  – send \( \rightarrow P_{\geq0.95} [ \text{true} \ U_{\leq10} \text{deliver} ] \)
  – “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, $\sim \in \{<,>,\leq,\geq\}$, $k \in \mathbb{N}$

- **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:
    – \( \omega \models X \phi \iff s_1 \models \phi \)
    – \( \omega \models \phi_1 U \leq k \phi_2 \iff \exists i \leq k \) such that \( s_i \models \phi_2 \) and \( \forall j < i, s_j \models \phi_1 \)
    – \( \omega \models \phi_1 U \phi_2 \iff \exists k \geq 0 \) such that \( \omega \models \phi_1 U \leq k \phi_2 \)

• Some examples of satisfying paths:
  – \( X \) succ
    
    \[
    \{\text{try}\} \{\text{succ}\} \{\text{succ}\} \{\text{succ}\}
    \]
    
    \[
    s_0 \rightarrow s_1 \rightarrow s_1 \rightarrow s_3 \rightarrow s_3 \rightarrow s_3 \rightarrow \ldots
    \]

  – \( \neg \) fail U succ
    
    \[
    \{\text{try}\} \{\text{try}\} \{\text{succ}\} \{\text{succ}\}
    \]
    
    \[
    s_0 \rightarrow s_1 \rightarrow s_1 \rightarrow s_3 \rightarrow s_3 \rightarrow s_3 \rightarrow \ldots
    \]
PCTL semantics for DTMCs

• 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])
More PCTL…

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

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

- 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. $AF$ “tails” (CTL) $\neq P_{\geq 1}[F$ “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
  – $P_{=?} [ F \text{err/total} > 0.1 ]$
  – “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 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
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 $P_{\sim p} [\psi ]$ 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 \( \text{Prob}(s, \phi_1 U \phi_2) \) for all \( s \in S \)
- First, identify all states where the probability is 1 or 0
  - \( S^{\text{yes}} = \text{Sat}(P_{\geq 1}[\phi_1 U \phi_2]) \)
  - \( S^{\text{no}} = \text{Sat}(P_{\leq 0}[\phi_1 U \phi_2]) \)
- Then solve linear equation system for remaining states

- We refer to the first phase as “precomputation”
  - two algorithms: \( \text{Prob}_0 \) (for \( S^{\text{no}} \)) and \( \text{Prob}_1 \) (for \( S^{\text{yes}} \))
  - 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 \( S^{\text{yes}} \) and \( S^{\text{no}} \) (no round–off)
  - for \( P_{\sim p}[\cdot] \) where \( p \) is 0 or 1, no further computation required
• Probabilities $\text{Prob}(s, \phi_1 \cup \phi_2)$ can now be obtained as the unique solution of the following set of linear equations:

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

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

• 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: $P_{>0.8} \left[ \neg a \cup b \right]$
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:

$x_4 = x_5 = 1$
$x_1 = x_3 = 0$
$x_0 = 0.1x_1 + 0.9x_2 = 0.8$
$x_2 = 0.1x_2 + 0.1x_3 + 0.3x_5 + 0.5x_4 = 8/9$

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

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

$\text{Sat}(P_{\geq 1} [\neg a \cup b ]) = \{ s_2, s_4, s_5 \}$
PCTL model checking – Summary

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

• **Probabilistic operator** \( P \):
  – \( \chi \Phi \): 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 1)

- Discrete–time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
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, $P_{\neg p} [...]$ 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 $a \in AP$ is an atomic proposition
  - usual equivalences hold: $F \phi \equiv \text{true U} \phi$, $G \phi \equiv \neg(F \neg \phi)$

- **LTL semantics (for a path $\omega$)**
  - $\omega \models \text{true}$ always
  - $\omega \models a \iff a \in L(\omega(0))$
  - $\omega \models \psi_1 \land \psi_2 \iff \omega \models \psi_1$ and $\omega \models \psi_2$
  - $\omega \models \neg \psi \iff \omega \not\models \psi$
  - $\omega \models X \psi \iff \omega[1...] \models \psi$
  - $\omega \models \psi_1 U \psi_2 \iff \exists k \geq 0 \text{ s.t. } \omega[k...] \models \psi_2 \land \forall i < k \omega[i...] \models \psi_1$

where $\omega(i)$ is $i^{th}$ state of $\omega$, and $\omega[i...]$ is suffix starting at $\omega(i)$
LTL examples

- \((\text{F tmp}_1 \text{fail}) \land (\text{F tmp}_2 \text{fail})\)
  - “both servers suffer temporary failures at some point”

- GF ready
  - “the server always eventually returns to a ready-state”

- FG error
  - “an irrecoverable error occurs”

- G (req \(\rightarrow\) X 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. \( P_{\geq 1} [ GF \text{ ready} ] \) – “with probability 1, the server always
    eventually returns to a ready-state”
  – e.g. \( P_{<0.01} [ FG \text{ error} ] \) – “with probability at most 0.01, an
    irrecoverable error occurs”

• PCTL* subsumes both LTL and PCTL
  – e.g. \( P_{>0.5} [ GF \text{ crit}_1 ] \land P_{>0.5} [ GF \text{ crit}_2 ] \)
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

- LTL model checking for DTMCs relies on:
  - computing probability of reaching a set of “accepting” BSCCs
  - e.g. for two simple LTL formulae: GF a (“always eventually a”), FG a (“eventually always a’) we have:

  - Prob(s, GF a) = Prob(s, F T_{GFa})
    - where \(T_{GFa}\) = union of all BSCCs containing some state satisfying a

  - Prob(s, FG a) = Prob(s, F T_{FGa})
    - where \(T_{FGa}\) = union of all BSCCs containing only a–states

- To extend this idea to arbitrary LTL formula, we use \(\omega\)-automata...

Example:
Prob(s_0, GF a)
= Prob(s_0, F T_{GFa})
= Prob(s_0, F \{s_3, s_2, s_5\})
= 2/3 + 1/6 = 5/6
Deterministic Rabin automata

- ω-automata represent sets of infinite words
  - e.g. Buchi automata, Rabin automata, ...
  - for probabilistic model checking, need deterministic automata
  - so we use deterministic Rabin automata (DRAs)

- A deterministic Rabin automaton is a tuple \((Q, \Sigma, \delta, q_0, \text{Acc})\):
  - \(Q\) is a finite set of states, \(q_0 \in Q\) is an initial state
  - \(\Sigma\) is an alphabet, \(\delta : Q \times \Sigma \to Q\) is a transition function
  - \(\text{Acc} = \{ (L_i, K_i) \}_{i=1..k} \subseteq 2^Q \times 2^Q\) is an acceptance condition

- A run of a word on a DRA is accepting iff:
  - for some pair \((L_i, K_i)\), the states in \(L_i\) are visited finitely often and (some of) the states in \(K_i\) are visited infinitely often

  - or in LTL: \(\bigvee_{1 \leq i \leq k} (FG \neg L_i \land GF K_i)\)
LTL & DRAs

• Example: DRA for $FG a$
  - acceptance condition is $Acc = \{ (\{q_0\},\{q_1\}) \}$

• Can convert any LTL formula $\psi$ on atomic propositions $AP$
  - into an equivalent DRA $A_\psi$ over alphabet $2^{AP}$
  - i.e. $\omega \models \psi \iff \text{trace}(\omega) \in L(A_\psi)$ for any path $\omega$
  - can potentially incur a double exponential blow-up (but, in practice, this does not occur and $\psi$ is small anyway)

• LTL model checking for DTMCs – the basic idea
  - construct product of DTMC $D$ and DRA $A_\psi$
  - compute $\text{Prob}^D(s, \psi)$ on product DTMC $D \otimes A$
Product DTMC for a DRA

- **The product DTMC** $D \otimes A$ for:
  - for DTMC $D = (S, s_{init}, P, L)$ and
  - and (total) DRA $A = (Q, \Sigma, \delta, q_0, \{(L_i, K_i)\}_{i=1..k})$
  - is the DTMC $(S \times Q, (s_{init}, q_{init}), P', L')$ where:
    
    $q_{init} = \delta(q_0, L(s_{init}))$
    
    $P'((s_1, q_1), (s_2, q_2)) = \begin{cases} 
    P(s_1, s_2) & \text{if } q_2 = \delta(q_1, L(s_2)) \\
    0 & \text{otherwise}
    \end{cases}$

    $l_i \in L'(s, q)$ if $q \in L_i$ and $k_i \in L'(s, q)$ if $q \in K_i$

- **Note:**
  - $D \otimes A$ can be seen as unfolding of $D$ where $q$ for each state $(s, q)$ records state of automata $A$ for path fragment so far
  - since $A$ is deterministic, $D \otimes A$ is a DTMC
  - each path in $D$ has a corresponding (unique) path in $D \otimes A$
  - the probabilities of paths in $D$ are preserved in $D \otimes A$
Product DTMC for a DRA

- For DTMC $D$ and DRA $A$

\[
Prob^D(s, A) = Prob^{D \otimes A}((s, q_s), \bigvee_{1 \leq i \leq k} (FG \neg l_i \land GF k_i))
\]

- where $q_s = \delta(q_0, L(s))$

- Hence:

\[
Prob^D(s, A) = Prob^{D \otimes A}((s, q_s), F T_{Acc})
\]

- where $T_{Acc}$ is the union of all accepting BSCCs in $D \otimes A$

- an accepting BSCC $T$ of $D \otimes A$ is such that, for some $1 \leq i \leq k$, no states in $T$ satisfy $l_i$ and some state in $T$ satisfies $k_i$

- Reduces to computing BSCCs and reachability probabilities

- so overall complexity for LTL is doubly exponential in $|\psi|$, polynomial in $|M|$; but can be reduced to singly exponential
Example: LTL for DTMCs

- Compute $\text{Prob}(s_0, \neg b \land \text{GF} \ a)$ for DTMC $D$:

DTMC $D$

```
\begin{array}{c}
\begin{array}{c}
 s_0 \\
 s_1 \\
 s_2 \\
 s_3 \\
 s_4 \\
 s_5 \\
\end{array} \\
\begin{array}{c}
 \{b\} \\
 \{a\} \\
 \{b\} \\
 \{a\} \\
 1 \\
 \{a\} \\
\end{array} \\
\begin{array}{c}
 0.6 \\
 0.3 \\
 1 \\
 1 \\
 0.1 \\
 0.9 \\
\end{array} \\
\begin{array}{c}
 0.1 \\
 0.2 \\
 0.3 \\
 0.9 \\
 0.1 \\
 1 \\
\end{array} \\
\end{array}
```

DRA $A_\psi$ for $\psi = \neg b \land \text{GF} \ a$

```
\begin{array}{c}
\begin{array}{c}
 q_0 \\
 q_1 \\
 q_2 \\
\end{array} \\
\begin{array}{c}
 a \land \neg b \\
 \neg a \land \neg b \\
 b \\
\end{array} \\
\begin{array}{c}
 \neg a \land \neg b \\
 b \\
 \text{true} \\
\end{array} \\
\end{array}
```

$\text{Acc} = \{(\{\}, \{q_1\})\}$
Example: LTL for DTMCs

**DTMC D**

[Diagram of DTMC D]

**DRA A_ψ for ψ = G¬b ∧ GF a**

[Diagram of DRA A_ψ]

**Product DTMC D ⊗ A_ψ**

[Diagram of Product DTMC D ⊗ A_ψ]
Example: LTL for DTMCs

DTMC $D$

- States: $s_0, s_1, s_2, s_3, s_4, s_5$
- Transitions:
  - $s_0 \xrightarrow{0.1} s_1$
  - $s_0 \xrightarrow{0.6} s_3$
  - $s_1 \xrightarrow{0.2} s_3$
  - $s_1 \xrightarrow{0.5} s_2$
  - $s_2 \xrightarrow{0.9} s_5$
  - $s_2 \xrightarrow{0.1} s_1$
  - $s_3 \xrightarrow{0.3} s_4$
  - $s_3 \xrightarrow{1} s_4$
  - $s_4 \xrightarrow{1} s_5$
  - $s_4 \xrightarrow{0.3} s_1$
  - $s_5 \xrightarrow{1} s_2$
- Initial Distribution: $\{b\}$

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

- States: $q_0, q_1, q_2$
- Transitions:
  - $q_0 \xrightarrow{a \land \neg b} q_1$
  - $q_0 \xrightarrow{\neg a \land \neg b} q_2$
  - $q_1 \xrightarrow{\neg a \land \neg b} q_0$
  - $q_1 \xrightarrow{a \land \neg b} q_1$
  - $q_2 \xrightarrow{b} q_1$
  - $q_2 \xrightarrow{\text{true}} q_2$
- Initial Distribution: $\{\emptyset\}$

Product DTMC $D \otimes A_\psi$

- States: $s_0q_0, s_1q_2, s_2q_2, s_3q_1, s_4q_0, s_4q_2, s_5q_2$
- Transitions:
  - $s_0q_0 \xrightarrow{0.1} s_1q_2$
  - $s_0q_0 \xrightarrow{0.6} s_3q_1$
  - $s_1q_2 \xrightarrow{T_1} 1$
  - $s_1q_2 \xrightarrow{T_2} 0.2$
  - $s_2q_2 \xrightarrow{T_3} 0.1$
  - $s_2q_2 \xrightarrow{T_3} 0.9$
  - $s_3q_1 \xrightarrow{1} s_4q_0$
  - $s_3q_2 \xrightarrow{0.3} s_4q_2$
  - $s_4q_2 \xrightarrow{0.5} s_1q_2$
  - $s_4q_2 \xrightarrow{0.3} s_3q_2$
- Initial Distribution: $\{a\}$

Probabilities:

- $\text{Prob}^D(s, \psi) = \text{Prob}^{D \otimes A_\psi}(F T_1) = 3/4.$
Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
• **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 $\langle S, s_{\text{init}}, P, L \rangle$, a reward structure is a pair $(\rho, \iota)$
  - $\rho : S \to \mathbb{R}_{\geq 0}$ is the state reward function (vector)
  - $\iota : S \times S \to \mathbb{R}_{\geq 0}$ 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

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

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

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

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 \( \omega = s_0s_1s_2... \)

\[
X_{l=k}(\omega) = \rho(s_k)
\]

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

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

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

- **Instantaneous**: $R_{\sim r} [ l^=k ]$
- **Cumulative**: $R_{\sim r} [ C^{=t} ]$
  - 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]
Summary

• **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

• **Tomorrow: Markov decision processes (MDPs)**