Overview of tutorial

- Motivation
  - What are probabilities good for?

- Probabilistic Systems
  - Discrete Time Markov Chains
  - Markov Decision Processes
  - Continuous Time Markov Chains
  - Probabilistic Timed Automata

- Model Checking
  - Probabilistic Temporal Logics
  - Algorithms
  - Tools and Applications

- Outlook
  - Related work
  - Future research directions
Systems and their Models

- Concurrent systems:
  - communication protocols
  - telephone networks
  - distributed algorithms
  - multimedia protocols
  - ...

- Modelled by transition systems (= automata):
  - states (= current mode, values of variables)
  - transitions (= actions)
  - labelling of states and/or transitions
Specifications

Typically expressed in temporal logic:

- **qualitative requirements (LTL/CTL/CTL*)**
  - “if a packet is sent then eventually acknowledgement is received”

\[ sent \rightarrow \diamond ack \]

- “system never aborts”

\[ \square (\neg abort) \]

- **quantitative timing requirements (TCTL)**
  - “acknowledgement occurs within 5 time units after packet sent”

\[ z.\diamond (ack \land (z \leq 5)) \]
Verification by Model Checking

A model checker:

Inputs
system description (the model)
and specification (formula of logic)

Outputs
YES if system meets specification,
or NO otherwise, plus diagnostic trace.

Model checking tools increasingly used in industry, to mention:
- Hardware/software: smv (Clarke & McMillan)
- Promela programs: SPIN (Holzmann)
- Process algebras: FDR (Formal Systems Ltd)
- Real-time systems: Uppaal (Larsen)
- Hybrid systems: HyTech (Henzinger)
Probabilistic Systems

Arise from

- random assignment in randomized algorithms

\[ x := \frac{1}{2} : \text{head or } \frac{1}{2} : \text{tail} \]

- modelling faulty behaviour, lossy media

- modelling stochastic behaviour
  - job queues with given arrivals/service rates
  - network protocols with a given rate of transmission

NB discrete probability distributions suffice for random assignment and faulty behaviour, continuous time distributions needed to model arrival rates.
Probabilistic Specifications

Can express

- **probabilistic reachability, invariance**
  - “termination occurs with probability 1”
  - “deadlock does not occur with probability 1”
  - “system may abort with probability at most 0.05”

- **fault tolerance properties**
  - “every packet is eventually delivered with probability at least 0.95”

- **soft deadlines, as opposed to hard**
  - “acknowledgement occurs within 5 time units after packet sent with probability at least 0.98”

- **quality of service properties**
  - soft deadlines in the presence of stochastic timing
  - throughput, average load, mean waiting time
  - ...

Need **probabilistic** extensions of temporal logics.
Discrete Time Markov Chains

Also called fully probabilistic systems.

A Discrete Time Markov Chain (DTMCs) is a tuple $M = (S, P, L)$, where
- $S$ set of states
- $P : S \times S \to [0, 1]$ transition matrix
  (i.e. $\sum_{t \in S} P(s, t) = 1$ for all $s \in S$)
- $L : S \to 2^A_P$ state labelling with atomic prop.

DTMC example modelling simple probabilistic protocol which sends messages over unreliable medium:

Note only probabilistic choice in each state.
Working out probabilities

For the example DTMC on previous page:

- **Transition probabilities:** if in state $s_1$, then
  - can get to $s_0$ with probability 0.98
    - (successful delivery in next state)
  - can get to $s_3$ with probability 0.01
    - (message lost in next state)
  - can get to $s_2$ or $s_3$ with probability $0.01 + 0.01 = 0.02$
    - (unsuccessful delivery in next state)

- **Probabilities of finite paths:** intuitively, probabilities accumulate along paths
  - probability of $s_0 s_1 s_0$ is $1 \cdot 0.98$
  - probability of $s_0 s_1 s_3 s_1 s_0$ is $1 \cdot 0.01 \cdot 1 \cdot 0.98$

- **Probability space:**
  - unfold DTMCs into infinite paths
  - take infinite extensions of a finite path as basic events
Let $M = (S, \mathbf{P}, L)$ be a DTMC.

Paths $\pi$ of $M$ are (finite or infinite) sequences of states

$s_0s_1s_2\cdots s_j\cdots$

such that $\mathbf{P}(s_j, s_{j+1}) > 0$ for all $j \geq 0$.

Basic events correspond to basic cylinders $\mathcal{B}(\rho)$, where

$\rho = ss_1s_2s_3\cdots s_k$ is a finite path, i.e. sets of infinite paths starting in $s$:

$$\mathcal{B}(\rho) = \{ \pi \mid \rho \text{ is a prefix of } \pi \}$$

with probability measure:

$$\text{Prob}(\mathcal{B}(\rho)) = \mathbf{P}(s, s_1) \cdot \mathbf{P}(s_1, s_2) \cdots \mathbf{P}(s_{k-1}, s_k)$$

We complete basic cylinders to the smallest $\sigma$-algebra $\mathcal{F}(s)$ by closing wrt countable union and complement.

$\text{Prob}$ extends uniquely to the $\sigma$-algebra $\mathcal{F}(s)$.

The probability space is $(\text{Paths}_{\text{inf}}, \mathcal{F}, \text{Prob})$. 
The logic PCTL

(Hansson & Jonsson 1994)
Based on CTL + probabilistic operator $[\cdot]_{\leq p}$.

The syntax of state and path formulas of PCTL is:

$$
\phi ::= \text{true} \mid a \mid \phi_1 \land \phi_2 \mid \neg\phi \mid [\alpha]_{\leq p}
$$

$$
\alpha ::= X\phi \mid \phi_1 \mathcal{U} \phi_2
$$

where $p \in [0, 1]$ is a probability bound or threshold, and $\bowtie \in \{\leq, <, \geq, >\}$.

Meaning of probabilistic operator $[\cdot]_{\bowtie p}$:
- applies to path formulas
- involves computing probability measure $\text{Prob}$
- yields a (Boolean) state formula
PCTL over Discrete Markov Chains

Let $M = (S, P, L)$ be a DTMC.

Semantics of state formulas $\phi$, by induction, is:

\[
M, s \models \text{true} \quad \text{for all } s \\
M, s \models a \quad \text{iff } a \in L(s) \\
M, s \models \neg \phi \quad \text{iff } M, s \not\models \phi \\
M, s \models \phi_1 \land \phi_2 \quad \text{iff } M, s \models \phi_1 \text{ and } M, s \models \phi_2
\]

\[
M, s \models [\alpha]_{\bowtie p} \\
\text{iff } \Prob\{\pi \mid M, \pi \models \alpha, \pi \text{ starts in } s\} \bowtie p
\]

and of path formulas:

\[
M, \pi \models X \phi \quad \text{iff } \pi = s_0s_1s_2 \cdots \text{ and } M, s_1 \models \phi \\
M, \pi \models \phi_1 U \phi_2 \quad \text{iff } \pi = s_0s_1s_2 \cdots \text{ and } \exists k \text{ s.t.} \\
\quad M, s_k \models \phi_2 \text{ and} \\
\quad \forall j < k. M, s_j \models \phi_1 \\
\text{(i.e. } \phi_1 \text{ holds until } \phi_2 \text{ holds} \\
\text{and } \phi_2 \text{ must become } \text{true})
\]
More on PCTL

- Boolean operators only allowed for state formulas (otherwise conditional probabilities needed)

- Note on negation:
  \[ s \models [\alpha]_{\geq p} \text{ iff } s \models \neg[\alpha]_{<1-p} \]
  \[ s \models [\alpha]_{> p} \text{ iff } s \models \neg[\alpha]_{\leq 1-p} \]

- CTL path quantifiers \( \forall, \exists \) omitted since:
  - no effect on DTMCs
  - can derive analogues using:
    \[ [\text{true } \mathcal{U} \phi]_{>0} \text{ (cf } EF\phi) \]
    \[ [\text{true } \mathcal{U} \phi]_{\geq 1} \text{ (cf } AF\phi) \]

- Can add bounded until \( \phi_1 \mathcal{U}^{\leq k} \phi_2, k \) counts steps
  \[ M, \pi \models \phi_1 \mathcal{U}^{\leq k} \phi_2 \text{ iff } \pi = s_0s_1s_2 \cdots \text{ and } \exists l \leq k \text{ s.t. } \\
  M, s_l \models \phi_2 \text{ and } \\
  \forall j < l. M, s_j \models \phi_1 \]

- Derived operators:
  - \( \forall, \rightarrow, \) etc
  - \( \Diamond \phi \overset{\text{def}}{=} \text{true } \mathcal{U} \phi, \Box \phi \overset{\text{def}}{=} \neg \Diamond \neg \phi \)
  - \( \Diamond^{\leq k} \phi \overset{\text{def}}{=} \text{true } \mathcal{U}^{\leq k} \phi, \Box^{\leq k} \phi \overset{\text{def}}{=} \neg \Diamond^{\leq k} \neg \phi \)
Some examples

For the protocol example:

- $[\text{Xerror}]_{>0}$:
  \( error \) occurs in next state with positive probability
  \[
  s_0 \not\models [\text{Xerror}]_{>0} \\
  s_1 \models [\text{Xerror}]_{>0}
  \]

- $[\Diamond \text{init}]_{\geq 1}$:
  \( \text{init} \) is eventually reached with probability 1
  
  Sum up probabilities over paths:
  
  \[
  s_1s_0 \\
  s_1s_2s_0 \\
  s_1s_3s_1s_2s_0 \text{ or } s_1s_3s_1s_0 \\
  \vdots \\
  s_1(s_3s_1)^n s_2s_0 \text{ or } s_1(s_3s_1)^n s_0
  \]

  \[
  \sum_{n=0}^{\infty} 0.01^n \cdot (0.01 + 0.98) = \frac{1}{1-0.01} 0.99 = 1
  \]

  Thus \( s_1 \models [\Diamond \text{init}]_{\geq 1} \)
More examples

- \([\Diamond \leq^k \text{init}] \geq 0.99\):
  
  \(\text{init}\) is reached within \(k\) steps with probability 0.99

  \(s_1 \not\models [\Diamond \leq^0 \text{init}] \geq 0.99\) (probability is 0)

  \(s_1 \not\models [\Diamond \leq^1 \text{init}] \geq 0.99\) (probability is 0.98)

  \(s_1 \models [\Diamond \leq^2 \text{init}] \geq 0.99\) (probability is 0.98 + 0.01)

  \(s_1 \models [\Diamond \leq^3 \text{init}] \geq 0.99\) (probability is 0.99 + 0.98 \cdot 0.01)

  ...

- \([\text{trying } \mathcal{U} \text{ init}] > 0.9\), where \(\text{trying} \downarrow \text{snd} \lor \text{lost}\):

  Again, summing up over paths:

  \(s_1s_0\)

  \(s_1s_3s_1s_0\)

  ...

  \(s_1(s_3s_1)^n s_0\)

\[
\sum_{n=0}^{\infty} 0.01^n \cdot 0.98 = \frac{1}{1-0.01} 0.98 = \frac{100}{99} \cdot \frac{98}{100} = \frac{98}{99}
\]

Thus, \(s_1 \models [\text{trying } \mathcal{U} \text{ init}] > 0.9\).
# Model Checking for PCTL

Let $M = (S, P, L)$ be a DTMC, $\phi$ a PCTL formula.

The model checking algorithm proceeds as for CTL by induction on $\phi$.

<table>
<thead>
<tr>
<th>Input: DTMC $M = (S, P, L)$, $\phi$ a PCTL formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: $Sat(\phi) = {s \in S \mid M, s \models \phi}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>case $\phi$ of</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>true:</strong></td>
<td>return $S$</td>
</tr>
<tr>
<td>$a$:</td>
<td>return ${s \in S \mid a \in L(s)}$</td>
</tr>
<tr>
<td>$\phi_1 \land \phi_2$:</td>
<td>return $Sat(\phi_1) \cap Sat(\phi_2)$</td>
</tr>
<tr>
<td>$\neg \psi$:</td>
<td>return $S \setminus Sat(\psi)$</td>
</tr>
<tr>
<td>$[\alpha]^{\diamond p}$:</td>
<td>for each $s$</td>
</tr>
<tr>
<td>$x_s := Prob{\pi \mid M, \pi \models \alpha, \pi \text{ starts in } s}$</td>
<td></td>
</tr>
<tr>
<td>return ${s \in S \mid x_s \bowtie p}$</td>
<td></td>
</tr>
</tbody>
</table>
How to Calculate the Probabilities

Let $M = (S, P, L)$ be a DTMC, $\phi$ a PCTL formula.

Represent
- DTMC as its matrix $P$
- $Sat(\phi)$ as a column vector $b^\phi : S \rightarrow \{0, 1\}$
given by

$$b^\phi_s = \begin{cases} 1 & \text{if } M, s \models \phi \\ 0 & \text{otherwise} \end{cases}$$

For the protocol example, the matrix $P$ and the vector $b^{error}$ for $Sat(error)$ are:

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0.98 & 0 & 0.01 & 0.01 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad b^{error} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$
Model Checking Next State

- Computing probability of paths satisfying $X\phi$:
  for each $s \in S$,
  sum up the probabilities of transitions to $\phi$-states

\[
x_s = \sum_{t \in \text{Sat}(\phi)} P(s, t)
\]

- Equivalently, express as matrix-by-vector multiply

\[
x = P \cdot b^\phi
\]

where
- $P$ is the matrix and
- $b^\phi$ is the column vector for $\text{Sat}(\phi)$. 
Linear equations

- Consider the path formula $\Diamond init$:

![Diagram of states]

- Derive recursive linear equations
  - let $x_s$ denote the probabilities, for $s \in S$
  - set the probability for $init$-state to 1, i.e. $x_0 = 1$
  - write down flow equations

  \[
  \begin{align*}
  x_0 &= 1 \cdot x_1 \\
  x_1 &= 0.01 \cdot x_2 + 0.01 \cdot x_3 + 0.98 \cdot x_0 \\
  x_2 &= 1 \cdot 1 \\
  x_3 &= 1 \cdot x_1
  \end{align*}
  \]

- Solving the equations yields unique solution

  \[x_1 = 1 = x_0 = x_3 = x_2\]
Equations for PCTL Until

• Consider \([\phi_1 \bigcirc \phi_2]_{\geq 0}\). Calculate:

\[ S^{YES} := Sat(\phi_2) \]
\[ S^{>0} := Reach(\phi_1, \phi_2) \text{ (states from which one can reach } \phi_2\text{-state via } \phi_1\text{-states)} \]
\[ S^{NO} := S \setminus S^{>0} \]
\[ S' := S \setminus (S^{NO} \cup S^{YES}) \]

and define

\[
x_s = \begin{cases} 
0 & \text{if } s \in S^{NO} \\
1 & \text{if } s \in S^{YES} \\
\sum_{t \in S} P(s, t) \cdot x_t & \text{if } s \in S' 
\end{cases}
\]

• Obtain a set of linear equations in \(|S|\) unknowns, simplify to a set of equations in \(|S'|\) unknowns:

\[
x_s = \sum_{t \in S'} P(s, t) \cdot x_t + \sum_{t \in S^{YES}} P(s, t)
\]
Solving Equations for PCTL Until

- Equivalently, in matrix form:

\[ x = A \cdot x + P \cdot b \]

where
\[ A \] is the submatrix for \( S^p \) and
\[ b \] a column vector for \( S^{YES} \).

- Solve through direct method:
  - Gaussian elimination
  - LU decomposition
  - etc

- or via iteration:

\[
\begin{align*}
x_s^0 &= 0 \\
x_s^{n+1} &= \sum_{t \in S^p} P(s, t) \cdot x_t^n + \sum_{t \in S^{YES}} P(s, t)
\end{align*}
\]
Example: linear equations

• Consider \( \Psi = [trying \cup init]_{>0.9} \), where
  \[ trying \stackrel{\text{def}}{=} snd \lor lost. \]

• We calculate:
  \[
  S^{YES} = \{s_0\}
  \]
  \[
  S^{NO} = \{s_2\}
  \]
  \[
  S^? = \{s_1, s_3\}
  \]

• Obtain the linear equations:
  \[
  x_1 = 0.98 + 0.01 \cdot x_3 \\
  x_3 = x_1
  \]

• which yield the solution:
  \[
  \mathbf{x} = (1, \frac{98}{99}, 0, \frac{98}{99})
  \]

• Hence
  \[
  Sat(\Psi) = \{s_0, s_1, s_3\}
  \]
Some Special Cases

- **Probabilistic reachability**
  - reaching a target set $T$ of states with probability $\propto p$
  - expressible as

  $\Diamond a_T \propto p$

  where $a_T$ is an atomic proposition true at $s$ iff $s \in T$

- **Probabilistic invariance**
  - remaining in set $I$ of states with probability $\propto p$
  - expressible as

  $\Box a_I \propto p$

  where $a_I$ is an atomic proposition true at $s$ iff $s \in I$
More Special Cases

- Qualitative properties
  - are expressible as

\[ [\alpha] \geq 1 \]

for \( \alpha \) a path formula
- reachability analysis of the underlying graph suffices
- can improve the model checking algorithm by precomputing:

\[ S^{YES} := \text{set of all states satisfying } \alpha \text{ with probability 1} \]

- Likewise

\[ [\alpha] > 0 \]

can be established by reachability analysis
Markov Decision Processes

Markov Decision Processes:
- more general than DTMCs
- non-deterministic choice between (discrete) probability distributions on successor states.
- also called concurrent probabilistic systems

Let $\mu(S)$ denote the set of probability distributions on $S$. A Markov Decision Process (MDP) is $M = (S, Steps, L)$ where
- $S$ is a set of states
- $Steps(s) \subseteq \mu(S)$ for each $s \in S$
- $L : S \rightarrow 2^{AP}$ labelling with atomic prop.

Link arrows of the same probability distribution by an arc.
Paths and adversaries

Let $M = (S, \text{Steps}, L)$ be a MDP.

- **The execution of $M$**
  - alternation of non-deterministic and probabilistic choices
  - non-deterministic choice resolved using adversaries (similar to non-randomized policies in MDPs).

**Paths** $\pi$ of $M$ are (finite or infinite) sequences of states

$$s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} s_2 \xrightarrow{\mu_2} \cdots$$

where $s_i \in S$, $\mu_i \in \text{Steps}(s_i)$ and $\mu_i(s_{i+1}) > 0$.

- **An adversary $A$**
  - maps every finite path $\rho$ onto a distribution (the step to be executed) $A(\rho)$ such that

$$A(\rho) \in \text{Steps}([\rho])$$

- induces probability measure $Prob^A$ on the $\sigma$-algebra $\mathcal{F}^A(s)$ of infinite paths corresponding to $A$.  

26
Paths and adversaries: example

- Unfolding the MDP

into a computation tree from $s_0$, obtain

- Adversary must decide:
  - at $s_0$, whether to execute $\mu_1$ or $\mu_2$,
  - at $s_0s_2s_0$, whether to execute $\mu_1$ or $\mu_2$,
  - etc
PCTL over Markov Decision Processes

(Bianco&de Alfaro 1995), (Baier & Kwiatkowska 1998)

Based on PCTL over DTMCs + quantification over adversaries.

The syntax is the same as PCTL over DTMCs.

Semantics is parameterised by \(Adv\), a class of adversaries, for example, all or all fair adversaries.

The meaning of the probabilistic operator is:

\[
M, s \models_{Adv} [\alpha]_{\bowtie p} \iff \text{Prob}^A\{\pi \mid M, \pi \models_{Adv} \alpha, \pi \text{ starts in } s\} \bowtie p \text{ for all adversaries } A \in Adv
\]

with the intuition is that probability bound is enforced under any scheduling policy of non-deterministic choices.

The other operators are as for DTMCs.
Model checking for PCTL over MDPs

Let $M = (S, \text{Steps}, L)$ be a MDP, $\phi$ a PBTL formula.

Assume $Adv$ is the class of all adversaries.

- **The algorithm**
  - proceeds by induction on the structure of $\phi$
  - as for DTMCs except for probability calculation

- **Probability calculation**
  - for path formula $\alpha$, calculate minimum probability
    for all $s \in S$:

  $$p_{s}^{\min}(\alpha) = \inf \{ p_{s}^{A}(\alpha) \mid A \in Adv \}$$

  where

  $$p_{s}^{A}(\alpha) = Prob^{A}\{ \pi \mid M, \pi \models_{Adv} \alpha, \pi \text{ starts in } s \}$$

- **Establish validity of the probability bound**

  $$Sat([\alpha]_{\bowtie p}) = \{ s \in S \mid p_{s}^{\min}(\alpha) \bowtie p \}$$
Model checking Next State over MDPs

- **Note**
  - there are **infinitely many adversaries** even if $S$ finite
  - suffices to consider **simple adversaries** (stationary policies in MDP)

- **Thus**
  - the **minimum probability** for $X \phi$ is

\[
p^\text{min}_s(X \phi) := \min\{\sum_{t \in \text{Sat}(\phi)} \mu(t) \mid \mu \in \text{Steps}(s)\}
\]

- can be computed through **matrix-by-vector** multiplication plus the **min function**
Model checking Until over MDPs

- First calculate
  - $S^{YES}$, $S^{NO}$ and $S'$ as for DTMCs
  - the minimum probability vector rewrites to:

\[
p_s^{\text{min}} = \begin{cases} 
  0 & \text{if } s \in S^{NO} \\
  1 & \text{if } s \in S^{YES} \\
  \min_{\mu \in \text{Steps}(s)} \left[ \sum_{t \in S'} \mu(t) \cdot p_t^{\text{min}} + \sum_{t \in S^{YES}} \mu(t) \right] & \text{if } s \in S'
\end{cases}
\]

- Next, either
  - evaluate $p_s^{\text{min}}$ iteratively for $s \in S'$,
  or
  - solve the linear optimization problem:

Maximise:
\[
\sum_{s \in S'} x_s
\]
Subject to the constraints:
\[
x_s \leq \sum_{t \in S'} \mu(t) \cdot x_t + \sum_{t \in S^{YES}} \mu(t)
\]
for all $s \in S'$ and all $\mu \in \text{Steps}(s)$

(Think of balancing flow, replacing the equality with inequality.)
Consider the MDP:

\[ \phi = [a \cup b]_{\geq \frac{1}{4}}. \]

Calculate

- \( S^{YES} = \{z, v\} \)
- \( S^{>0} = \{s, r, t, z, v\} \)
- \( S^{NO} = \{w, u\} \)
- \( S^? = \{s, r, t\} \)
Example linear optimisation

Consider again the example from the previous slide. The linear programming problem for

$$\phi = \forall [a \cup b]_{> \frac{1}{4}}$$

is:

Maximise:

$$x_r + x_s + x_t$$

subject to the constraints:

$$x_r \leq x_r \quad (\mu_1)$$
$$x_r \leq x_t \quad (\mu_2)$$
$$x_s \leq \frac{1}{4} \quad (\nu_1)$$
$$x_s \leq \frac{1}{2} x_t \quad (\nu_2)$$
$$x_t = \frac{1}{2} x_s + \frac{1}{2} \quad (\rho)$$

It yields

$$x_s = \frac{1}{4}, \; x_r = 0, \; x_t = \frac{5}{8}$$

as optimal.

Thus,

$$t \in Sat([a \cup b]_{> \frac{1}{4}}) \quad s, \; r \notin Sat([a \cup b]_{> \frac{1}{4}})$$

(under any scheduling policy).
**Probabilistic LTL**

(Vardi 1985), (Courcoubetis & Yannakakis 1988)

- The syntax is:

\[
\alpha ::= \text{true} \mid a \mid \alpha_1 \land \alpha_2 \mid \neg \alpha \mid X \alpha \mid \alpha_1 \cup \alpha_2
\]

with the usual interpretation \( \pi \models \alpha \) for paths \( \pi \)

- Interpretation over a nonprobabilistic system

\[
M, s \models \alpha \quad \text{iff} \quad \forall \pi \text{ starting in } s. M, \pi \models \alpha
\]

- Interpretation over a DTMC \( M = (S, P, L) \)

\[
M, s \models [\alpha]_{\bowtie p} \quad \text{iff} \quad Prob\{\pi \mid M, \pi \models \alpha, \pi \text{ starts in } s\} \bowtie p
\]

- Interpretation over a MDP \( M = (S, Steps, L) \)

\[
M, s \models_{Adv} [\alpha]_{\bowtie p} \quad \text{iff} \quad \text{Prob}^A\{\pi \mid M, \pi \models_{Adv} \alpha, \pi \text{ starts in } s\} \bowtie p
\]

for all adversaries \( A \in Adv \)
Example: LTL vs Probabilistic LTL

Consider the DTMC modelling a simple probabilistic protocol:

\[ s_1 \not\models \beta \quad \text{in the classical sense, since} \exists \text{an infinite path from } s_1 \text{ which invalidates } \Diamond \text{init} \]

\[ s_1 \not\models [\beta] \geq 1 \quad \text{in the probabilistic sense} \]
Model checking of Probabilistic LTL

- Main difficulty: conjunctions
  - consider a fixed state $s$ and $\Diamond a \land \Diamond b$
  - calculate $\Pr\{\pi \text{ from } s \mid \pi \models \Diamond a\}$
  - and $\Pr\{\pi \text{ from } s \mid \pi \models \Diamond \alpha\}$
  - but what is $\Pr\{\pi \text{ from } s \mid \pi \models \Diamond a \land \Diamond b\}$?

  depends on overlap of the two sets of paths

- Idea (Courcoubetis & Yannakakis 1988)
  - successive transformation of the formula $\alpha$ into propositional formula $\hat{\alpha}$ and DTMC $M$ into $\hat{M}$
  - while preserving the probability of satisfaction:

  $$M, s \models [\alpha]_{\bowtie p} \iff \hat{M}, \hat{s} \models [\Diamond \hat{\alpha}]_{\bowtie p}$$

- Thus
  - reduce to PCTL model checking
  - obtain a larger DTMC
Model checking of Probabilistic LTL ctd

- **Transformation:**
  - eliminate temporal connectives from \( \alpha \)
  - replace \( \Diamond \beta \) with a **new** atomic proposition \( b \)
  - split every state into two **copies** \( \langle s, b \rangle, \langle s, \neg b \rangle \)
  - calculate probability \( p := \text{Prob}\{ \pi | \pi \models \Diamond \beta \} \)
  - continue, splitting as necessary, for remaining connectives

\[
\begin{align*}
&\langle s, b_1, b_2 \rangle & & q \\
&\langle s, b_1 \rangle & & q \\
&\langle s, \neg b_1 \rangle & & 1 - p \\
&s & & 1 - q \\
&\langle s, b_1, \neg b_2 \rangle & & 1 - q \\
&\langle s, \neg b_1, b_2 \rangle & & 1 - q \\
&\langle s, \neg b_1, \neg b_2 \rangle & & p \cdot q \\
\end{align*}
\]

\[
\text{Prob}\{ \pi \text{ from } s | \pi \models \Diamond a \land \Diamond b \} = p \cdot q
\]

- **Method**
  - extends to CTL*, MDPs
Additional comments

- Probabilistic reachability and invariance
  - are special cases as for DTMCs

- Qualitative properties
  - via reachability analysis of the graph
  - yield improvement through precomputation

- Fairness
  - essential for model checking liveness
  - reducible to linear optimisation

- LTL/PCTL* model checking
  - via transformation of the underlying MDP
  - more expensive

- Continuous Time Markov Chains
  - have associated logic CSL
    (Baier, Katoen & Hermanns 1999)
  - and tool (Hermanns, Katoen, Meyer-Kayser & Siegle 2000)
  - can express time bounded until and steady state
Feasibility issues

- PCTL model checking is feasible
  - complexity polynomial in the size of the state space
  and exponential in the size of the formula
  - soluble iteratively or through e.g. the ellipsoid method

- Techniques applicable in more general context
  - e.g. real-time systems, cf probabilistic
    timed automata (Kwiatkowska, Norman,
    Segala & Sproston 2000)

- Implementation
  - via linear algebra
  - sparse matrix packages
  - variants of BDDs (Binary Decision Diagrams)
The PRISM Tool

- **Probabilistic Symbolic Model Checker**
  - experimental tool
  - being developed at Birmingham

- **Features:**
  - Markov Decision Processes
  - PCTL model checking
    (including fairness)
  - system description language

- **Details:**
  - symbolic (based on MTBDDs)
  - written in Java, C++
  - uses CUDD [Somenzi]

- **http://www.cs.bham.ac.uk/~dxp/prism**
System Description Language

• Motivation:
  - front end for tool
  - facilitate case study construction
  - efficient translation into MTBDDs

• Similar to:
  - Reactive Modules [Alur, Henzinger, ...]
  - SANs [Plateau, ...]

• Basic idea:
  - asynchronous concurrent composition
    of a set of \( n \) modules

• Each module:
  - has a set of local variables
  - can write its own variables
  - can read variables of all modules

• A module’s behaviour:
  - can be probabilistic
  - can be nondeterministic
  - can depend on the global state of the system
System Description Language...

- Formally:
  - $n$ modules: $\mathcal{M} = \{M_1, \ldots, M_n\}$
  - local state spaces: $S_1, \ldots, S_n$
  - global state space: $S = S_1 \times \cdots \times S_n$
  - local behaviour of $M_i$: $S \rightarrow 2^{\mu(S_i)}$
  - global behaviour of $\mathcal{M}$: $S \rightarrow 2^{\mu(S)}$
    (asynchronous scheduling of modules)

- Additional features (omitted from presentation):
  - global variables (shared memory)
  - synchronization (communication, message passing)
  - rates or probabilities (continuous time)
  - language features (constants, macros, etc.)
Example

- Symmetric two process mutual exclusion

module $M_1$
  \[ x : [1..3]; \]
  \[ (x = 1) \rightarrow 0.8 : (x' = 1) + 0.2 : (x' = 2); \]
  \[ (x = 2) \land (y = 3) \rightarrow (x' = 2); \]
  \[ (x = 2) \land (y \neq 3) \rightarrow (x' = 3); \]
  \[ (x = 3) \rightarrow (x' = 3); \]
  \[ (x = 3) \rightarrow (x' = 1); \]
endmodule

module $M_2$
  \[ y : [1..3]; \]
  \[ (y = 1) \rightarrow 0.8 : (y' = 1) + 0.2 : (y' = 2); \]
  \[ (y = 2) \land (x = 3) \rightarrow (y' = 2); \]
  \[ (y = 2) \land (x \neq 3) \rightarrow (y' = 3); \]
  \[ (y = 3) \rightarrow (y' = 3); \]
  \[ (y = 3) \rightarrow (y' = 1); \]
endmodule
Case Studies

- Randomized distributed algorithms
  - Randomized dining philosophers
    - Lehmann, Rabin ’82
    - Pnueli, Zuck ’86
  - N-Process mutual exclusion
    - Rabin ’82
    - Pnueli, Zuck ’86
  - Randomized consensus protocol
    - Aspnes, Herlihy ’90

- Continuous time models
  - Kanban network
    - Ciardo, Tilgner ’96
  - Tandem queueing network
    - Hermanns, Meyer-Kayser, Siegle ’99
  - Cyclic queue polling system
    - Ibe, Trivedi ’90
Modelling probability and real-time

- **Aim:**
  - Continuous real-time
  - Probabilistic and nondeterministic choice
  - Amenable to model checking:
    - w.r.t. quantitative, probabilistic properties

- **Extend timed automata (TA) with:**
  - discrete probability distributions
  - Strategy: associate probability with discrete transitions
  - Probabilistic Timed Automata (PTA)

- **systems:** likelihoods $\leftrightarrow$ discrete state changes
  - e.g. lossy media, fault-tolerant systems
- In **send**: message arrives after 4/before 5 ms
- Probability of 0.01 of message loss
- System **waits** for 3 ms after successful delivery
Probabilistic timed automata

- Clock constraints: $CC_X$
  - $x \leq 3$, $y < x + 1 \land y \geq 2$
  - $\zeta :\! = \! x \leq c \lor x \geq c \lor x - y \leq c \lor x - y \geq c \lor \neg \zeta \lor \zeta \lor \zeta$
  - where $x, y \in \mathcal{X}$ and $c \in \mathbb{N}$

- $T = (\mathcal{S}, L, s_{\text{init}}, \mathcal{X}, \text{inv}, \text{prob}, \langle \tau_s \rangle_{s \in \mathcal{S}})$
  
  \[
  \begin{array}{l|l}
  \mathcal{S} & \text{finite set of nodes} \\
  L : \mathcal{S} \longrightarrow 2^{\mathcal{AP}} & \text{labelling function} \\
  s_{\text{init}} \in \mathcal{S} & \text{start node} \\
  \mathcal{X} & \text{finite set of clocks} \\
  \text{inv} : \mathcal{S} \longrightarrow CC_X & \text{invariant condition} \\
  \text{prob} : \mathcal{S} \rightarrow \mathcal{P}_{fn}(\mu(\mathcal{S} \times 2^{\mathcal{X}})) & \text{probability distn.s} \\
  \langle \tau_s \rangle_{s \in \mathcal{S}} & \text{enabling conditions} \\
  \\
  - Definition of $\tau_s$: \\
  \quad o \text{ for any } s \in \mathcal{S}, \tau_s : \text{prob}(s) \longrightarrow CC_X
  \\

- Underlying structure: infinite-state Markov Decision Process (MDP)
  - infinite nondeterministic branching
  - finite probabilistic branching
Probabilistic real-time properties

- e.g. with probability 0.9875 or greater, the message is correctly delivered within 5 ms
  - temporal logic: $z.[\Diamond (\text{deliver} \land z < 5)] \geq 0.9875$

\[
\begin{align*}
\cdots \Diamond \text{deliver} \cdots & \quad \text{temporal modality} & \quad \text{CTL} \\
z.\cdots z < 5 \cdots & \quad \text{exact time requirements} & \quad \text{TCTL} \\
\cdots [\cdots] \geq 0.9875 & \quad \text{probabilistic operator} & \quad \text{PCTL}
\end{align*}
\]

- PTCTL syntax:
  \[
  \phi ::= \text{true} \mid a \mid \zeta \mid \phi \land \phi \mid \neg \phi \\
  z.\phi \mid [\phi U \phi]_{\pi_0, \lambda}
  \]

  where $a \in \text{AP}$, $\zeta \in \text{CC}, z \in Z$ where $Z$ are formula clocks, and $\lambda \in [0, 1]$,
  - with abbreviations: e.g. $\Diamond \text{a} \equiv \text{true} U \text{a}$

- Reachability
  - e.g. with probability 0.99 or greater, error is reached
  - express as $(\{\text{error}\}, \geq, 0.99)$
  - applications: specification of invariants or safety properties
Region equivalence and PTA

- **Region equivalence:**
  - induces finite partition of (P)TA state space
  - equivalent states: satisfy the same clock constraints
    - *now*, and
    - *after some time passage*

- **Finitary:** for each node
  - partitions each *unit* hypercube in \( \mathbb{R}^{|X \cup \mathcal{Z}|} \) into a finite set of equivalence classes
  - *bounded* by maximal constant of PTA and formula

- **Prove:**
  - region equivalent states satisfy the same PTCTL formulas
Probabilistic region graph

- Region equivalence $\Rightarrow$ decidable model checking

- **TA:**
  - derive transition relation between region equivalence classes
  - obtain a finite labelled transitions system
  - model check using established techniques

- **PTA:**
  - derive distributions over transitions between region equivalence classes
  - obtain a finite concurrent probabilistic system
  - model check using established techniques

- **Require:** transformation PTCTL $\rightarrow$ PBTL

- **Problem:** complexity
  - exponential in the number of clocks
Summary and Outlook

- **Markov Decision Processes**
  - generalise Discrete Time Markov Chains
  - PCTL model checking
    - implemented in the PRISM tool
    - numerical computation is a bottleneck
    - LTL model checking useful
  - more case studies needed

- **Continuous Time Markov Chains**
  - temporal logic CSL and tool
  - no non-determinism
  - allow steady state analysis

- **Probabilistic Timed Automata**
  - the logic PTCTL
    - appropriate for soft deadlines
  - discrete probability distributions
    - region graph, forward/backward reachability
    - no implementation yet
  - continuous probabilities
    - region graph approach
    - high complexity