Probabilistic Computation Tree Logic

- PCTL is a language for formally specifying properties over DTMCs.
- It can also be used to specify properties over MDPs.
- It is a branching-time temporal logic based on CTL.
- Formula interpretation is Boolean, i.e., a state satisfies a formula or not.
- The main operator is $P_J(\varphi)$
  - where $\varphi$ constrains the set of paths and $J$ is a threshold on the probability.
  - it is the probabilistic counterpart of $\exists$ and $\forall$ path-quantifiers in CTL.
  - ranges over all possible resolutions of nondeterminism.

PCTL Syntax

[Bianco & De Alfaro, 1995]

Probabilistic Computation Tree Logic: Syntax

PCTL consists of state- and path-formulas.
- PCTL state formulas over the set $AP$ obey the grammar:
  \[
  \Phi ::= \text{true} \mid a \mid \Phi_1 \land \Phi_2 \mid \neg \Phi \mid P_J(\varphi)
  \]
  where $a \in AP$, $\varphi$ is a path formula and $J \subseteq [0,1]$, $J \neq \emptyset$ is a non-empty interval.
- PCTL path formulae are formed according to the following grammar:
  \[
  \varphi ::= \bigcirc \Phi \mid \Phi_1 U \Phi_2 \mid \Phi_1 U^{\leq n} \Phi_2
  \]
  where $\Phi, \Phi_1$, and $\Phi_2$ are state formulae and $n \in \mathbb{N}$.
  Abbreviate $P_{[0,0.5]}(\varphi)$ by $P_{\leq 0.5}(\varphi)$ and $P_{[0,1]}(\varphi)$ by $P_{>0}(\varphi)$.

Intuitive semantics

- $s_0 \alpha_0 s_1 \alpha_1 s_2 \ldots \models \Phi U^{\leq n} \Psi$ if $\Phi$ holds until $\Psi$ holds within $n$ steps (where $s_0 \alpha_0$ is a single step).
- $s \models P_J(\varphi)$ if probability under all policies that paths starting in $s$ fulfill $\varphi$ lies in $J$. 
Overview

1. PCTL Semantics
2. PCTL Model Checking
3. Complexity
4. Example: Dining Cryptographers Problem
5. Fairness
6. Summary

PCTL Semantics

Markov decision process (MDP)

An MDP $\mathcal{M}$ is a tuple $(S, \text{Act}, P, \iota_{\text{init}}, AP, L)$ where
- $S$ is a countable set of states with initial distribution $\iota_{\text{init}} : S \to [0, 1]$
- $\text{Act}$ is a finite set of actions
- $P : S \times \text{Act} \times S \to [0, 1]$, transition probability function such that:
  $$\sum_{s' \in S} P(s, \alpha, s') \in \{0, 1\}$$
- $AP$ is a set of atomic propositions and labeling $L : S \to 2^{AP}$.

Semantics of $\mathbb{P}$-operator

The probabilistic operator $\mathbb{P}_J(\cdot)$ imposes probability bounds for all policies. In particular, we have
- $s \models \mathbb{P} \leq p(\varphi)$ if $P^\max(s \models \varphi) \leq p$
- $\sup_S P_r^\mathbb{S}(s \models \varphi)$
- $s \models \mathbb{P} \geq p(\varphi)$ if $P^\min(s \models \varphi) \geq p$
- $\inf_S P_r^\mathbb{S}(s \models \varphi)$

For finite MDPs we have:
- $P_r^\max(s \models \varphi) = \max_S P_r^\mathbb{S}(s \models \varphi)$
- $P_r^\min(s \models \varphi) = \min_S P_r^\mathbb{S}(s \models \varphi)$

since for any finite MDP there exists an fm-policy that maximises or minimises $\varphi$. 

Notation

The satisfaction relation $\models$ is defined for PCTL state formulas by:

$$s \models a \iff a \in L(s)$$
$$s \models \neg \varphi \iff \text{not } (s \models \varphi)$$
$$s \models \varphi \land \psi \iff (s \models \varphi) \text{ and } (s \models \psi)$$
$$s \models P_J(\varphi) \iff \text{ for all policies } \mathcal{G} \text{ on } \mathcal{M}. P^\mathcal{G}_r(s \models \varphi) \in J$$

where $P^\mathcal{G}_r(s \models \varphi) = P_{s^\mathcal{G}} \{ \pi \in \text{Paths}(s) \mid \pi \models \psi \}$. 

Satisfaction relation for state formulas

Joost-Pieter Katoen
Modeling and Verification of Probabilistic Systems
Satisfaction relation for path formulas

Let \( \pi = s_0 \alpha_0 s_1 \alpha_1 s_2 \alpha_2 \ldots \) be an infinite path in (possibly infinite) MDP \( M \). Recall that \( \pi[i] = s_i \) denotes the \((i+1)\)-st state along \( \pi \).

The satisfaction relation \( \models \) is defined for state formulas by:

\[
\pi \models \square \Phi \quad \text{iff} \quad s_1 \models \Phi
\]

\[
\pi \models \Phi \lor \Psi \quad \text{iff} \quad \exists k \geq 0. (\pi[k] \models \Psi \land \forall 0 \leq i < k. \pi[i] \models \Phi)
\]

\[
\pi \models \Phi \land \psi \quad \text{iff} \quad \exists k \geq 0. (k \leq n \land \pi[k] \models \psi \land \forall 0 \leq i < k. \pi[i] \models \Phi)
\]

There is indeed no difference with the PCTL semantics for DTMC paths.

---

### Overview

- PCTL Semantics
- PCTL Model Checking
- Complexity
- Example: Dining Cryptographers Problem
- Fairness
- Summary
Core model checking algorithm

Propositional formulas

Sat(\cdot) is defined by structural induction as for PCTL on DTMCs.

Probabilistic operator \( \mathbb{P} \)

In order to determine whether \( s \in \text{Sat}(\mathbb{P} \leq p(\Phi)) \), the probability \( P^{\text{max}}(s \models \varphi) \) needs to be established. Then

\[
\text{Sat}(\mathbb{P} \leq p(\varphi)) = \{ s \in S \mid P^{\text{max}}(s \models \varphi) \leq p \}.
\]

The same holds for strict upper bounds \(< p\).

Similarly, lower bounds amount to determining \( P^{\text{min}}(s \models \varphi) \), e.g.,

\[
\text{Sat}(\mathbb{P} > p(\varphi)) = \{ s \in S \mid P^{\text{min}}(s \models \varphi) > p \}.
\]

Example

Consider MDP:

\[
\text{[init]} 0.5 0.3 \quad \text{[tails]}
\]

\[
\text{[heads]} \quad a \quad 1 \quad 0.5 \quad b \quad c \quad 0.5 \quad a
\]

and PCTL-formula:

\[
\mathbb{P} \geq \frac{1}{2} (\bigcirc \text{heads})
\]

1. \( \text{Sat(\text{heads})} = \{ s_2 \} \)
2. \( x_s = P^{\text{min}}(s_1 \models \bigcirc \text{heads}) = \min(0, 0.5) = 0 \)
3. Applying that to all states yields:

\[
(P^{\text{min}}(s \models \bigcirc \Phi))_{s \in S} = \begin{pmatrix}
0 & 1 & 0 & 0 \\
0.7 & 0.3 & 0 & 0 \\
0 & 0.5 & 0.5 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{pmatrix} \cdot \begin{pmatrix}
0 \\
0 \\
0.5 \\
1 \\
0
\end{pmatrix} = \begin{pmatrix}
0 \\
0.5 \\
0 \\
1 \\
0
\end{pmatrix}
\]

4. Thus: \( \text{Sat}(\mathbb{P} \geq 0.5(\bigcirc \text{heads})) = \{ s_2 \} \).

The next-step operator

Recall that: \( s \models \mathbb{P} \leq p(\bigcirc \Phi) \) if and only if \( P^{\text{max}}(s \models \bigcirc \Phi) \leq p \).

Lemma

\[
P^{\text{max}}(s \models \bigcirc \Phi) = \max \{ \sum_{t \in \text{Sat}(\Phi)} P(s, \alpha, t) \mid \alpha \in \text{Act}(s) \}.
\]

Algorithm

Determine \( x_s = P^{\text{max}}(s \models \bigcirc \Phi) \) and return \( \text{Sat}(\Psi) = \{ s \in S \mid x_s \leq p \} \).

The case for minimal probabilities is similar and omitted here.

Bounded until (1)

Recall that: \( s \models \mathbb{P} \geq p(\Phi \bigcup^n \Psi) \) if and only if \( P^{\text{min}}(s \models \Phi \bigcup^n \Psi) \geq p \).

Lemma

\[
P^{\text{min}}(s \models \Phi \bigcup^n \Psi) = \begin{cases}
1 & \text{if } s \in S_{\text{Sat}(\Psi)} \\
0 & \text{if } s \in S_{\text{Sat}(\Phi)} \\
0 & \text{if } s \in S_\text{tail} \land n=0 \\
\min \{ \sum_{s' \in S} P(s, \alpha, s') \cdot P^{\text{min}}(s' \models \Phi \bigcup^{n-1} \Psi) \mid \alpha \in \text{Act}(s) \} & \text{otherwise}
\end{cases}
\]

The case for maximal probabilities is analogous.
Bounded until (2)

**Lemma**

Let \( S_{1} = \text{Sat}(\psi) \), \( S_{0} = S \setminus (\text{Sat}(\phi) \cup \text{Sat}(\psi)) \), and \( S_{i} = S \setminus (S_{i-1} \cup S_{i+1}) \).

Then: \( P_{\text{min}}^{\phi}(s \models \phi \cup \psi) \) equals

\[
\begin{cases}
1 & \text{if } s \in S_{1} \\
0 & \text{if } s \in S_{0} \\
0 & \text{if } s \in S_{i} \land n=0 \\
\min\left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot P_{\text{min}}^{\phi}(s' \models \phi \cup \psi) \mid \alpha \in \text{Act}(s) \right\} & \text{otherwise}
\end{cases}
\]

**Algorithm**

1. Let \( P_{\phi, \psi} \) be the probability matrix of \( M[S_{0} \cup S_{1}] \).
2. Then \( (P_{\text{min}}^{\phi}(s \models \phi \cup \psi))_{s \in S} = b_{\psi} \).
3. And \( (P_{\text{min}}^{\phi}(s \models \phi \cup \psi))_{s \in S} = P_{\phi, \psi} \cdot (P_{\text{min}}^{\phi}(s \models \phi \cup \psi))_{s \in S} \).
4. This requires \( n \) matrix-vector multiplications in total and \( n \) minimum operators.

**Until**

Recall that: \( s \models \Box_{G}(\phi \psi) \) if and only if \( P_{\text{min}}^{\phi}(s \models \phi \psi) \geq p \).

**Algorithm**

1. Determine \( S_{1} = \text{Sat}(P_{1}(\phi \psi)) \) by a graph analysis.
2. Determine \( S_{0} = \text{Sat}(P_{0}(\phi \psi)) \) by a graph analysis.
3. Then solve a linear program (or use value iteration) over all remaining states.

**Importance of pre-computation**

1. Determining \( S_{0} \) ensures unique solution of linear program.
2. Determining \( S_{1} \) reduces the number of variables in the linear program.
3. Gives exact results for the states in \( S_{1} \) and \( S_{0} \) (i.e., no round-off).
4. For qualitative properties, no further computation is needed.

**Example**

**Precomputations**

**Qualitative reachability**

1. Determine all states for which probability is zero
   1.1 minimum: \( \{ s \in S \mid P_{\text{min}}^{\phi}(s \models \phi \psi) = 0 \} \)
   1.2 maximum: \( \{ s \in S \mid P_{\text{max}}^{\phi}(s \models \phi \psi) = 0 \} \)
2. Determine all states for which probability is one
   2.1 minimum: \( \{ s \in S \mid P_{\text{min}}^{\phi}(s \models \phi \psi) = 1 \} \)
   2.2 maximum: \( \{ s \in S \mid P_{\text{max}}^{\phi}(s \models \phi \psi) = 1 \} \)
3. Then solve a linear program (or use value iteration) over all remaining states.

The first case has been treated in the previous lecture (for \( \Diamond G \)).
Qualitative reachability

- Goal is to compute \( \{ s \in S \mid Pr_{\text{max}}(s \models \Diamond G) = 1 \} \)
- First make all states in \( G \) absorbing, i.e., \( P(s, \alpha_s, s) = 1 \)
- Iteratively remove state \( t \) for which \( Pr_{\text{max}}(t \models \Diamond G) < 1 \)

Sketch of algorithm

1. Let \( U_0 = S \setminus \text{Sat}(\exists \Diamond G) \); this can be done by a graph analysis
2. Remove all actions \( \alpha \) from state \( u \) for which \( \text{Post}(s, \alpha) \cap U_0 \neq \emptyset \)
3. If after removal of actions \( \text{Act}(u) = \emptyset \), then remove state \( u \).
4. Repeat this procedure for all states, yielding the new MDP \( M' \).
5. As this may yield new states from which \( G \) repeat the above steps until all states can reach \( G \)

This procedure is quadratic in the size of the MDP.

Overview

- PCTL Semantics
- PCTL Model Checking
- Complexity
- Example: Dining Cryptographers Problem
- Fairness
- Summary

Time complexity

Let \( |\Phi| \) be the size of \( \Phi \), i.e., the number of logical and temporal operators in \( \Phi \).

Time complexity of PCTL model checking of MDPs

For finite MDP \( M \) and PCTL state-formula \( \Phi \), the PCTL model-checking problem can be solved in time

\[
O \left( \frac{1}{\varepsilon} \cdot O(y(\text{size}(M))) \cdot n_{\text{max}} \cdot |\Phi| \right)
\]

where \( n_{\text{max}} = \max \{ n \mid \psi_1 U^* \psi_2 \text{ occurs in } \Phi \} \) with and \( n_{\text{max}} = 1 \) if \( \Phi \) does not contain a bounded until-operator.
Overview

- PCTL Semantics
- PCTL Model Checking
- Complexity
- Example: Dining Cryptographers Problem
- Fairness
- Summary

Dining cryptographers problem

Dining cryptographer’s protocol

1. Each cryptographer flips an unbiased coin and only informs the cryptographer on the right of the outcome.
2. Each cryptographer states whether the two coins that it can see—the one it flipped and the one the left-hand neighbour flipped—are the same (agree) or different (disagree).

Caveat: if a cryptographer actually paid for the dinner, then it instead states the opposite (disagree if the coins are the same and agree if the coins are different).

Claim

An odd number of agree indicates that the master paid, while an even number indicates that a cryptographer paid.

Problem statement

Three cryptographers gather around a table for dinner.
- The waiter informs them that the meal has been paid by someone, who could be one of the cryptographers or their master.
- The cryptographers respect each other’s right to make an anonymous payment, but want to find out whether the master paid or not.

Is it possible to obtain this information without revealing the identity of the cryptographer that paid?

Example scenario in which master paid (left) or cryptographer A paid (right) and provides a misleading vote.
### Dining cryptographers problem

#### Dining cryptographer’s protocol

1. Each cryptographer flips an unbiased coin and only informs the cryptographer on the right of the outcome.
2. Each cryptographer states whether the two coins that it can see—the one it flipped and the one the left-hand neighbour flipped—are the same (agree) or different (disagree).

Caveat: if a cryptographer actually paid for the dinner, then it instead states the opposite (disagree if the coins are the same and agree if the coins are different).

#### Generalisation

The dining cryptographer’s protocol can be generalised to any number $N$ of cryptographer. Then:

- if $N$ is odd, then an odd number of agrees indicates that the master paid while an even number indicates that a cryptographer paid.
- if $N$ is even, then an even number of agrees indicates that the master paid while an odd number indicates that a cryptographer paid.

### Checking correctness

<table>
<thead>
<tr>
<th>$N$</th>
<th>master pays:</th>
<th>cryptographers pay:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time:</td>
<td>iterations:</td>
</tr>
<tr>
<td>3</td>
<td>0.028</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0.061</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>0.141</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>0.322</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>0.778</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>1.467</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>2.67</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>6.30</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>56.9</td>
<td>31</td>
</tr>
<tr>
<td>20</td>
<td>268</td>
<td>41</td>
</tr>
</tbody>
</table>

\[
pay \Rightarrow P_{=1} \left( \Diamond (\text{done} \wedge \text{par} = N\%2) \right) \wedge \neg pay \Rightarrow P_{=1} \left( \Diamond (\text{done} \wedge \text{par} \neq N\%2) \right).
\]

That is: if the master paid, the parity of the number of agrees matches the parity of $N$ and, if a cryptographer paid, it does not.

### MDP generation times

<table>
<thead>
<tr>
<th>$N$:</th>
<th>States:</th>
<th>Transitions:</th>
<th>Construction time (s):</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>286</td>
<td>585</td>
<td>0.001</td>
</tr>
<tr>
<td>4</td>
<td>1,733</td>
<td>4,580</td>
<td>0.01</td>
</tr>
<tr>
<td>5</td>
<td>9,876</td>
<td>32,315</td>
<td>0.03</td>
</tr>
<tr>
<td>6</td>
<td>54,055</td>
<td>211,566</td>
<td>0.07</td>
</tr>
<tr>
<td>7</td>
<td>287,666</td>
<td>1,312,045</td>
<td>0.11</td>
</tr>
<tr>
<td>8</td>
<td>1,499,657</td>
<td>7,813,768</td>
<td>0.22</td>
</tr>
<tr>
<td>9</td>
<td>7,695,856</td>
<td>45,103,311</td>
<td>0.34</td>
</tr>
<tr>
<td>10</td>
<td>39,005,611</td>
<td>253,985,650</td>
<td>0.52</td>
</tr>
<tr>
<td>11</td>
<td>115,533,171,626</td>
<td>1,128,394,416,085</td>
<td>3.27</td>
</tr>
<tr>
<td>20</td>
<td>304,287,522,253,461</td>
<td>3,962,386,180,540,340</td>
<td>13.48</td>
</tr>
</tbody>
</table>

The number of states and transitions in the MDP representing the model for the dining cryptographers problem with $N$ cryptographers.

### Checking anonymity

To verify anonymity – when a cryptographer pays then no cryptographer can tell who has paid – we check that any possible combination of agree and disagree has the same likelihood no matter which of the cryptographers pays. This needs to be checked for all $2^N$ possible outcomes. Above the results are listed for one possible outcome.
Overview

- PCTL Semantics
- PCTL Model Checking
- Complexity
- Example: Dining Cryptographers Problem
- Fairness
- Summary

Fairness

- A policy $\mathcal{G}$ is fair if for every state $s$, the probability under $\mathcal{G}$ of all fair paths from $s$ is one
- A fairness assumption is realizable in MDP $\mathcal{M}$ if there is some fair policy for $\mathcal{M}$
- Realizable fairness assumptions are irrelevant for maximal reachability probabilities (i.e., safety)
- They are however relevant for minimal reachability probabilities (i.e., liveness)
- Computing minimal reachability probabilities under strongly fair policies is reducible to computing maximal reachability probabilities

Summary

- PCTL is a variant of CTL with operator $\Phi = \mathbb{P}J(\varphi)$.
- PCTL model checking is performed by a recursive descent over $\Phi$.
- Checking whether $s \models \mathbb{P}^>p(\varphi)$ amounts to determine $P^\text{min}_s(\varphi)$.
- Checking whether $s \models \mathbb{P}^<p(\varphi)$ amounts to determine $P^\text{max}_s(\varphi)$.
- The next operator amounts to a single matrix-vector multiplication and a max/min.
- The bounded-until operator $U^{\leq n}$ amounts to $n$ matrix-vector multiplications + $n$ minimums (or maximums).
- The until-operator amounts to solving a linear inequation system.
- The worst-case time complexity is polynomial in the size of the MDP and linear in the size of the formula.

Next lecture: Monday, June 20.