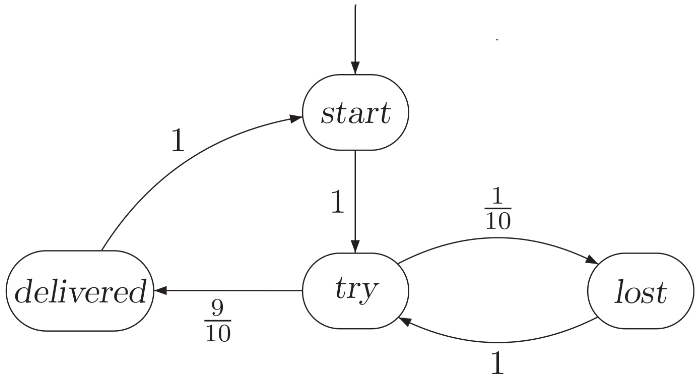
Probabilistic Model Checking

Stefan Pranger

02. 06. 2022

Communication Protocol



So far: Reachability Probabilities and how to compute them.

Today: More expressive properties!

Longterm Behaviour of MCs

"Fairness" from Probabilistic Behaviour

Every follow-up state u of $t \in S$ will be visited infinitely often, given t is visited infinitely often.

Longterm Behaviour of MCs

"Fairness" from Probabilistic Behaviour

Every follow-up state u of $t \in S$ will be visited infinitely often, given t is visited infinitely often.

Bottom Strongly Connected Components

We have heard about SCCs already

Recap: A *strongly connected component* is set of states, such that there is a path between any two states in the component.

A *bottom* SCC is an SCC such that no state outside the SCC is reachable.

This will come in quite handy later today!

Longterm Behaviour of MCs

From the two discussed properties we can deduce the following:

$$Pr\{\pi \in Paths(s) \mid \inf(\pi) \in BSCC(\mathcal{M})\} = 1$$

In words: Almost surely a BSCC will be reached and all of its states will be visited infinitely often.

Probabilistic Computation Tree Logic

Probabilistic Computation Tree Logic [PCTL] is the probabilistic extension of CTL.

- Boolean state representation.
- $\forall$ and $\exists$ are replaced by $\mathbb{P}_J(\varphi)$, where $J \subseteq [0, 1]$
  - The interpretation for each state $s \in S$: $Pr(\mathcal{M}, s \models \varphi) \in J$

PCTL - Syntax

Subdivision into *state* (Φ)- and *path*-formulae (φ):

$$\Phi ::= \mathit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathbb{P}_J(\varphi)$$

$$\varphi ::= \mathbf{X}\Phi \mid \Phi_1 \ \mathbf{U} \ \Phi_2 \mid \Phi_1 \ \mathbf{U}^{\leq n} \Phi_2$$

where $a \in AP$ and $J \subseteq [0,1]$.

PCTL - Satisfaction Relation

For a given state $s \in S$

$$
\begin{aligned}
s &\models a && \text{iff } a \in L(s), \\
s &\models \neg\varphi && \text{iff } s \not\models \varphi, \\
s &\models \varphi \wedge \psi && \text{iff } s \models \varphi \text{ and } s \models \psi, \\
s &\models \mathbb{P}_J(\varphi) && \text{iff } Pr(s \models \varphi) \in J
\end{aligned}
$$

For paths $\pi \in \mathcal{M}$:

$$
\begin{aligned}
\pi &\models \mathbf{X}\varphi && \text{iff} \quad \pi[1] \models \varphi \\
\pi &\models \varphi \,\mathbf{U}\, \psi && \text{iff} \quad \exists j \geq 0.\ (\pi[j] \models \psi \wedge (\forall 0 \leq k < j.\ \pi[k] \models \varphi) \\
\pi &\models \varphi \,\mathbf{U}^{\leq n} \psi && \text{iff} \quad \exists 0 \leq j \leq n.\ (\pi[j] \models \psi \wedge (\forall 0 \leq k < j.\ \pi[k] \models \varphi)
\end{aligned}
$$

PCTL - Semantics

$$s \models \mathbb{P}_J(\varphi) \text{ iff } Pr(s \models \varphi) \in J$$

We will use shorthand notations:

- $\mathbb{P}_{=1} = \mathbb{P}_{[1,1]}$
- $\mathbb{P}_{\geq 0.5} = \mathbb{P}_{[0.5,1]}$
- $\mathbb{P}_{>0} = \mathbb{P}_{(0,1]}$
- etc.

Model Checking PCTL

Similar to CTL model checking.

For each subformulae ψ of the parse tree we compute the satisfaction set $Sat(\psi)$ in a bottom-up manner.

$$
\begin{aligned}
Sat(true) &= S, \\
Sat(a) &= \{s \in S \mid a \in L(s)\}, \forall a \in AP, \\
Sat(\varphi \wedge \psi) &= Sat(\varphi) \cap Sat(\psi), \\
Sat(\neg\varphi) &= S \setminus Sat(\varphi).
\end{aligned}
$$

Model Checking PCTL

Similar to CTL model checking.

For each subformulae ψ of the parse tree we compute the satisfaction set $Sat(\psi)$ in a bottom-up manner.

$$\begin{aligned} Sat(true) &= S, \\ Sat(a) &= \{s \in S \mid a \in L(s)\}, \forall a \in AP, \\ Sat(\varphi \wedge \psi) &= Sat(\varphi) \cap Sat(\psi), \\ Sat(\neg\varphi) &= S \setminus Sat(\varphi). \end{aligned}$$

The interesting subformulae are of the form $\psi = \mathbb{P}_J(\varphi)$

In order to compute $Sat(\mathbb{P}_J(\varphi))$ we need to compute $Pr(s \models \varphi)$ for all $s \in S$, then

$$Sat(\mathbb{P}_J(\varphi)) = \{s \in S \mid Pr(s \models \varphi) \in J\}$$

Model Checking PCTL: $\mathbf{X}$-Operator

Model Checking PCTL: $\mathbf{X}$-Operator

A single matrix-vector multiplication

$$(Pr(\mathcal{M}, s \models \mathbf{X}\psi))_{s \in S} = \mathbf{A} \cdot \mathbf{b}_\psi$$

$$\text{where } \mathbf{b}_\varphi(s) = 1 \text{ iff } s \in Sat(\varphi)$$

$$\text{Sat}(\mathbb{P}_J(\mathbf{X}\varphi)) = \{s \in S \mid (\mathbf{A} \cdot \mathbf{b}_\varphi)(s) \in J\}$$

Model Checking PCTL: $\mathbf{U}^{\leq n}$-Operator

We want to compute

$$(Pr(\mathcal{M}, s \models \varphi\, \mathbf{U}^{\leq n} \psi))_{s \in S}$$

Model Checking PCTL: $\mathbf{U}^{\leq n}$-Operator

We want to compute

$$(Pr(\mathcal{M}, s \models \varphi\, \mathbf{U}^{\leq n} \psi))_{s \in S}$$

Let $S_{=1} = \mathrm{Sat}(\psi)$, $S_{=0} = S \setminus (\mathrm{Sat}(\varphi) \cup \mathrm{Sat}(\psi))$ and $S_{?} = S \setminus (\mathrm{Sat}(S_{=0}) \cup \mathrm{Sat}(S_{=1}))$

Model Checking PCTL: $\mathbf{U}^{\leq n}$-Operator

We want to compute

$$(Pr(\mathcal{M}, s \models \varphi \, \mathbf{U}^{\leq n} \psi))_{s \in S}$$

Let $S_{=1} = \mathrm{Sat}(\psi)$, $S_{=0} = S \setminus (\mathrm{Sat}(\varphi) \cup \mathrm{Sat}(\psi))$ and $S_{?} = S \setminus (\mathrm{Sat}(S_{=0}) \cup \mathrm{Sat}(S_{=1}))$

$$Pr(\mathcal{M}, s \models \varphi \, \mathbf{U}^{\leq n} \psi) = \begin{cases} 0 & \text{if } s \in S_{=0} \\ 1 & \text{if } s \in S_{=1} \\ 0 & \text{if } s \in S_{?} \wedge n = 0 \\ \sum_{s' \in S} Pr(s, s') \cdot Pr(\mathcal{M}, s' \models \varphi \, \mathbf{U}^{\leq n-1} \psi) & \text{else} \end{cases}$$

Model Checking PCTL: $\mathbf{U}^{\leq n}$-Operator

We want to compute

$$(Pr(\mathcal{M}, s \models \varphi\, \mathbf{U}^{\leq n} \psi))_{s \in S}$$

Let $S_{=1} = \mathrm{Sat}(\psi)$, $S_{=0} = S \setminus (\mathrm{Sat}(\varphi) \cup \mathrm{Sat}(\psi))$ and $S_{?} = S \setminus (\mathrm{Sat}(S_{=0}) \cup \mathrm{Sat}(S_{=1}))$

Algorithm:

- We let $\mathbf{A}_{\varphi,\psi}$ be the matrix of the induced Markov chain $\mathcal{M}[S_{=1} \cup S_{=0}]$
- The probability for $n = 0$: $(Pr(\mathcal{M}, s \models \varphi\, \mathbf{U}^{\leq 0} \psi))_{s \in S} = \mathbf{b}_{\psi}$
- With multiple matrix-vector multiplications:
  $(Pr(\mathcal{M}, s \models \varphi\, \mathbf{U}^{\leq i} \psi))_{s \in S} = \mathbf{A}_{\varphi,\psi} \cdot (Pr(\mathcal{M}, s \models \varphi\, \mathbf{U}^{\leq i-1} \psi))_{s \in S}$

Model Checking PCTL: $\mathbf{U}$ -Operator

Model Checking PCTL: $\mathbf{U}$ -Operator

We can use the same procedure as discussed in the last lecture:

- We compute:
  - $S_{=1} = \text{Sat}(\mathbb{P}_{=1}(\varphi\ \mathbf{U}\ \psi))$ and $S_{=0} = \text{Sat}(\mathbb{P}_{=0}(\varphi\ \mathbf{U}\ \psi))$

- Computing the probabilities for $S_?$ by solving a linear equation system.

Model Checking PCTL - Time Complexity

We denote with $size(\mathcal{M})$ the number of states plut the number of transitions (s, s') for which $\mathbb{P}(s, s') > 0$.

- $\mathbf{X}$-Operator:
  $\mathcal{O}(poly(size(\mathcal{M})))$
- $\mathbf{U}^{\leq n}$-Operator: Let n_{max} be the maximal step-bound appearing in any subformula.
  $\mathcal{O}(n_{max} \cdot poly(size(\mathcal{M})))$
- $\mathbf{U}$ -Operator:
  $\mathcal{O}(poly(size(\mathcal{M})))$

In total we have that the model checking problem $\mathcal{M}, s \models \varphi$ can be solved in

$\mathcal{O}(n_{max} \cdot poly(size(\mathcal{M})) \cdot |\varphi|)$

where $|\varphi|$ is the amount of subformulae to be checked.

PCTL*

As with CTL* we drop the requirement to prefix every linear operator with $\mathbb{P}_J$.

$$\begin{aligned} \Phi ::= &\ true \\ &\mid a \\ &\mid \Phi_1 \wedge \Phi_2 \\ &\mid \neg\Phi \\ &\mid \mathbb{P}_J(\varphi) \end{aligned}$$

$$\begin{aligned} \varphi ::= &\ \Phi \\ &\mid \neg\varphi \\ &\mid \mathbf{X}\varphi \\ &\mid \Phi_1\ \mathbf{U}\ \Phi_2 \end{aligned}$$

Model Checking PCTL*

Same as PCTL just one crucial difference: We now have subformulae from LTL which need a different procedure.

Such formulae $\psi = \mathbb{P}_J(\varphi)$, where φ is an LTL formula need special treatment:

Model Checking PCTL*

Same as PCTL just one crucial difference: We now have subformulae from LTL which need a different procedure.

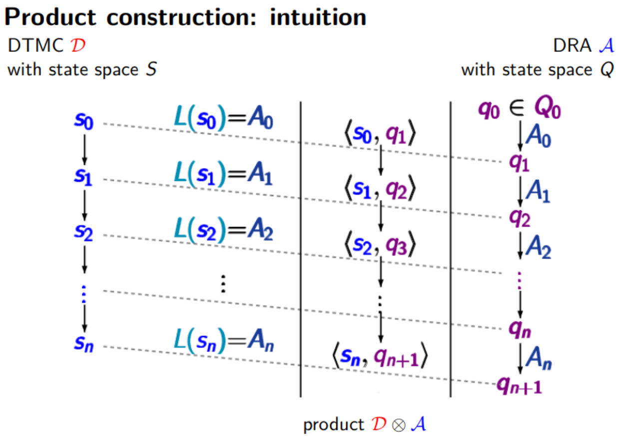
Such formulae $\psi = \mathbb{P}_J(\varphi)$, where φ is an LTL formula need special treatment:

- All maximal state subformulae in φ are replaced by their satisfaction set to get LTL formulae φ'.
- $Sat(\mathbb{P}_J(\varphi)) = \{s \in S \mid Pr(s \models \varphi) \in J\}$

We need to use *automata-based* techniques to compute such satisfaction sets.

Product-MC

Intuition: The automata acts as a *witness* for φ'



Taken from https://moves.rwth-aachen.de/wp-content/uploads/WS1516/mvps/mvps2015_lec05.pdf#page=46

Product-MC

Let $\mathcal{M} = (S, \mathbb{P}, s_0, AP, L)$ be a Markov chain and $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, F)$ be a DFA.

$$\mathcal{M} \otimes \mathcal{A} = (S \times Q, \mathbb{P}', \langle s, q_s \rangle, \{\text{acc}\}, L')$$

- $L'(\langle s, q \rangle) = \{\text{acc}\}$ if $q \in F$, $L'(\langle s, q \rangle) = \emptyset$ else,
- $\mathbb{P}'(\langle s, q \rangle, \langle s', q' \rangle) = \begin{cases} \mathbb{P}(s, s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{else} \end{cases}$

Product-MC

Let $\mathcal{M} = (S, \mathbb{P}, s_0, AP, L)$ be a Markov chain and $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, F)$ be a DFA.

$$\mathcal{M} \otimes \mathcal{A} = (S \times Q, \mathbb{P}', \langle s, q_s \rangle, \{\text{acc}\}, L')$$

- $L'(\langle s, q \rangle) = \{\text{acc}\}$ if $q \in F$, $L'(\langle s, q \rangle) = \emptyset$ else,
- $\mathbb{P}'(\langle s, q \rangle, \langle s', q' \rangle) = \begin{cases} \mathbb{P}(s, s') & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{else} \end{cases}$

Each path fragment $\pi = s_0 s_1 s_2 \ldots$ in $\mathcal{M}$ there exists a unique run $q_0 q_1 q_2 \ldots$ in $\mathcal{A}$.

This works in a similar fashion for different types of automata, after small modifications of the set of accepting states.

Automata Types

A very brief overview:

Let $\mathcal{M}$ be a Markov chain and $\mathcal{A}$ a deterministic automata.

Safety Properties - *Something bad should never happen*

Let P_{safe} be a safety property and $\mathcal{A}$ a **DFA** for the set of bad prefixes of P_{safe}. We are interested in

$$
\begin{aligned}
Pr(\mathcal{M}, s \models P_{safe}) = & Pr(\mathcal{M} \otimes \mathcal{A}, \langle s, q_s \rangle \not\models \mathbf{F}\ \text{acc}) \\
= & 1 - Pr(\mathcal{M} \otimes \mathcal{A}, \langle s, q_s \rangle \models \mathbf{F}\ \text{acc})
\end{aligned}
$$

Automata Types

A very brief overview:

Let $\mathcal{M}$ be a Markov chain and $\mathcal{A}$ a deterministic automata.

DBA-Definable Properties

Let P be a property that can be described by a **deterministic Büchi automata** $\mathcal{A}$. We are interested in

$$Pr(\mathcal{M}, s \models \mathcal{A}) = Pr(\mathcal{M} \otimes \mathcal{A}, \langle s, q_s \rangle \models \mathbf{GF}\ \mathrm{acc})$$

Recall that the longterm behaviour of $\mathcal{M}$ guarantees that we end up in a BSCC T and see all states in T infinitely often.

This means that we only need to solve a reachability problem in $\mathcal{M} \otimes \mathcal{A}$!

Automata Types

A very brief overview:

Let $\mathcal{M}$ be a Markov chain and $\mathcal{A}$ a deterministic automata.

DRA-Based Analysis

Let P be an ω-regular property. P can be described by a **deterministic Rabin automata** $\mathcal{A}$.

The acceptance condition of $\mathcal{A}$ is a set of tuples of atomic propositions $\{(L_1, K_1), \ldots, (L_m, K_m)\}$. For one $i \in [1, m]$ we want to see only *finitely* many atomic propositions from L_i and *infinitely* many from K_i.

Automata Types

A very brief overview:

Let $\mathcal{M}$ be a Markov chain and $\mathcal{A}$ a deterministic automata.

DRA-Based Analysis

Let P be an ω-regular property. P can be described by a **deterministic Rabin automata $\mathcal{A}$**.

The acceptance condition of $\mathcal{A}$ is a set of tuples of atomic propositions $\{(L_1, K_1), \ldots, (L_m, K_m)\}$. For one $i \in [1, m]$ we want to see only *finitely* many atomic propositions from L_i and *infinitely* many from K_i.

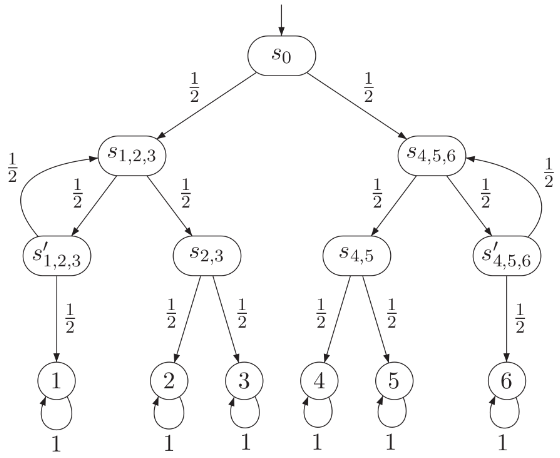
We can work with BSCCs again!

T is an accepting BSCC if: $T \cap (S \times L_i) = \emptyset$ and $T \cap (S \times K_i) \neq \emptyset$

Let $U = \bigcup_{T \textit{ an accepting BSCC}} T$, then $Pr(\mathcal{M}, s \models \mathcal{A}) = Pr(\mathcal{M} \otimes \mathcal{A}, \langle s, q_s \rangle \models \mathbf{F}\, U)$

Example

Knuth-Yao-Die: Simulating a die only using a fair coin.



Example

```
dtmc

module die
    s : [0..7] init 0;
    d : [0..6] init 0;

    [] s=0 -> 0.5 : (s'=1) + 0.5 : (s'=2);
    [] s=1 -> 0.5 : (s'=3) + 0.5 : (s'=4);
    [] s=2 -> 0.5 : (s'=5) + 0.5 : (s'=6);
    [] s=3 -> 0.5 : (s'=1) + 0.5 : (s'=7) & (d'=1);
    [] s=4 -> 0.5 : (s'=7) & (d'=2) + 0.5 : (s'=7) & (d'=3);
    [] s=5 -> 0.5 : (s'=7) & (d'=4) + 0.5 : (s'=7) & (d'=5);
    [] s=6 -> 0.5 : (s'=2) + 0.5 : (s'=7) & (d'=6);
    [] s=7 -> 1: (s'=7);

endmodule

label "one" = s=7&d=1;
label "two" = s=7&d=2;
label "three" = s=7&d=3;
label "done" = s=7;
```

```
P>=1/6 [ F (s=4 & X (s=7&d=3)) ]; P=? [ (F (X (s=6 & (XX s=5)))) & (F G (d!=5))];
```