

Probabilistic Model Checking

Christel Baier

Technical University Dresden

Joost-Pieter Katoen

RWTH Aachen

David Parker

University of Oxford

Probabilistic Computation Tree Logic and Quantitative Linear-Time Properties

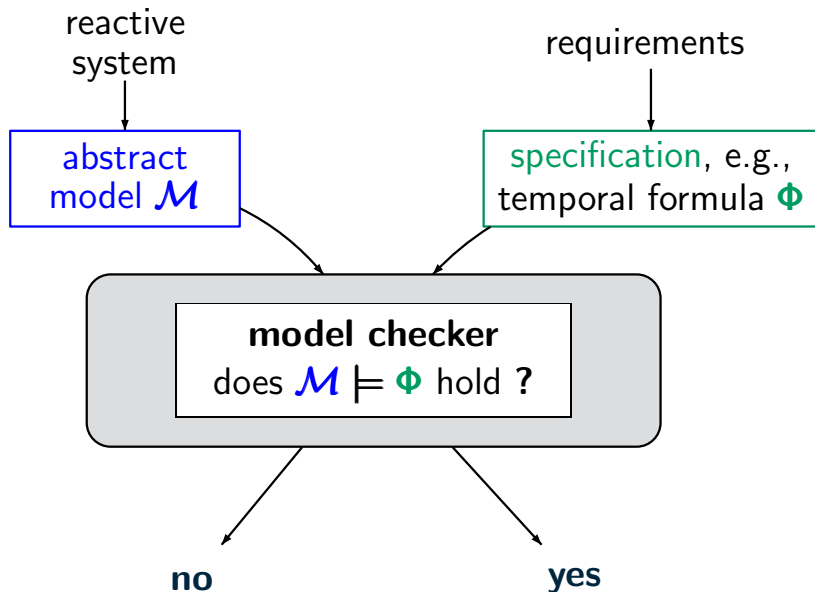
Christel Baier

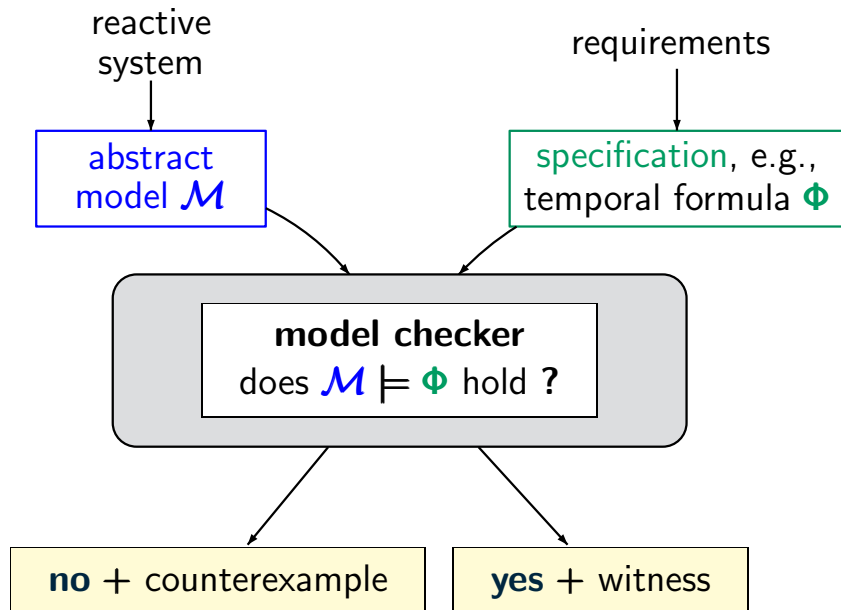
Technical University Dresden

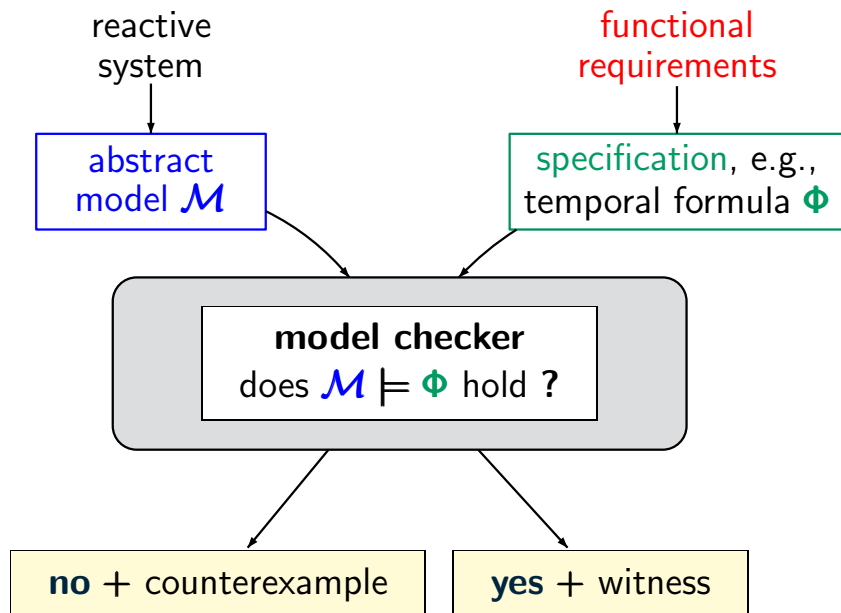
- randomized algorithms [RABIN'1960]
fingerprints, input sampling, breaking symmetry, ...
models: discrete-time Markov chains (DTMC)
Markov decision processes (MDP)

- **randomized algorithms** [RABIN'1960]
fingerprints, input sampling, breaking symmetry, ...
models: discrete-time Markov chains (DTMC)
Markov decision processes (MDP)
- **performance modeling** [ERLANG 1907]
emphasis on steady-state and transient measures
models: continuous-time Markov chains

- randomized algorithms [RABIN'1960]
fingerprints, input sampling, breaking symmetry, ...
models: discrete-time Markov chains (DTMC)
Markov decision processes (MDP)
- performance modeling [ERLANG 1907]
emphasis on steady-state and transient measures
models: continuous-time Markov chains
- stochastic control theory [BELLMAN 1957]
operations research
models: Markov decision processes
- modelling biological systems, security protocols
- ⋮
- ⋮

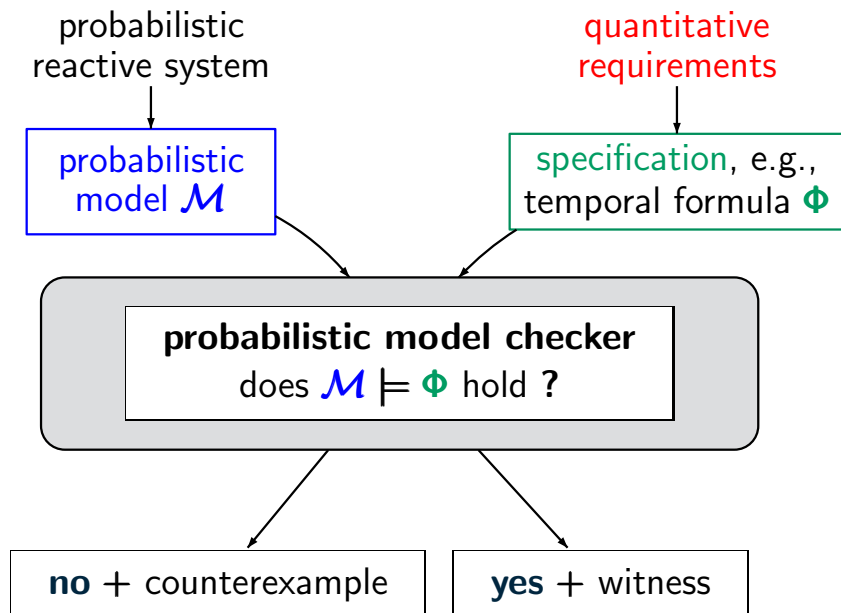


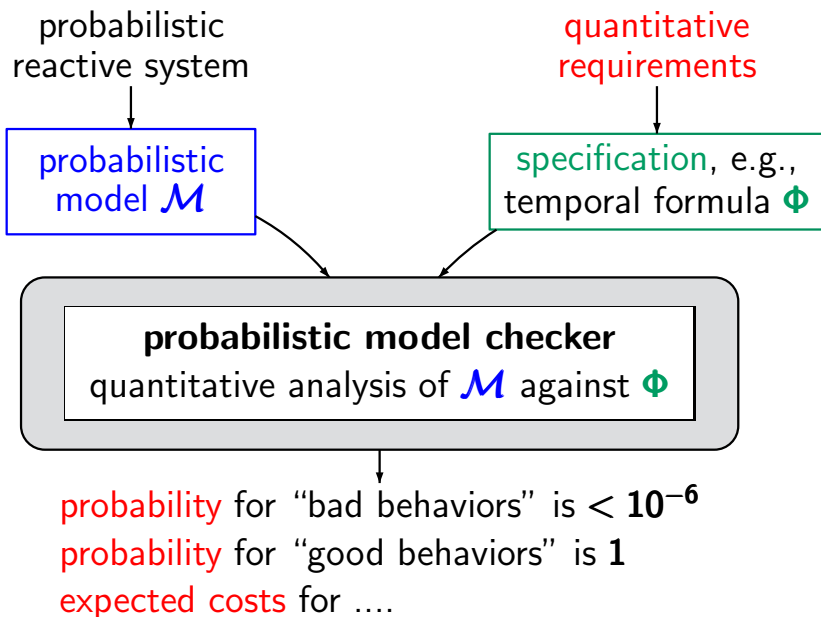


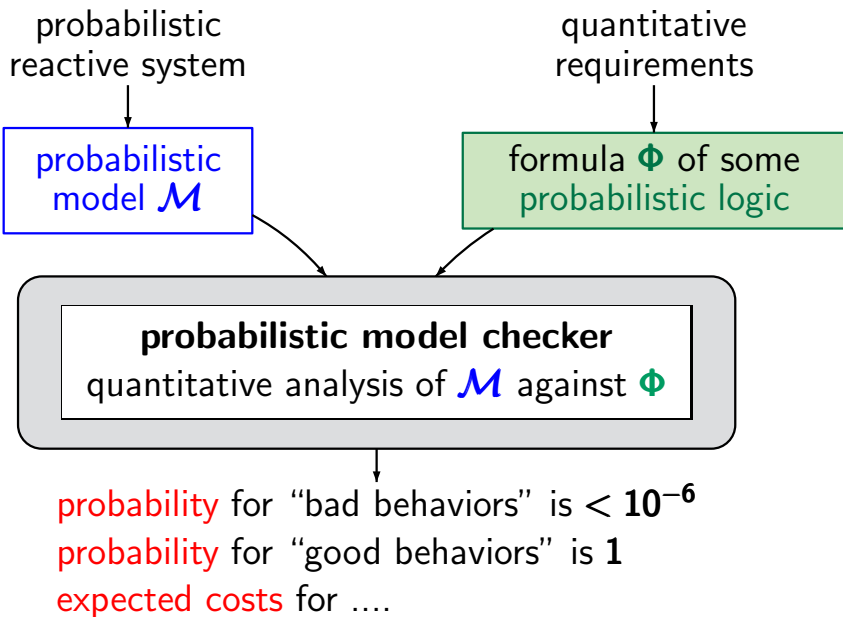


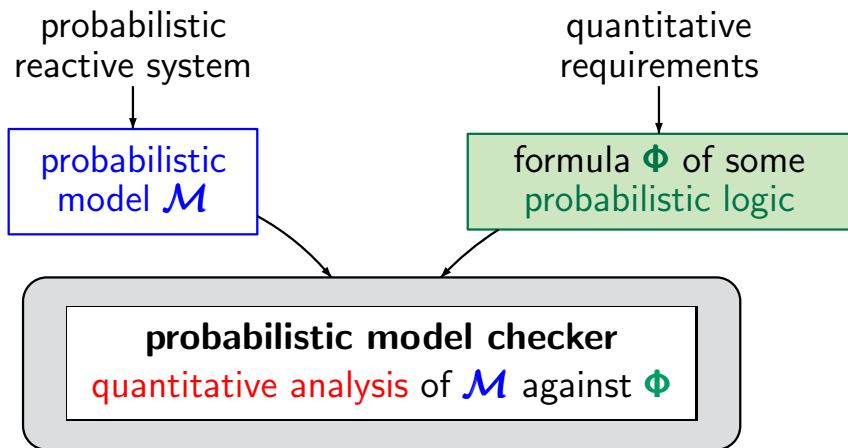
Probabilistic model checking

PMC-05



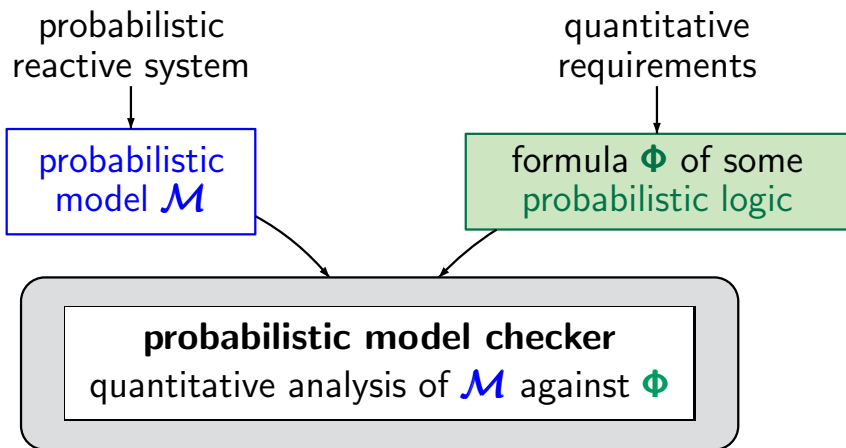




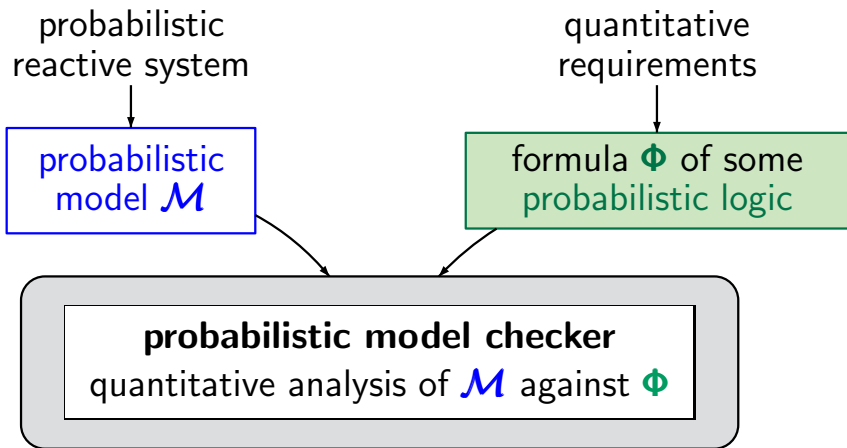


quantitative analysis relies on a combination of

- model checking techniques
- known concepts for stochastic models



logical approach \rightsquigarrow unambiguous measure specifications



logical approach \rightsquigarrow unambiguous measure specifications
model checking \rightsquigarrow automatic computation of
quantitative measures (probabilities, expectation)

part 1: Markov chains



probabilistic computation tree logic
(PCTL/PCTL*)

part 2: Markov decision processes (MDP)

PCTL/PCTL* over MDP

partial order reduction for MDP

MDP with fairness

is a transition system with **probabilities**
for the successor states

$$\mathcal{M} = (\mathcal{S}, P, \dots)$$

- state space \mathcal{S}

$$\mathcal{M} = (\mathcal{S}, P, \dots)$$

- state space \mathcal{S} ← here: finite

$$\mathcal{M} = (\mathcal{S}, P, \dots)$$

- state space \mathcal{S} ← here: finite
- transition probability function $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\sum_{s' \in \mathcal{S}} P(s, s') = 1$$

$$\mathcal{M} = (\mathcal{S}, P, \dots)$$

- state space \mathcal{S} ← here: finite
- transition probability function $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\sum_{s' \in \mathcal{S}} P(s, s') = 1$$



discrete-time or time-abstract

$$\mathcal{M} = (\mathcal{S}, P, AP, L)$$

- state space \mathcal{S} ← here: finite
- transition probability function $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ s.t.
$$\sum_{s' \in \mathcal{S}} P(s, s') = 1$$
- AP set of atomic propositions
- labeling function $L : \mathcal{S} \rightarrow 2^{AP}$

$$\mathcal{M} = (\mathcal{S}, P, AP, L)$$

← + initial distribution
or initial state

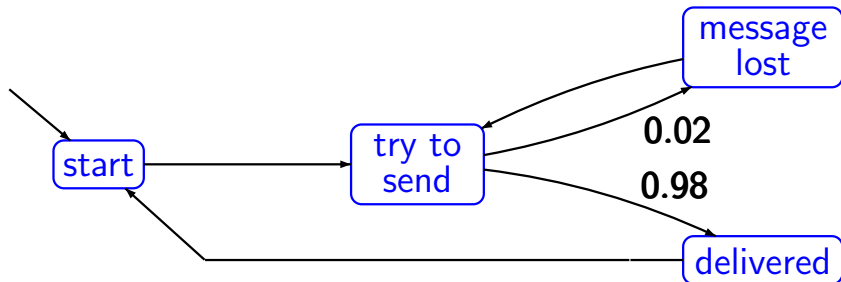
- state space \mathcal{S}
- transition probability function $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ s.t.
$$\sum_{s' \in \mathcal{S}} P(s, s') = 1$$
- AP set of atomic propositions
- labeling function $L : \mathcal{S} \rightarrow 2^{AP}$

$$\mathcal{M} = (\mathcal{S}, P, AP, L)$$

← + initial distribution
or initial state

- state space \mathcal{S}
- transition probability function $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\sum_{s' \in \mathcal{S}} P(s, s') = 1$$



Discrete-time Markov chain

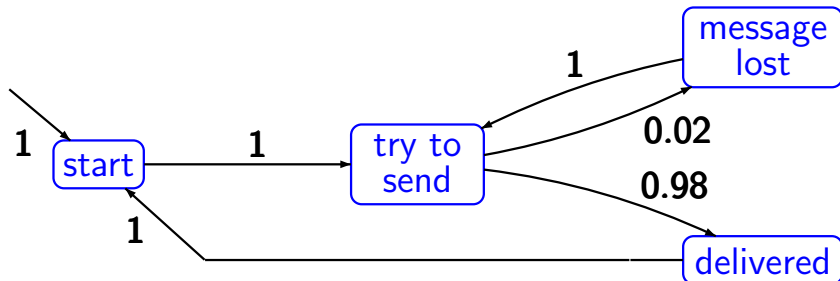
PMC-12

$$\mathcal{M} = (\mathcal{S}, P, AP, L)$$

← + initial distribution
or initial state

- state space \mathcal{S}
- transition probability function $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\sum_{s' \in \mathcal{S}} P(s, s') = 1$$



Discrete-time Markov chain

PMC-12

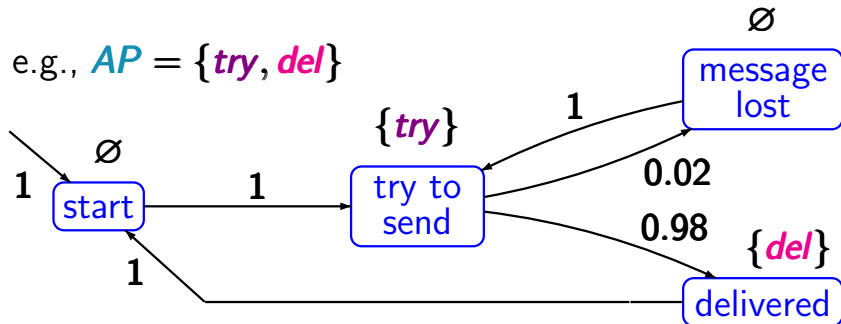
$$\mathcal{M} = (\mathcal{S}, P, AP, L)$$

← + initial distribution
or initial state

- state space \mathcal{S}
- transition probability function $P : \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\sum_{s' \in \mathcal{S}} P(s, s') = 1$$

e.g., $AP = \{try, del\}$



$\mathcal{M} = (\mathcal{S}, P, AP, L, \rho_0)$ Markov chain



initial distribution $\rho_0 : \mathcal{S} \rightarrow [0, 1]$

$\mathcal{M} = (\mathcal{S}, P, AP, L, \rho_0)$ Markov chain



initial distribution $\rho_0 : \mathcal{S} \rightarrow [0, 1]$

probability measure for sets of paths:

$\mathcal{M} = (\mathcal{S}, P, AP, L, \rho_0)$ Markov chain

initial distribution $\rho_0 : \mathcal{S} \rightarrow [0, 1]$

probability measure for sets of paths:

consider the σ -algebra generated by **cylinder sets**

$\Delta(s_0 s_1 \dots s_n) =$ set of infinite paths
 $s_0 s_1 \dots s_n s_{n+1} s_{n+2} s_{n+3} \dots$

finite path

$\mathcal{M} = (\mathcal{S}, P, AP, L, \rho_0)$ Markov chain

initial distribution $\rho_0 : \mathcal{S} \rightarrow [0, 1]$

probability measure for sets of paths:

consider the σ -algebra generated by **cylinder sets**

$\Delta(s_0 s_1 \dots s_n) =$ set of infinite paths
 $s_0 s_1 \dots s_n s_{n+1} s_{n+2} s_{n+3} \dots$

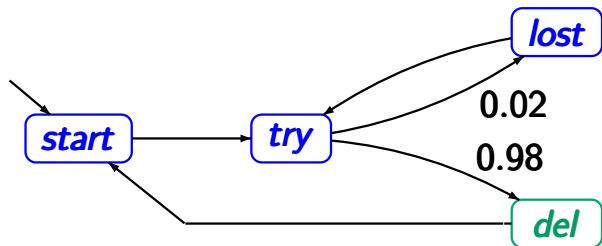
finite path

probability measure is given by:

$$\Pr^{\mathcal{M}}(\Delta(s_0 s_1 \dots s_n)) = \rho_0(s_0) \cdot \prod_{1 \leq i \leq n} P(s_{i-1}, s_i)$$

Example

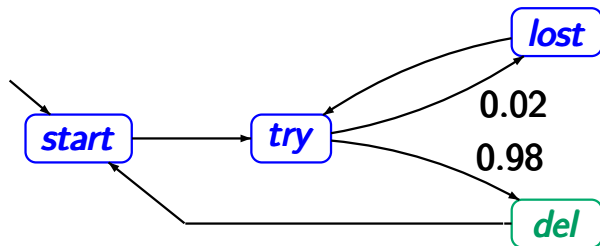
PMC-16



probability for delivering the message within **5** steps:

Example

PMC-16



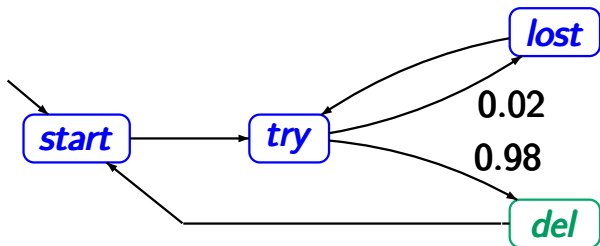
probability for delivering the message within **5** steps:

$$= \Pr^M(\text{start try del}) + \Pr^M(\text{start try lost try del})$$

$$= 0.98 + 0.02 \cdot 0.98 = 0.9996$$

Example

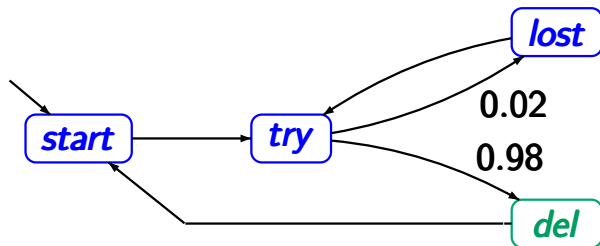
PMC-16



probability for **eventually** delivering the message:

Example

PMC-16



probability for **eventually** delivering the message:

$$= \sum_{n=0}^{\infty} \Pr^{\mathcal{M}}(\text{start try } (\text{lost try})^n \text{ del})$$

$$= \sum_{n=0}^{\infty} 0.02^n \cdot 0.98 = 1$$

Almost surely, i.e., with probability **1**:

A bottom strongly connected component will be reached and all its states visited infinitely often.

Almost surely, i.e., with probability **1**:

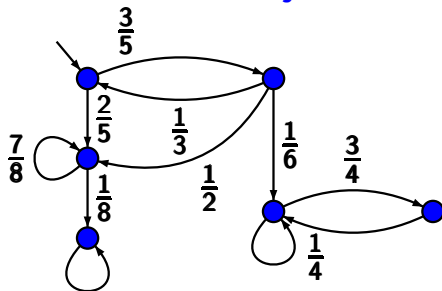
A bottom strongly connected component will be reached and all its states visited infinitely often.

$$\Pr^M \left\{ s_0 s_1 s_2 \dots : \exists i \geq 0 \exists \text{ BSCC } C \text{ s.t.} \right. \\ \left. \forall j \geq i. s_j \in C \wedge \forall s \in C \exists^\infty j. s_j = s \right\} = 1$$

Almost surely, i.e., with probability **1**:

A bottom strongly connected component will be reached and all its states visited infinitely often.

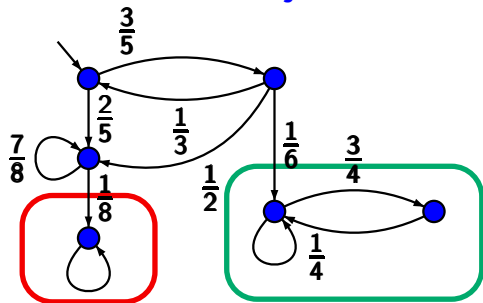
$$\Pr^M \left\{ s_0 s_1 s_2 \dots : \exists i \geq 0 \exists \text{ BSCC } C \text{ s.t.} \right. \\ \left. \forall j \geq i. s_j \in C \wedge \forall s \in C \exists j. s_j = s \right\} = 1$$



Almost surely, i.e., with probability **1**:

A bottom strongly connected component will be reached and all its states visited infinitely often.

$$\Pr^M \left\{ s_0 s_1 s_2 \dots : \exists i \geq 0 \exists \text{ BSCC } C \text{ s.t.} \right. \\ \left. \forall j \geq i. s_j \in C \wedge \forall s \in C \exists j. s_j = s \right\} = 1$$



2 BSCCs

part 1: Markov chains

probabilistic computation tree logic ←
(PCTL/PCTL*)

part 2: Markov decision processes (MDP)

PCTL/PCTL* over MDP

partial order reduction

fairness

PCTL/PCTL* [Hansson/Jonsson 1994]

- probabilistic variants of **CTL/CTL***
- contains a **probabilistic operator** \mathbb{P}
to specify lower/upper probability bounds

PCTL/PCTL* [Hansson/Jonsson 1994]

- probabilistic variants of **CTL/CTL***
- contains a **probabilistic operator** \mathbb{P}
to specify lower/upper probability bounds
- operators for expected costs, long-run averages, ...
not considered here, but can be added

state formulas:

$$\Phi ::= \textit{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \dots$$

path formulas:

$$\varphi ::= \dots$$

state formulas:

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

path formulas:

$$\varphi ::= \dots$$

where $a \in AP$ is an atomic proposition

$I \subseteq [0, 1]$ is a probability interval

state formulas:

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

path formulas:

$$\varphi ::= \dots$$

where $a \in AP$ is an atomic proposition

$I \subseteq [0, 1]$ is a probability interval

qualitative properties: $\mathbb{P}_{>0}(\varphi)$ or $\mathbb{P}_{=1}(\varphi)$

quantitative properties: e.g., $\mathbb{P}_{>0.5}(\varphi)$ or $\mathbb{P}_{\leq 0.01}(\varphi)$

state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \dots$$

↑
state formula

state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \dots$$

$$\bigcirc \hat{=} \text{next}$$

state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

$\bigcirc \hat{=}$ next $\mathbf{U} \hat{=}$ until

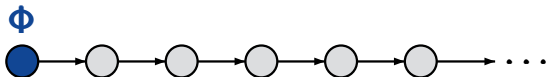
state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

state formula

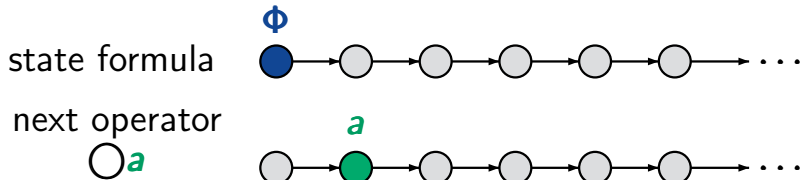


state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

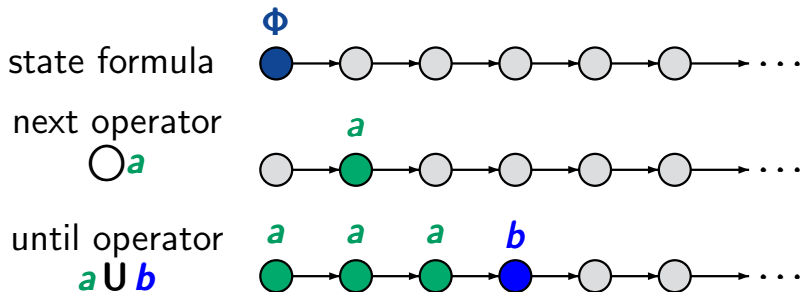


state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathbf{U} \varphi_2$$



Derived path operators: eventually, always

PMC-28

state formulas:

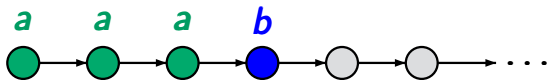
$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \mathbb{P}_I(\varphi)$$

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

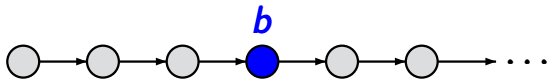
until operator

$$a \mathbf{U} b$$



eventually

$$\diamond b \stackrel{\text{def}}{=} \text{true} \mathbf{U} b$$



state formulas:

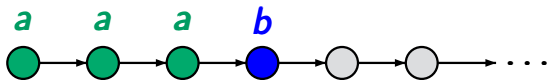
$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg \Phi \mid \mathbb{P}_I(\varphi)$$

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

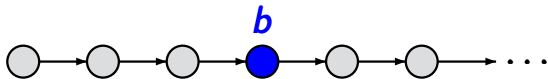
until operator

$$a \mathbf{U} b$$



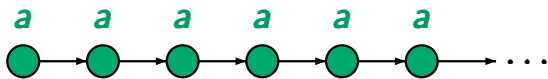
eventually

$$\diamond b \stackrel{\text{def}}{=} \text{true} \mathbf{U} b$$



always

$$\square a \stackrel{\text{def}}{=} \neg \diamond \neg a$$



Let $\mathcal{M} = (\mathcal{S}, P, AP, L)$ be a Markov chain.

define by structural induction:

- a satisfaction relation \models for states $s \in \mathcal{S}$ and **PCTL*** state formulas
- a satisfaction relation \models for infinite path fragments σ in \mathcal{M} and **PCTL*** path formulas

$s \models \text{true}$ $s \models a \quad \text{iff} \quad a \in L(s)$ $s \models \neg\Phi \quad \text{iff} \quad s \not\models \Phi$ $s \models \Phi_1 \wedge \Phi_2 \quad \text{iff} \quad s \models \Phi_1 \text{ and } s \models \Phi_2$ $s \models \mathbb{P}_I(\varphi) \quad \text{iff} \quad \text{Pr}^M(s, \varphi) \in I$

$$s \models \text{true}$$

$$s \models a \quad \text{iff} \quad a \in L(s)$$

$$s \models \neg\Phi \quad \text{iff} \quad s \not\models \Phi$$

$$s \models \Phi_1 \wedge \Phi_2 \quad \text{iff} \quad s \models \Phi_1 \text{ and } s \models \Phi_2$$

$$s \models \mathbb{P}_I(\varphi) \quad \text{iff} \quad \boxed{\text{Pr}^M(s, \varphi)} \in I$$

probability measure of the set of paths π
with $\pi \models \varphi$

when s is viewed as the unique starting state

let $\pi = s_0 s_1 s_2 s_3 \dots$ be an infinite path in \mathcal{M}

let $\pi = s_0 s_1 s_2 s_3 \dots$ be an infinite path in \mathcal{M}

$$\pi \models \Phi \quad \text{iff} \quad s_0 \models \Phi$$

$$\pi \models \neg\varphi \quad \text{iff} \quad \pi \not\models \varphi$$

$$\pi \models \varphi_1 \wedge \varphi_2 \quad \text{iff} \quad \pi \models \varphi_1 \text{ and } \pi \models \varphi_2$$

$$\pi \models \bigcirc\varphi \quad \text{iff} \quad s_1 s_2 s_3 \dots \models \varphi$$

$$\pi \models \varphi_1 \mathbf{U} \varphi_2 \quad \text{iff} \quad \text{there exists } \ell \geq 0 \text{ such that}$$

$$s_\ell s_{\ell+1} s_{\ell+2} \dots \models \varphi_2$$

$$s_i s_{i+1} s_{i+2} \dots \models \varphi_1 \quad \text{for } 0 \leq i < \ell$$

communication protocol:

$$\mathbb{P}_{\leq 0.001}(\diamond \textit{error})$$

$$\mathbb{P}_{=1}(\square(\textit{try_to_send} \longrightarrow \mathbb{P}_{\geq 0.9}(\bigcirc \textit{delivered})))$$

$$\mathbb{P}_{=1}(\square(\textit{try_to_send} \longrightarrow \neg \textit{start} \cup \textit{delivered}))$$

communication protocol:

$$\mathbb{P}_{\leq 0.001}(\diamond \textit{error})$$

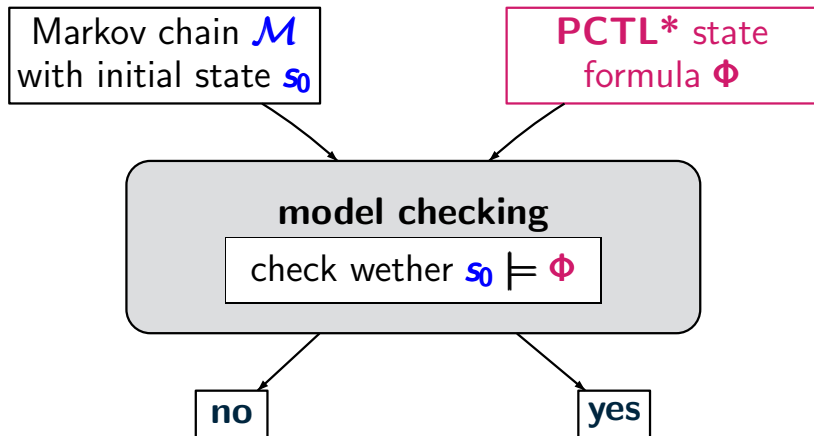
$$\mathbb{P}_{=1}(\square(\textit{try_to_send} \longrightarrow \mathbb{P}_{\geq 0.9}(\bigcirc \textit{delivered})))$$

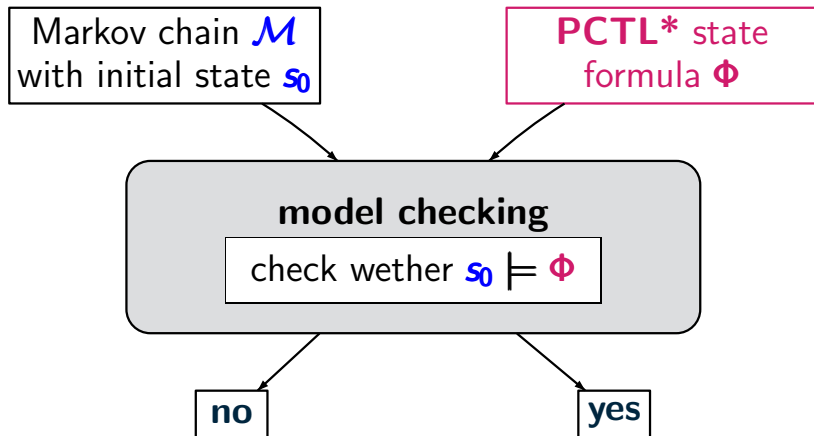
$$\mathbb{P}_{=1}(\square(\textit{try_to_send} \longrightarrow \neg \textit{start} \cup \textit{delivered}))$$

leader election protocol for n processes in a ring

- each process chooses a random number in $\{1, \dots, k\}$ as id
- all ids are synchronously passed around the ring
- if there is a unique id then elect the process with the max. unique id, otherwise repeat

$$\mathbb{P}_{=1}(\diamond \textit{leader_elected}), \mathbb{P}_{\geq 0.9}(\bigvee_{i \leq n} \bigcirc^i \textit{leader_elected})$$





idea: recursively compute $Sat(\Psi) = \{s : s \models \Psi\}$
for all sub-state formulas Ψ of Φ and
check whether $s_0 \in Sat(\Phi)$

Recursive computation of the satisfaction sets

PMC-41

$$\text{Sat}(\text{true}) = S \quad \text{state space of } \mathcal{M}$$

$$\text{Sat}(a) = \{s \in S : a \in L(s)\}$$

$$\text{Sat}(\Phi_1 \wedge \Phi_2) = \text{Sat}(\Phi_1) \cap \text{Sat}(\Phi_2)$$

$$\text{Sat}(\neg\Phi) = S \setminus \text{Sat}(\Phi)$$

$$\text{Sat}(\text{true}) = S \quad \text{state space of } \mathcal{M}$$

$$\text{Sat}(a) = \{s \in S : a \in L(s)\}$$

$$\text{Sat}(\Phi_1 \wedge \Phi_2) = \text{Sat}(\Phi_1) \cap \text{Sat}(\Phi_2)$$

$$\text{Sat}(\neg\Phi) = S \setminus \text{Sat}(\Phi)$$

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : \text{Pr}^{\mathcal{M}}(s, \varphi) \in I\}$$

$$\text{Sat}(\text{true}) = S \quad \text{state space of } \mathcal{M}$$

$$\text{Sat}(a) = \{s \in S : a \in L(s)\}$$

$$\text{Sat}(\Phi_1 \wedge \Phi_2) = \text{Sat}(\Phi_1) \cap \text{Sat}(\Phi_2)$$

$$\text{Sat}(\neg\Phi) = S \setminus \text{Sat}(\Phi)$$

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : \Pr^{\mathcal{M}}(s, \varphi) \in I\}$$



special case: $\varphi = \diamond\Phi$

$$\text{Sat}(\text{true}) = S \quad \text{state space of } \mathcal{M}$$

$$\text{Sat}(a) = \{s \in S : a \in L(s)\}$$

$$\text{Sat}(\Phi_1 \wedge \Phi_2) = \text{Sat}(\Phi_1) \cap \text{Sat}(\Phi_2)$$

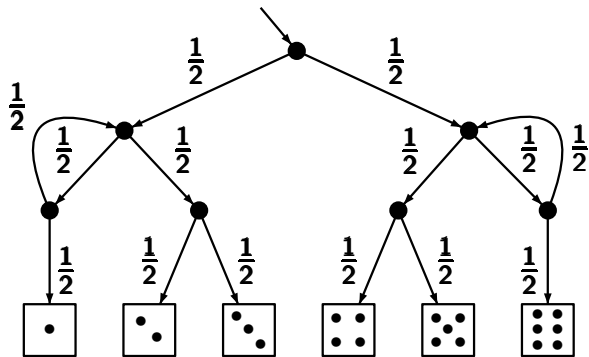
$$\text{Sat}(\neg\Phi) = S \setminus \text{Sat}(\Phi)$$

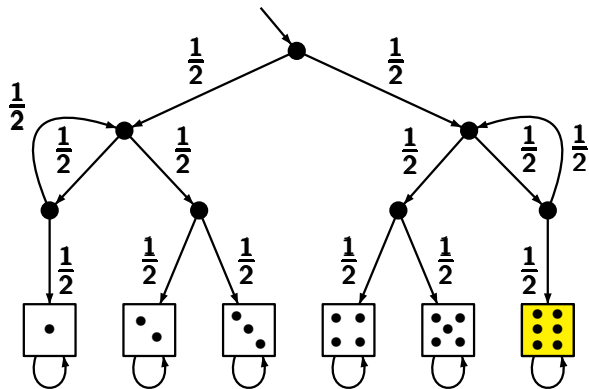
$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : \Pr^{\mathcal{M}}(s, \varphi) \in I\}$$



special case: $\varphi = \diamond\Phi$

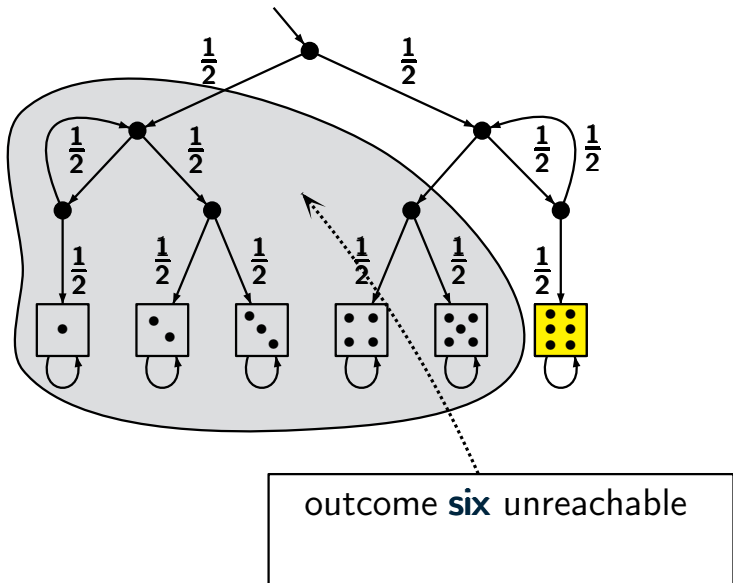
1. compute recursively $\text{Sat}(\Phi)$
2. compute $x_s = \Pr^{\mathcal{M}}(s, \diamond\Phi)$ by solving a linear equation system





probability for the outcome **six**

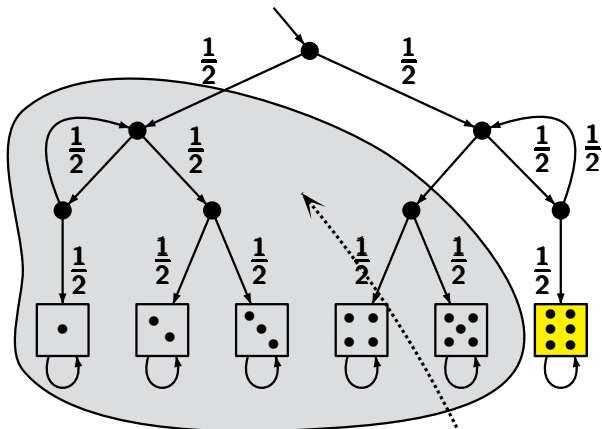
$$\Pr^M(\diamond \text{ six}) = ?$$



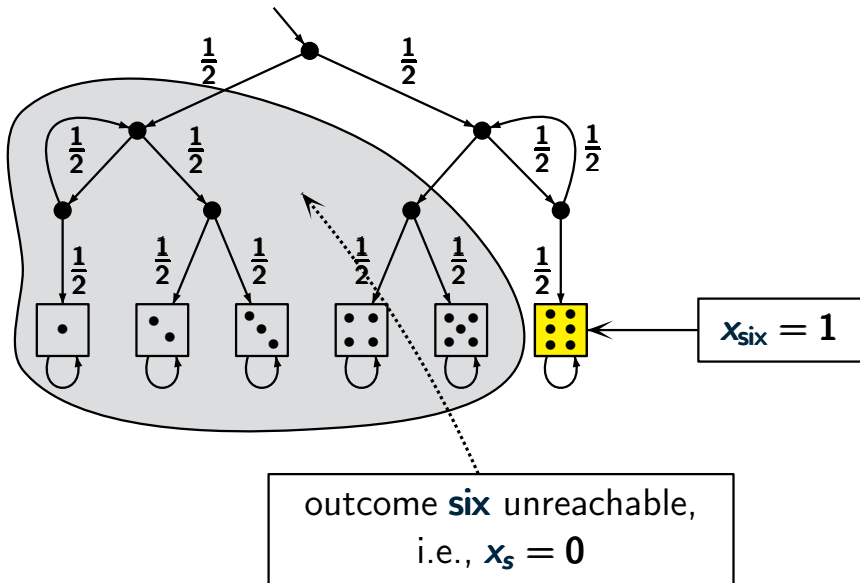
Simulating a dice by a coin

[Knuth]

PMC-43



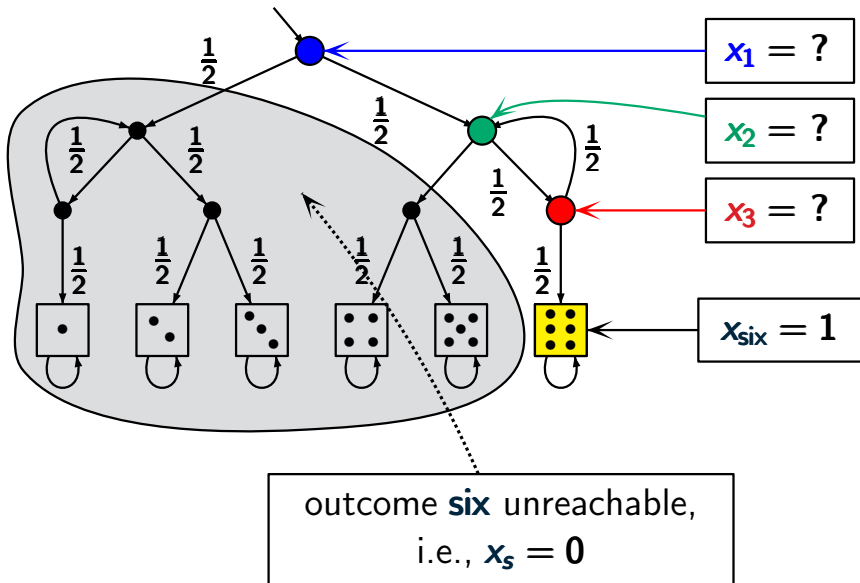
outcome **six** unreachable,
i.e., $x_s = 0$



Simulating a dice by a coin

[Knuth]

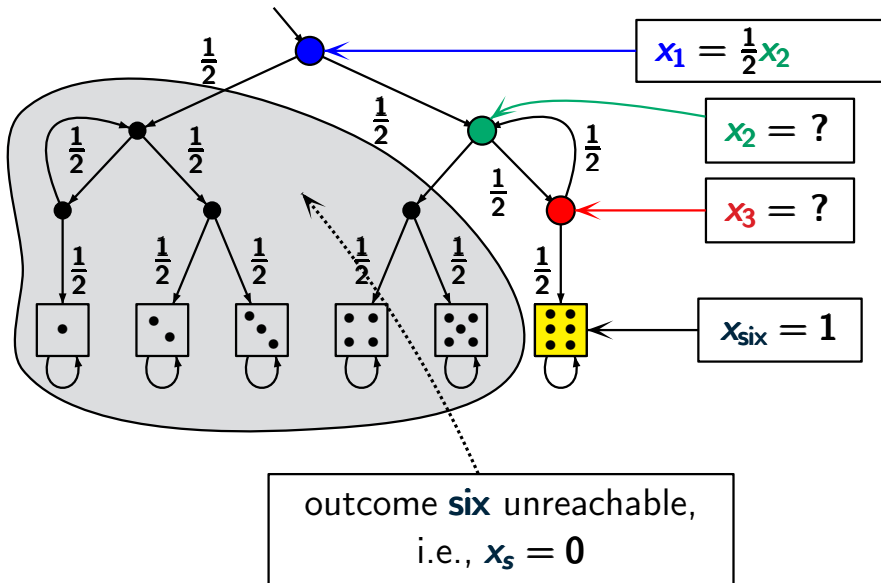
PMC-43



Simulating a dice by a coin

[Knuth]

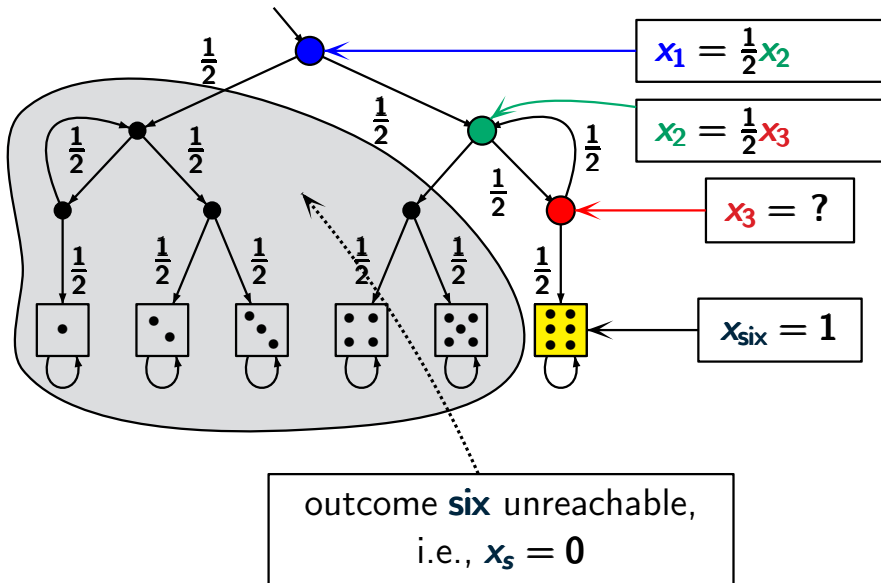
PMC-43



Simulating a dice by a coin

[Knuth]

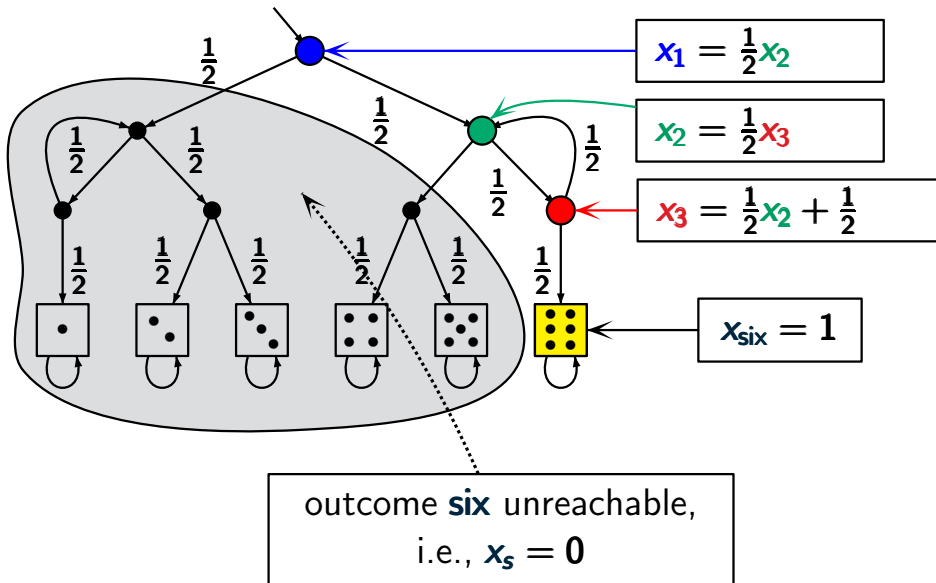
PMC-43



Simulating a dice by a coin

[Knuth]

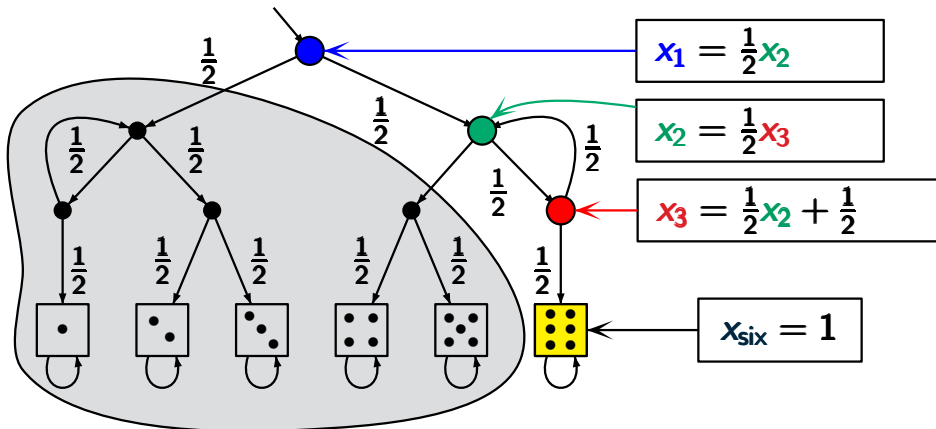
PMC-43



Simulating a dice by a coin

[Knuth]

PMC-43

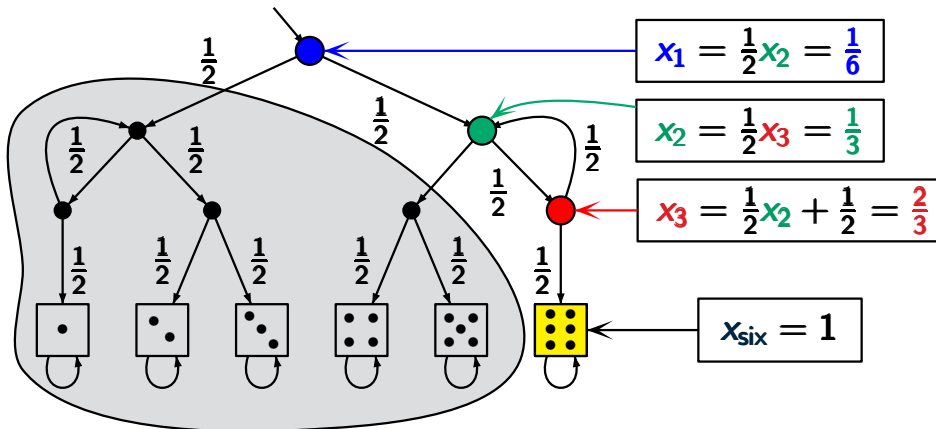


$$\begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

Simulating a dice by a coin

[Knuth]

PMC-43



$$\Pr^M(\diamond \text{ six}) = x_1 = \frac{1}{6}$$

$$\begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{2} \end{pmatrix}$$

sublogic of **PCTL*** where only path formulas of the form $\bigcirc\Phi$ and $\Phi_1 \text{ U } \Phi_2$ are allowed

sublogic of **PCTL*** where only path formulas of the form $\bigcirc\Phi$ and $\Phi_1 \mathbf{U} \Phi_2$ are allowed

state formulas:

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

path formulas:

$$\varphi ::= \bigcirc\Phi \mid \Phi_1 \mathbf{U} \Phi_2$$

sublogic of **PCTL*** where only path formulas of the form $\bigcirc\Phi$ and $\Phi_1 \mathbf{U} \Phi_2$ are allowed

state formulas:

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

path formulas:

$$\varphi ::= \bigcirc\Phi \mid \Phi_1 \mathbf{U} \Phi_2 \mid \diamond\Phi \mid \square\Phi$$

$$\mathbb{P}_I(\diamond\Phi) \stackrel{\text{def}}{=} \mathbb{P}_I(\mathit{true} \mathbf{U} \Phi)$$

sublogic of **PCTL*** where only path formulas of the form $\bigcirc\Phi$ and $\Phi_1 \mathbf{U} \Phi_2$ are allowed

state formulas:

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

path formulas:

$$\varphi ::= \bigcirc\Phi \mid \Phi_1 \mathbf{U} \Phi_2 \mid \diamond\Phi \mid \square\Phi$$

$$\mathbb{P}_I(\diamond\Phi) \stackrel{\text{def}}{=} \mathbb{P}_I(\mathit{true} \mathbf{U} \Phi)$$

$$\text{e.g., } \mathbb{P}_{<0.4}(\square\Phi) \stackrel{\text{def}}{=} \mathbb{P}_{>0.6}(\diamond\neg\Phi)$$

sublogic of **PCTL*** where only path formulas of the form $\bigcirc\Phi$ and $\Phi_1 \mathbf{U} \Phi_2$ are allowed

state formulas:

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

path formulas:

$$\varphi ::= \bigcirc\Phi \mid \Phi_1 \mathbf{U} \Phi_2 \mid \diamond\Phi \mid \square\Phi$$

$$\mathbb{P}_I(\diamond\Phi) \stackrel{\text{def}}{=} \mathbb{P}_I(\mathit{true} \mathbf{U} \Phi)$$

$$\text{e.g., } \mathbb{P}_{<0.4}(\square\Phi) \stackrel{\text{def}}{=} \mathbb{P}_{>0.6}(\diamond\neg\Phi)$$

$$\text{note: } \Pr^M(s, \square\Phi) \stackrel{\uparrow}{=} 1 - \Pr^M(s, \diamond\neg\Phi)$$

given: Markov chain $\mathcal{M} = (S, P, AP, L, s_0)$

PCTL state formula Φ

task: check whether $\mathcal{M} \models \Phi$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL state formula ϕ

task: check whether $\mathcal{M} \models \phi$

model checking relies on recursive computation of

$$\text{Sat}(\psi) = \{s \in \mathcal{S} : s \models \psi\} \text{ for all subformulas } \psi \text{ of } \phi$$

given: Markov chain $\mathcal{M} = (S, P, AP, L, s_0)$

PCTL state formula ϕ

task: check whether $\mathcal{M} \models \phi$

model checking relies on recursive computation of

$Sat(\psi) = \{s \in S : s \models \psi\}$ for all subformulas ψ of ϕ



in bottom-up manner, i.e., inner subformulas first

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL state formula ϕ

task: check whether $\mathcal{M} \models \phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all subformulas Ψ of ϕ

- treatment of propositional logic fragment:

given: Markov chain $\mathcal{M} = (S, P, AP, L, s_0)$

PCTL state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in S : s \models \Psi\}$ for all subformulas Ψ of Φ

- treatment of propositional logic fragment: obvious

$$Sat(true) = S$$

$$Sat(a) = \{s \in S : a \in L(s)\}$$

$$Sat(\neg\Psi) = S \setminus Sat(\Psi)$$

$$Sat(\Psi_1 \wedge \Psi_2) = Sat(\Psi_1) \cap Sat(\Psi_2)$$

given: Markov chain $\mathcal{M} = (S, P, AP, L, s_0)$

PCTL state formula ϕ

task: check whether $\mathcal{M} \models \phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in S : s \models \Psi\}$ for all subformulas Ψ of ϕ

- treatment of propositional logic fragment: obvious
- treatment of the probability operator $\mathbb{P}_I(\varphi)$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL state formula ϕ

task: check whether $\mathcal{M} \models \phi$

model checking relies on recursive computation of

$Sat(\psi) = \{s \in \mathcal{S} : s \models \psi\}$ for all subformulas ψ of ϕ

- treatment of propositional logic fragment: obvious
- treatment of the probability operator $\mathbb{P}_I(\varphi)$

compute $\Pr^{\mathcal{M}}(s, \varphi)$ for all states s and return

$$Sat(\mathbb{P}_I(\varphi)) = \{s \in \mathcal{S} : \Pr^{\mathcal{M}}(s, \varphi) \in I\}$$

compute $x_s = \Pr^M(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

compute $x_s = \Pr^M(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

next operator, i.e., $\varphi = \bigcirc \psi$

$$\text{compute } x_s = P(s, \text{Sat}(\psi)) = \sum_{s' \in \text{Sat}(\psi)} P(s, s')$$

compute $x_s = \Pr^M(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

next operator, i.e., $\varphi = \bigcirc \Psi$

$$\text{compute } x_s = P(s, \text{Sat}(\Psi)) = \sum_{s' \in \text{Sat}(\Psi)} P(s, s')$$

until operator, i.e., $\varphi = \Psi_1 \mathbf{U} \Psi_2$:

compute $x_s = \Pr^M(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

next operator, i.e., $\varphi = \bigcirc \Psi$

$$\text{compute } x_s = P(s, \text{Sat}(\Psi)) = \sum_{s' \in \text{Sat}(\Psi)} P(s, s')$$

until operator, i.e., $\varphi = \Psi_1 \mathbf{U} \Psi_2$: compute

$$S^0 = \{s \in S : x_s = 0\}$$

$$S^1 = \{s \in S : x_s = 1\}$$

compute $x_s = \Pr^M(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

next operator, i.e., $\varphi = \bigcirc \Psi$

$$\text{compute } x_s = P(s, \text{Sat}(\Psi)) = \sum_{s' \in \text{Sat}(\Psi)} P(s, s')$$

until operator, i.e., $\varphi = \Psi_1 \mathbf{U} \Psi_2$: compute

$$S^0 = \{s \in S : x_s = 0\} = \{s : s \not\models \exists \Psi_1 \mathbf{U} \Psi_2\}$$

$$S^1 = \{s \in S : x_s = 1\}$$

compute $x_s = \Pr^M(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

next operator, i.e., $\varphi = \bigcirc \Psi$

$$\text{compute } x_s = P(s, \text{Sat}(\Psi)) = \sum_{s' \in \text{Sat}(\Psi)} P(s, s')$$

until operator, i.e., $\varphi = \Psi_1 \mathbf{U} \Psi_2$: compute

$$S^0 = \{s \in S : x_s = 0\} = \{s : s \not\models \exists \Psi_1 \mathbf{U} \Psi_2\}$$

$$S^1 = \{s \in S : x_s = 1\} = \{s : s \not\models \exists (\neg \Psi_2) \mathbf{U} S^0\}$$

compute $x_s = \Pr^M(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

next operator, i.e., $\varphi = \bigcirc \Psi$

$$\text{compute } x_s = P(s, \text{Sat}(\Psi)) = \sum_{s' \in \text{Sat}(\Psi)} P(s, s')$$

until operator, i.e., $\varphi = \Psi_1 \mathbf{U} \Psi_2$: compute

$$S^0 = \{s \in S : x_s = 0\} = \{s : s \not\models \exists \Psi_1 \mathbf{U} \Psi_2\}$$

$$S^1 = \{s \in S : x_s = 1\} = \{s : s \not\models \exists (\neg \Psi_2) \mathbf{U} S^0\}$$

and solve the linear equation system

$$x_s = \sum_{s' \in S} P(s, s') \cdot x_{s'} + P(s, \text{Sat}(\Psi_2)) \text{ for } s \in S?$$

compute $x_s = \Pr^{\mathcal{M}}(s, \varphi)$ for all states s and return

$$\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in S : x_s \in I\}$$

next operator, i.e., $\varphi = \bigcirc \Psi$

compute $x_s = P(s, \text{Sat}(\Psi))$

time complexity:
 $\mathcal{O}(\text{poly}(\text{size}(\mathcal{M})))$

until operator, i.e., $\varphi = \Psi_1 \mathbf{U} \Psi_2$: compute

$$S^0 = \{s \in S : x_s = 0\} = \{s : s \not\models \exists \Psi_1 \mathbf{U} \Psi_2\}$$

$$S^1 = \{s \in S : x_s = 1\} = \{s : s \not\models \exists (\neg \Psi_2) \mathbf{U} S^0\}$$

and solve the linear equation system

$$x_s = \sum_{s' \in S} P(s, s') \cdot x_{s'} + P(s, \text{Sat}(\Psi_2)) \text{ for } s \in S?$$

PCTL* model checking

PMC-51

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

- propositional logic fragment: ✓
- probability operator $\mathbb{P}_I(\varphi)$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula ϕ

task: check whether $\mathcal{M} \models \phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

- propositional logic fragment: \checkmark
- probability operator $\mathbb{P}_I(\varphi)$

replace the **PCTL*** formula φ with an **LTL** formula φ'

↑
path formula without state formulas $\mathbb{P}_I(\dots)$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula ϕ

task: check whether $\mathcal{M} \models \phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

- propositional logic fragment: \checkmark
- probability operator $\mathbb{P}_I(\varphi)$

replace the **PCTL*** formula φ with an **LTL** formula φ'

compute $\text{Pr}^{\mathcal{M}}(s, \varphi')$ for all states s and return

$$Sat(\mathbb{P}_I(\varphi)) = \{s \in \mathcal{S} : \text{Pr}^{\mathcal{M}}(s, \varphi') \in I\}$$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

treatment of subformulas $\Psi = \mathbb{P}_I(\varphi)$

PCTL* path formula $\varphi \rightsquigarrow$ LTL formula φ'

by replacing each maximal state-subformula of φ
with a fresh atomic proposition

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

treatment of subformulas $\Psi = \mathbb{P}_I(\varphi)$

PCTL* path formula $\varphi \rightsquigarrow$ LTL formula φ'

$$\diamond(a \mathbf{U} \mathbb{P}_{\geq 0.7}(\square \diamond b) \wedge \square \mathbb{P}_{< 0.3}(\bigcirc \square c))$$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

treatment of subformulas $\Psi = \mathbb{P}_I(\varphi)$

PCTL* path formula $\varphi \rightsquigarrow$ LTL formula φ'

$$\diamond (a \mathbf{U} \mathbb{P}_{\geq 0.7}(\square \diamond b)) \wedge \square \mathbb{P}_{< 0.3}(\bigcirc \square c)$$

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

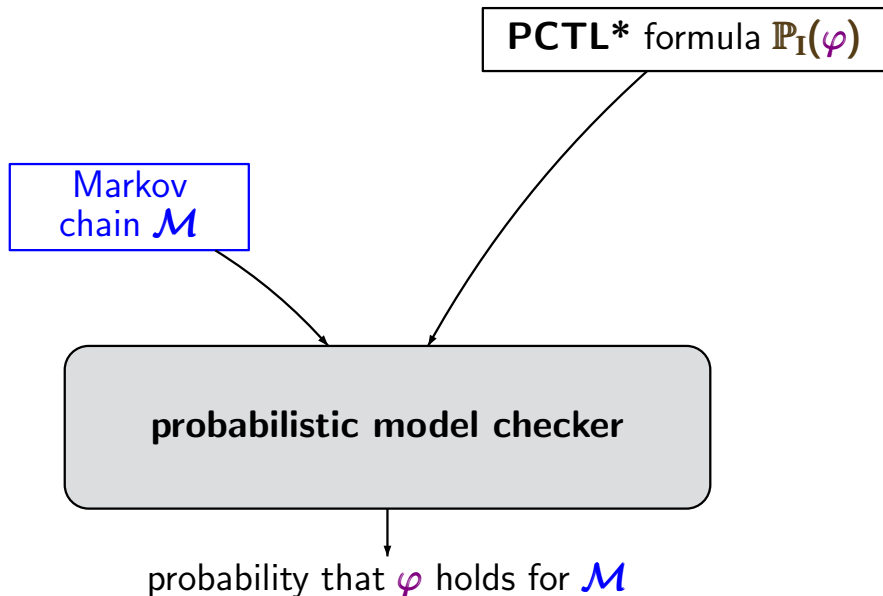
$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$ for all sub-state formulas Ψ

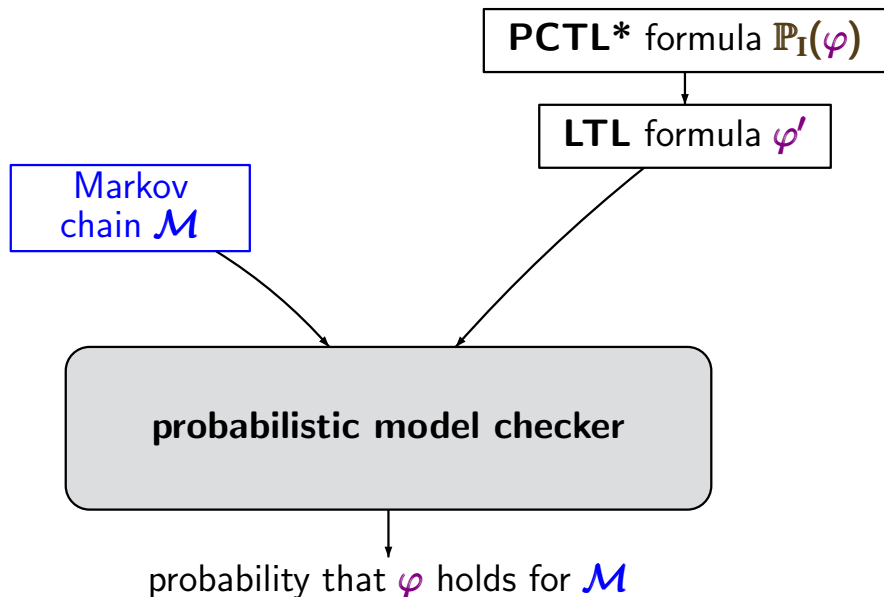
treatment of subformulas $\Psi = \mathbb{P}_I(\varphi)$

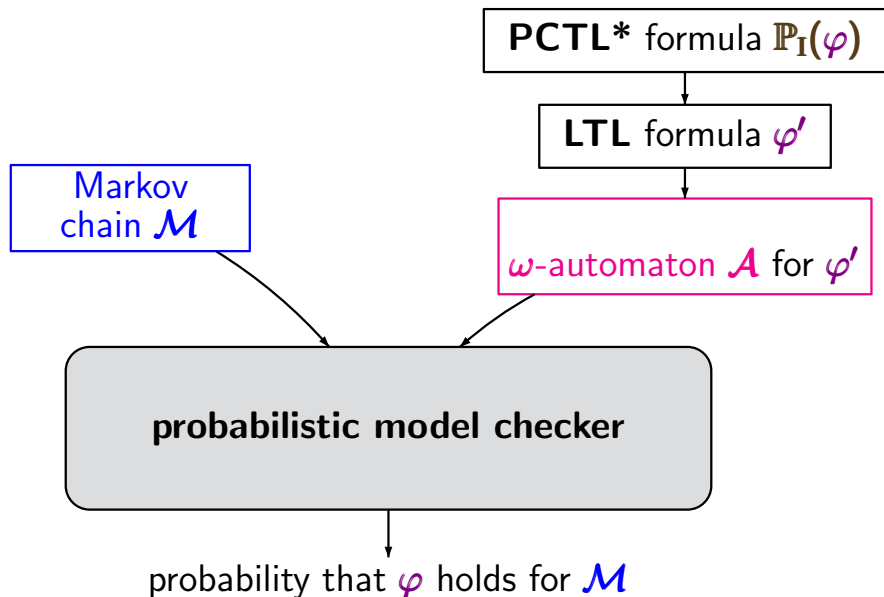
PCTL* path formula $\varphi \rightsquigarrow$ LTL formula φ'

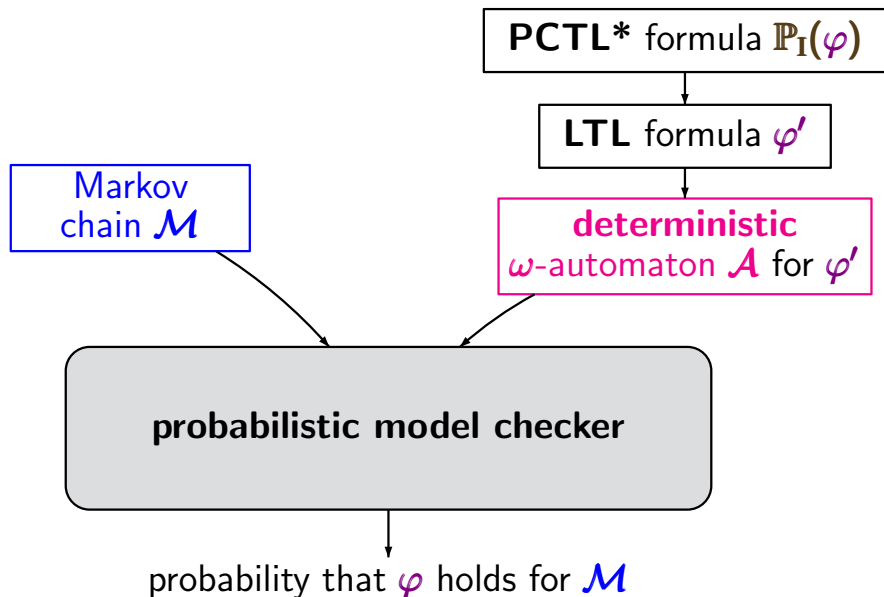
$\diamond(a \mathbb{U} \mathbb{P}_{\geq 0.7}(\square \diamond b) \wedge \square \mathbb{P}_{< 0.3}(\bigcirc \square c))$

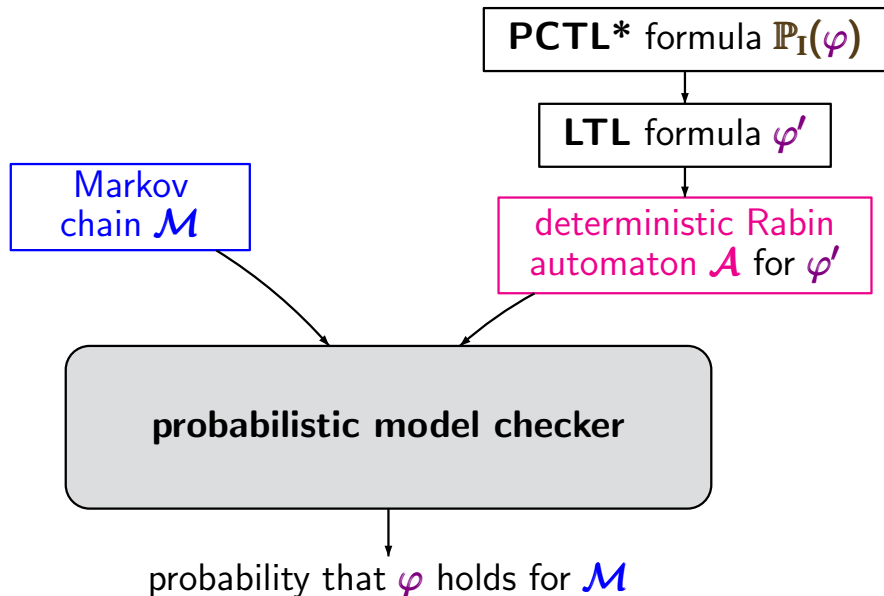
\Downarrow
 $\diamond(a \mathbb{U} d \wedge \square e)$











Deterministic Rabin automata (DRA)

PMC-54

A **DRA** is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, Acc)$ where

- Q finite state space
- $q_0 \in Q$ initial state
- Σ alphabet
- $\delta : Q \times \Sigma \longrightarrow Q$ deterministic transition function

A **DRA** is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \text{Acc})$ where

- Q finite state space
- $q_0 \in Q$ initial state
- Σ alphabet
- $\delta : Q \times \Sigma \longrightarrow Q$ deterministic transition function
- acceptance condition **Acc** is a set of pairs (L, U) with $L, U \subseteq Q$

A **DRA** is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \text{Acc})$ where

- Q finite state space
- $q_0 \in Q$ initial state
- Σ alphabet
- $\delta : Q \times \Sigma \longrightarrow Q$ deterministic transition function
- acceptance condition **Acc** is a set of pairs (L, U) with $L, U \subseteq Q$, say $\text{Acc} = \{(L_1, U_1), \dots, (L_k, U_k)\}$

A **DRA** is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \text{Acc})$ where

- Q finite state space
- $q_0 \in Q$ initial state
- Σ alphabet
- $\delta : Q \times \Sigma \longrightarrow Q$ deterministic transition function
- acceptance condition **Acc** is a set of pairs (L, U) with $L, U \subseteq Q$, say $\text{Acc} = \{(L_1, U_1), \dots, (L_k, U_k)\}$

semantics of the acceptance condition:

$$\bigvee_{1 \leq i \leq k} (\diamond \square \neg L_i \wedge \square \diamond U_i)$$

A **DRA** is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \text{Acc})$ where

- Q finite state space, q_0 initial state, Σ alphabet
- $\delta : Q \times \Sigma \longrightarrow Q$ transition function
- $\text{Acc} = \{(L_1, U_1), \dots, (U_k, U_k)\}$ with $L_i, U_i \subseteq Q$

accepted language:

$$\mathcal{L}_\omega(\mathcal{A}) = \{ \sigma \in \Sigma^\omega : \text{the run for } \sigma \text{ in } \mathcal{A} \text{ fulfills } \text{Acc} \}$$

A **DRA** is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \text{Acc})$ where

- Q finite state space, q_0 initial state, Σ alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ transition function
- $\text{Acc} = \{(L_1, U_1), \dots, (U_k, U_k)\}$ with $L_i, U_i \subseteq Q$

accepted language:

$$\mathcal{L}_\omega(\mathcal{A}) = \{ \sigma \in \Sigma^\omega : \text{the run for } \sigma \text{ in } \mathcal{A} \text{ fulfills } \text{Acc} \}$$

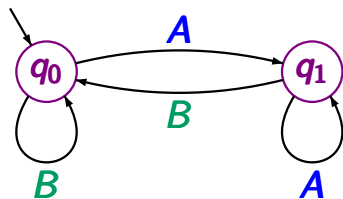
if $\rho = q_0 q_1 q_2 \dots$ is the run for $\sigma = A_0 A_1 A_2 \dots$ then

$$\exists i \in \{1, \dots, k\}. \text{inf}(\rho) \cap L_i = \emptyset \wedge \text{inf}(\rho) \cap U_i \neq \emptyset$$

where $\text{inf}(\rho) = \{ q \in Q : \exists l \in \mathbb{N}. q = q_l \}$

Example: DRA

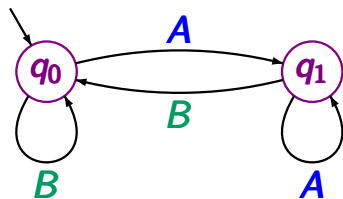
PMC-54B



$$Acc = \{(\{q_0\}, \{q_1\})\}$$

Example: DRA

PMC-54B

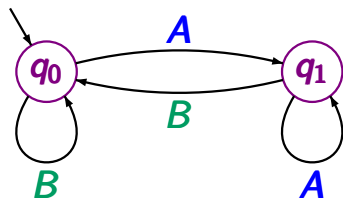


$$Acc = \{(\{q_0\}, \{q_1\})\}$$

$$\hat{=} \diamond \square \neg q_0 \wedge \square \diamond q_1$$

Example: DRA

PMC-54B



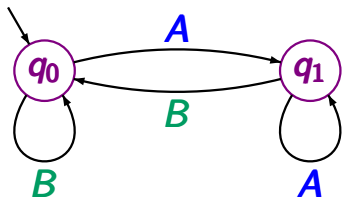
$$Acc = \{(\{q_0\}, \{q_1\})\}$$

$$\hat{=} \diamond \square \neg q_0 \wedge \square \diamond q_1$$

accepted language: $(A + B)^* A^\omega$

Example: DRA

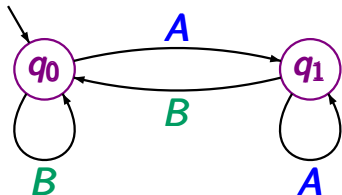
PMC-54B



$$Acc = \{(\{q_0\}, \{q_1\})\}$$

$$\hat{=} \diamond \square \neg q_0 \wedge \square \diamond q_1$$

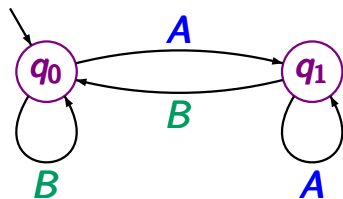
accepted language: $(A + B)^* A^\omega$



$$Acc = \{(\emptyset, \{q_1\})\}$$

Example: DRA

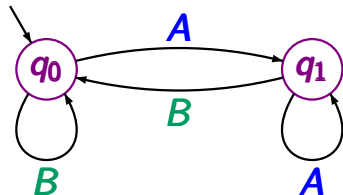
PMC-54B



$$Acc = \{(\{q_0\}, \{q_1\})\}$$

$$\hat{=} \diamond \square \neg q_0 \wedge \square \diamond q_1$$

accepted language: $(A + B)^* A^\omega$

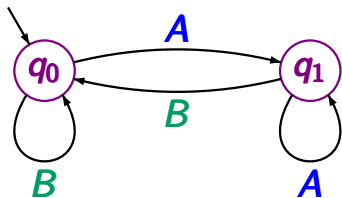


$$Acc = \{(\emptyset, \{q_1\})\}$$

$$\hat{=} \square \diamond q_1$$

Example: DRA

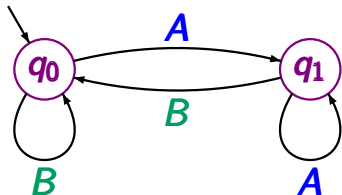
PMC-54B



$$Acc = \{(\{q_0\}, \{q_1\})\}$$

$$\cong \diamond \square \neg q_0 \wedge \square \diamond q_1$$

accepted language: $(A + B)^* A^\omega$



$$Acc = \{(\emptyset, \{q_1\})\}$$

$$\cong \square \diamond q_1$$

accepted language: $(B^* A)^\omega$

Fundamental result: LTL-2-DRA

PMC-55

For each LTL formula φ over AP there exists a DRA \mathcal{A} with the alphabet $\Sigma = 2^{AP}$ s.t.

$$\mathcal{L}_\omega(\mathcal{A}) = \{\sigma \in \Sigma^\omega : \sigma \models \varphi\}$$

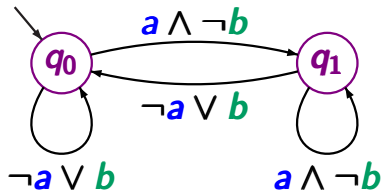
For each LTL formula φ over AP there exists a DRA \mathcal{A} with the alphabet $\Sigma = 2^{AP}$ s.t.

$$\mathcal{L}_\omega(\mathcal{A}) = \{\sigma \in \Sigma^\omega : \sigma \models \varphi\} \text{ and } |\mathcal{A}| = \mathcal{O}(2^{\exp(|\varphi|)})$$

For each LTL formula φ over AP there exists a DRA \mathcal{A} with the alphabet $\Sigma = 2^{AP}$ s.t.

$$\mathcal{L}_\omega(\mathcal{A}) = \{\sigma \in \Sigma^\omega : \sigma \models \varphi\} \text{ and } |\mathcal{A}| = \mathcal{O}(2^{\exp(|\varphi|)})$$

Example: $AP = \{a, b\}$



acceptance condition:

$$\diamond \square \neg q_0 \wedge \square \diamond q_1$$

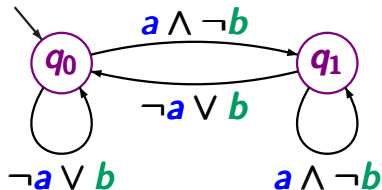
Fundamental result: LTL-2-DRA

PMC-55

For each LTL formula φ over AP there exists a DRA \mathcal{A} with the alphabet $\Sigma = 2^{AP}$ s.t.

$$\mathcal{L}_\omega(\mathcal{A}) = \{\sigma \in \Sigma^\omega : \sigma \models \varphi\} \text{ and } |\mathcal{A}| = \mathcal{O}(2^{\exp(|\varphi|)})$$

Example: $AP = \{a, b\} \rightsquigarrow \Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$



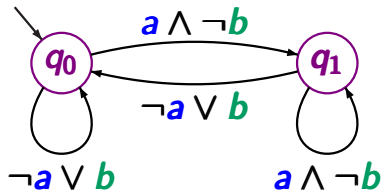
acceptance condition:

$$\diamond \square \neg q_0 \wedge \square \diamond q_1$$

For each LTL formula φ over AP there exists a DRA \mathcal{A} with the alphabet $\Sigma = 2^{AP}$ s.t.

$$\mathcal{L}_\omega(\mathcal{A}) = \{\sigma \in \Sigma^\omega : \sigma \models \varphi\} \text{ and } |\mathcal{A}| = \mathcal{O}(2^{\exp(|\varphi|)})$$

Example: $AP = \{a, b\} \rightsquigarrow \Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$



acceptance condition:

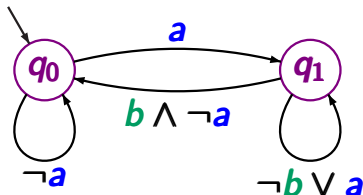
$$\diamond \square \neg q_0 \wedge \square \diamond q_1$$

LTL formula $\diamond \square (a \wedge \neg b)$

For each LTL formula φ over AP there exists a DRA \mathcal{A} with the alphabet $\Sigma = 2^{AP}$ s.t.

$$\mathcal{L}_\omega(\mathcal{A}) = \{\sigma \in \Sigma^\omega : \sigma \models \varphi\} \text{ and } |\mathcal{A}| = \mathcal{O}(2^{\exp(|\varphi|)})$$

Example: $AP = \{a, b\} \rightsquigarrow \Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$



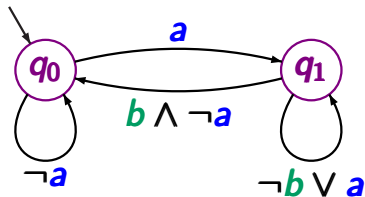
acceptance condition:

$$\diamond \square \neg q_1 \wedge \square \diamond q_0$$

For each LTL formula φ over AP there exists a DRA \mathcal{A} with the alphabet $\Sigma = 2^{AP}$ s.t.

$$\mathcal{L}_\omega(\mathcal{A}) = \{\sigma \in \Sigma^\omega : \sigma \models \varphi\} \text{ and } |\mathcal{A}| = \mathcal{O}(2^{\exp(|\varphi|)})$$

Example: $AP = \{a, b\} \rightsquigarrow \Sigma = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$

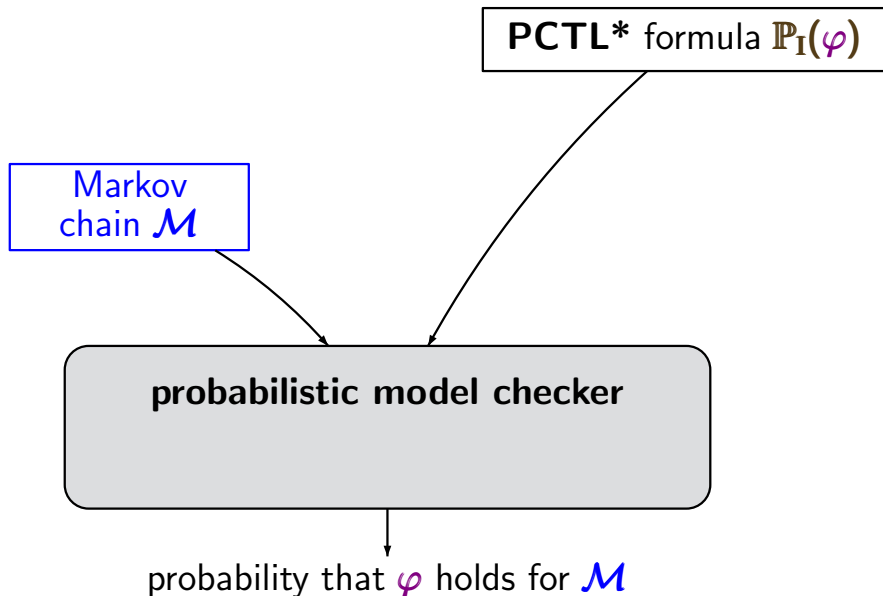


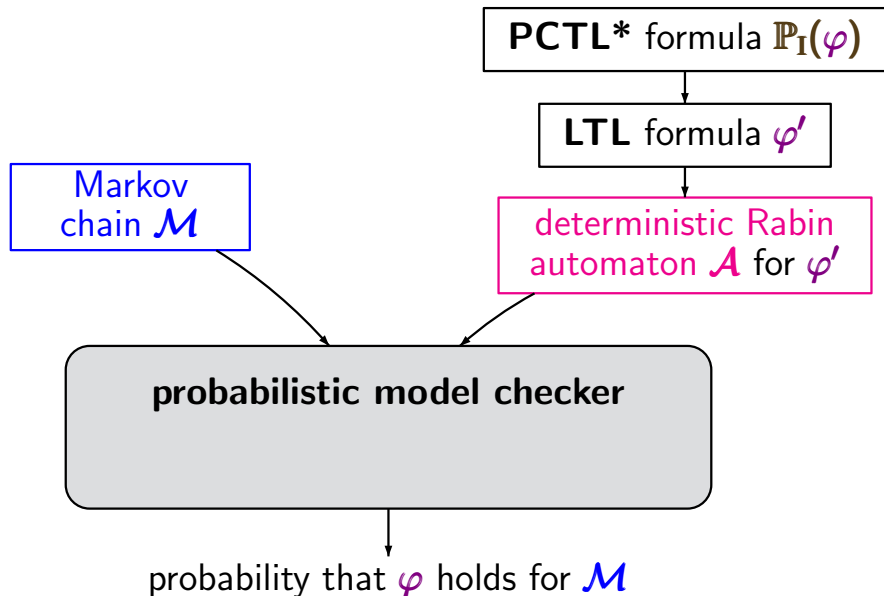
acceptance condition:

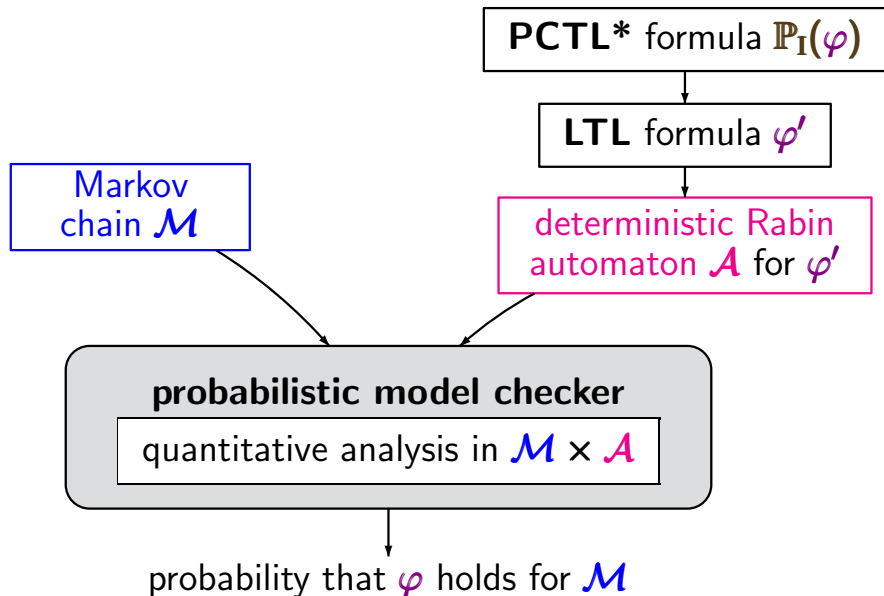
$$\diamond \square \neg q_1 \wedge \square \diamond q_0$$

LTL formula

$$\square (a \rightarrow \diamond (b \wedge \neg a)) \wedge \diamond \square \neg a$$







Product of a Markov chain and a DRA

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

goal: define a Markov chain $\mathcal{M} \times \mathcal{A}$

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

goal: define a Markov chain $\mathcal{M} \times \mathcal{A}$ such that

$$\Pr^{\mathcal{M}}(s, \mathcal{A}) = \Pr^{\mathcal{M}}\{\pi \in Paths(s) : trace(\pi) \in \mathcal{L}_{\omega}(\mathcal{A})\}$$

can be derived by a probabilistic reachability analysis
in the product-chain $\mathcal{M} \times \mathcal{A}$

Product of a Markov chain and a DRA

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

goal: define a Markov chain $\mathcal{M} \times \mathcal{A}$

path π
in \mathcal{M}



Product of a Markov chain and a DRA

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

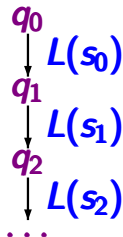
DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

goal: define a Markov chain $\mathcal{M} \times \mathcal{A}$

path π
in \mathcal{M}



run for $trace(\pi)$
in \mathcal{A}



Product of a Markov chain and a DRA

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

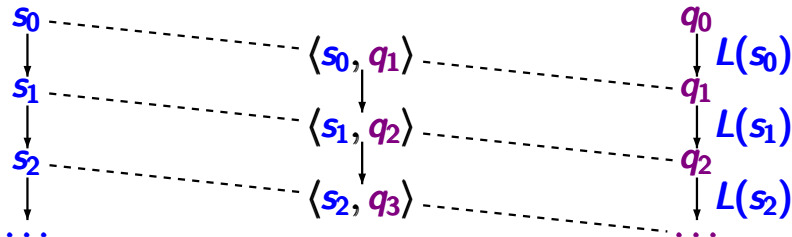
DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

goal: define a Markov chain $\mathcal{M} \times \mathcal{A}$

path π
in \mathcal{M}

path in
 $\mathcal{M} \times \mathcal{A}$

run for $trace(\pi)$
in \mathcal{A}



Product of a Markov chain and a DRA

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

Markov chain $\mathcal{M} \times \mathcal{A} = (S \times Q, P', \dots)$ where

Product of a Markov chain and a DRA

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

Markov chain $\mathcal{M} \times \mathcal{A} = (S \times Q, P', \dots)$ where

$$P'(\langle s, q \rangle, \langle s', q' \rangle) = \begin{cases} P(s, s') & : \text{if } q' = \delta(q, L(s')) \\ 0 & : \text{otherwise} \end{cases}$$

given: Markov chain $\mathcal{M} = (S, P, AP, L, s_0)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, Acc)$

Markov chain $\mathcal{M} \times \mathcal{A} = (S \times Q, P', \dots)$ where

$$P'(\langle s, q \rangle, \langle s', q' \rangle) = \begin{cases} P(s, s') & : \text{if } q' = \delta(q, L(s')) \\ 0 & : \text{otherwise} \end{cases}$$

initial state of $\mathcal{M} \times \mathcal{A}$: $\langle s_0, \delta(q_0, L(s_0)) \rangle$

Fundamental property of the product

given: Markov chain $\mathcal{M} = (S, P, AP, L)$
DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, \text{Acc})$
where $\text{Acc} = \{(L_i, U_i) : 1 \leq i \leq k\}$

Fundamental property of the product

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, \text{Acc})$

where $\text{Acc} = \{(L_i, U_i) : 1 \leq i \leq k\}$

given state $s \in S$, let $q_s = \delta(q_0, L(s))$

Fundamental property of the product

given: Markov chain $\mathcal{M} = (S, P, AP, L)$

DRA $\mathcal{A} = (Q, 2^{AP}, \delta, q_0, \text{Acc})$

where $\text{Acc} = \{(L_i, U_i) : 1 \leq i \leq k\}$

given state $s \in S$, let $q_s = \delta(q_0, L(s))$

$$\Pr^{\mathcal{M}}(s, \mathcal{A}) = \Pr^{\mathcal{M} \times \mathcal{A}}(\langle s, q_s \rangle, \bigvee_{1 \leq i \leq k} (\diamond \square \neg L_i \wedge \square \diamond U_i))$$

Fundamental property of the product

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L)$

DRA $\mathcal{A} = (\mathcal{Q}, 2^{AP}, \delta, q_0, \text{Acc})$

where $\text{Acc} = \{(L_i, U_i) : 1 \leq i \leq k\}$

given state $s \in \mathcal{S}$, let $q_s = \delta(q_0, L(s))$

$$\begin{aligned} \Pr^{\mathcal{M}}(s, \mathcal{A}) &= \Pr^{\mathcal{M} \times \mathcal{A}}(\langle s, q_s \rangle, \bigvee_{1 \leq i \leq k} (\diamond \square \neg L_i \wedge \square \diamond U_i)) \\ &= \Pr^{\mathcal{M} \times \mathcal{A}}(\langle s, q_s \rangle, \diamond \text{accBSCC}) \end{aligned}$$

Fundamental property of the product

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L)$

DRA $\mathcal{A} = (\mathcal{Q}, 2^{AP}, \delta, q_0, \text{Acc})$

where $\text{Acc} = \{(L_i, U_i) : 1 \leq i \leq k\}$

given state $s \in \mathcal{S}$, let $q_s = \delta(q_0, L(s))$

$$\begin{aligned} \Pr^{\mathcal{M}}(s, \mathcal{A}) &= \Pr^{\mathcal{M} \times \mathcal{A}}(\langle s, q_s \rangle, \bigvee_{1 \leq i \leq k} (\diamond \square \neg L_i \wedge \square \diamond U_i)) \\ &= \Pr^{\mathcal{M} \times \mathcal{A}}(\langle s, q_s \rangle, \diamond \text{accBSCC}) \end{aligned}$$

union of accepting bottom strongly connected component

Fundamental property of the product

given: Markov chain $\mathcal{M} = (\mathcal{S}, P, AP, L)$

DRA $\mathcal{A} = (\mathcal{Q}, 2^{AP}, \delta, q_0, \text{Acc})$

where $\text{Acc} = \{(L_i, U_i) : 1 \leq i \leq k\}$

given state $s \in \mathcal{S}$, let $q_s = \delta(q_0, L(s))$

$$\begin{aligned} \Pr^{\mathcal{M}}(s, \mathcal{A}) &= \Pr^{\mathcal{M} \times \mathcal{A}}(\langle s, q_s \rangle, \bigvee_{1 \leq i \leq k} (\diamond \square \neg L_i \wedge \square \diamond U_i)) \\ &= \Pr^{\mathcal{M} \times \mathcal{A}}(\langle s, q_s \rangle, \diamond \text{accBSCC}) \end{aligned}$$

union of accepting bottom strongly connected component,
i.e., BSCCs C in $\mathcal{M} \times \mathcal{A}$ s.t.

$$\exists i \in \{1, \dots, k\}. C \cap L_i = \emptyset \wedge C \cap U_i \neq \emptyset$$

given: Markov chain $\mathcal{M} = (S, P, AP, L, s_0)$

PCTL*-state formula Φ

task: check whether $\mathcal{M} \models \Phi$

model checking relies on recursive computation of

$Sat(\Psi) = \{s \in S : s \models \Psi\}$ for all sub-state formulas Ψ

- propositional logic fragment: obvious, as for **PCTL**
- probability operator $\mathbb{P}_I(\varphi)$
via **DRA** for φ and reduction to a
probabilistic reachability analysis in the product

Markov chain
 $\mathcal{M} = (\mathcal{S}, \mathcal{P}, AP, L)$

PCTL* formula $\mathbb{P}_I(\varphi)$

LTL formula φ'

DRA \mathcal{A}

probabilistic model checker
analysis of the product

return $Sat(\mathbb{P}_I(\varphi)) = \{s \in \mathcal{S} : x_{\bar{s}} \in \mathbf{I} \text{ where}$
 $\bar{s} = \langle s, \delta(q_0, L(s)) \rangle\}$

Markov chain
 $\mathcal{M} = (\mathcal{S}, P, AP, L)$

PCTL* formula $\mathbb{P}_I(\varphi)$

LTL formula φ'

DRA \mathcal{A}

product-Markov chain $\mathcal{M}' = \mathcal{M} \times \mathcal{A}$
compute the accepting BSCCs of \mathcal{M}'
compute $x_{\bar{s}} = \Pr^{\mathcal{M}'}(\bar{s}, \diamond \text{accBSCC})$

return $\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in \mathcal{S} : x_{\bar{s}} \in I \text{ where } \bar{s} = \langle s, \delta(q_0, L(s)) \rangle\}$

Markov chain
 $\mathcal{M} = (\mathcal{S}, \mathcal{P}, \mathcal{AP}, L)$

PCTL* formula $\mathbb{P}_I(\varphi)$

LTL formula φ'

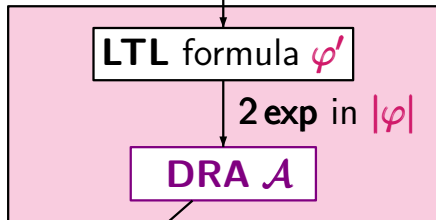
DRA \mathcal{A}

probabilistic reachability analysis
in the product-Markov chain $\mathcal{M} \times \mathcal{A}$
complexity: $\mathcal{O}(\text{poly}(|\mathcal{M}|, |\mathcal{A}|))$

return $\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in \mathcal{S} : x_{\bar{s}} \in I \text{ where}$
 $\bar{s} = \langle s, \delta(q_0, L(s)) \rangle\}$

Markov chain
 $\mathcal{M} = (\mathcal{S}, P, AP, L)$

PCTL* formula $\mathbb{P}_I(\varphi)$



probabilistic reachability analysis
in the product-Markov chain $\mathcal{M} \times \mathcal{A}$

complexity: $\mathcal{O}(\text{poly}(|\mathcal{M}|, |\mathcal{A}|))$

return $\text{Sat}(\mathbb{P}_I(\varphi)) = \{s \in \mathcal{S} : x_{\bar{s}} \in I \text{ where}$
 $\bar{s} = \langle s, \delta(q_0, L(s)) \rangle\}$

part 1: Markov chains
probabilistic computation tree logic
(PCTL/PCTL*)

part 2: Markov decision processes (MDP) ←
PCTL/PCTL* over MDP
partial order reduction
fairness

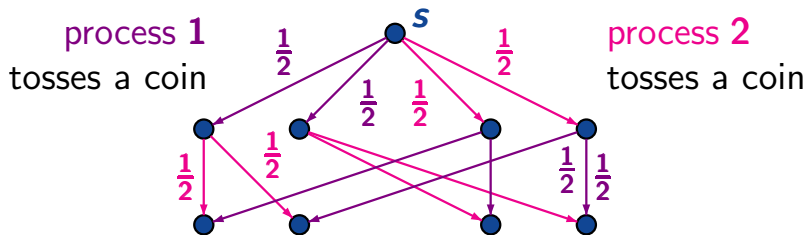
extend Markov chains by **nondeterminism**

extend Markov chains by **nondeterminism**

- modelling asynchronous distributed systems by **interleaving**

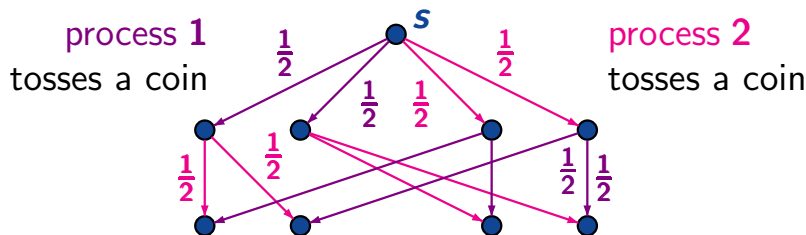
extend Markov chains by **nondeterminism**

- modelling asynchronous distributed systems by **interleaving**



extend Markov chains by **nondeterminism**

- modelling asynchronous distributed systems by **interleaving**
- useful for **abstraction purposes**
- representation of the interface with an unpredictable environment (e.g., human user)



$\mathcal{M} = (S, Act, P, AP, L)$ + initial state/distribution

$\mathcal{M} = (\mathcal{S}, \mathcal{Act}, P, AP, L)$ + initial state/distribution

- finite state space \mathcal{S}

$\mathcal{M} = (\mathcal{S}, \mathit{Act}, P, \mathit{AP}, L)$ + initial state/distribution

- finite state space \mathcal{S}
- Act finite set of actions

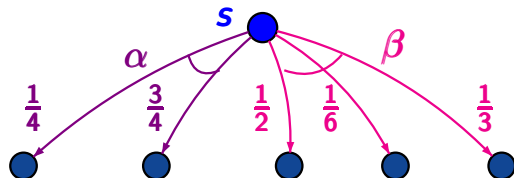
Markov decision process (MDP)

PMC-69

$\mathcal{M} = (\mathcal{S}, \mathit{Act}, P, AP, L)$ + initial state/distribution

- finite state space \mathcal{S}
- Act finite set of actions
- $P : \mathcal{S} \times \mathit{Act} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\forall s \in \mathcal{S} \quad \forall \alpha \in \mathit{Act}. \quad \sum_{s' \in \mathcal{S}} P(s, \alpha, s') \in \{0, 1\}$$



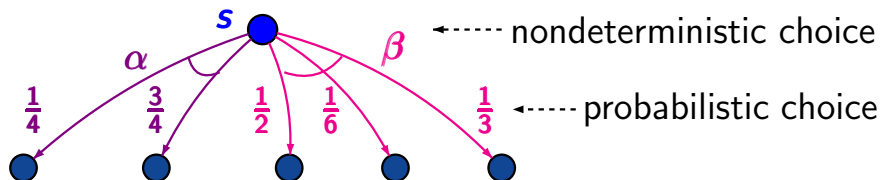
Markov decision process (MDP)

PMC-69

$\mathcal{M} = (\mathcal{S}, \mathit{Act}, P, AP, L)$ + initial state/distribution

- finite state space \mathcal{S}
- Act finite set of actions
- $P : \mathcal{S} \times \mathit{Act} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\forall s \in \mathcal{S} \quad \forall \alpha \in \mathit{Act}. \quad \sum_{s' \in \mathcal{S}} P(s, \alpha, s') \in \{0, 1\}$$



Markov decision process (MDP)

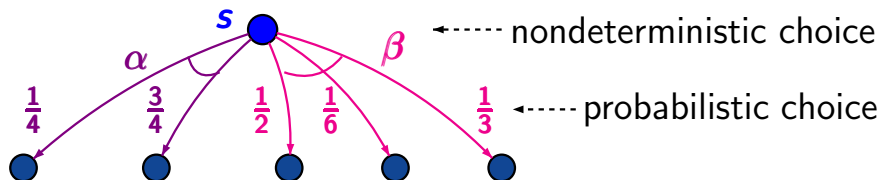
PMC-69

$\mathcal{M} = (\mathcal{S}, \mathcal{Act}, P, AP, L)$ + initial state/distribution

- finite state space \mathcal{S}
- \mathcal{Act} finite set of actions
- $P : \mathcal{S} \times \mathcal{Act} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\forall s \in \mathcal{S} \quad \forall \alpha \in \mathcal{Act}. \quad \sum_{s' \in \mathcal{S}} P(s, \alpha, s') \in \{0, 1\}$$

$\alpha \notin \mathcal{Act}(s)$ $\alpha \in \mathcal{Act}(s)$



Markov decision process (MDP)

PMC-69

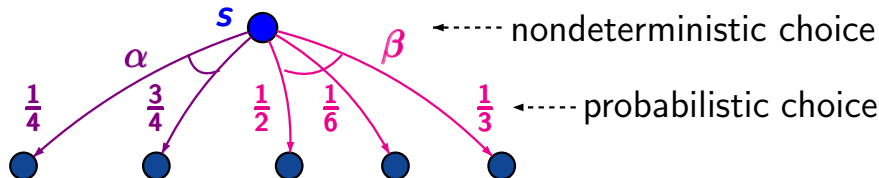
$\mathcal{M} = (\mathcal{S}, \mathcal{Act}, P, AP, L)$ + initial state/distribution

- finite state space \mathcal{S}
- \mathcal{Act} finite set of actions
- $P : \mathcal{S} \times \mathcal{Act} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\forall s \in \mathcal{S} \quad \forall \alpha \in \mathcal{Act}. \quad \sum_{s' \in \mathcal{S}} P(s, \alpha, s') \in \{0, 1\}$$

and $\mathcal{Act}(s) \neq \emptyset$

$\alpha \notin \mathcal{Act}(s)$ $\alpha \in \mathcal{Act}(s)$



$\mathcal{M} = (\mathcal{S}, \mathcal{Act}, P, \mathcal{AP}, L)$ + initial state/distribution

- finite state space \mathcal{S}
- \mathcal{Act} finite set of actions
- $P : \mathcal{S} \times \mathcal{Act} \times \mathcal{S} \rightarrow [0, 1]$ s.t.

$$\forall s \in \mathcal{S} \quad \forall \alpha \in \mathcal{Act}. \quad \sum_{s' \in \mathcal{S}} P(s, \alpha, s') \in \{0, 1\}$$

and $\mathcal{Act}(s) \neq \emptyset$

$\alpha \notin \mathcal{Act}(s)$ $\alpha \in \mathcal{Act}(s)$

- \mathcal{AP} set of atomic propositions
- labeling $L : \mathcal{S} \rightarrow 2^{\mathcal{AP}}$

Randomized mutual exclusion protocol

PMC-72

- 2 concurrent processes P_1, P_2 with 3 phases:
 - n_i noncritical actions of process P_i
 - w_i waiting phase of process P_i
 - c_i critical section of process P_i

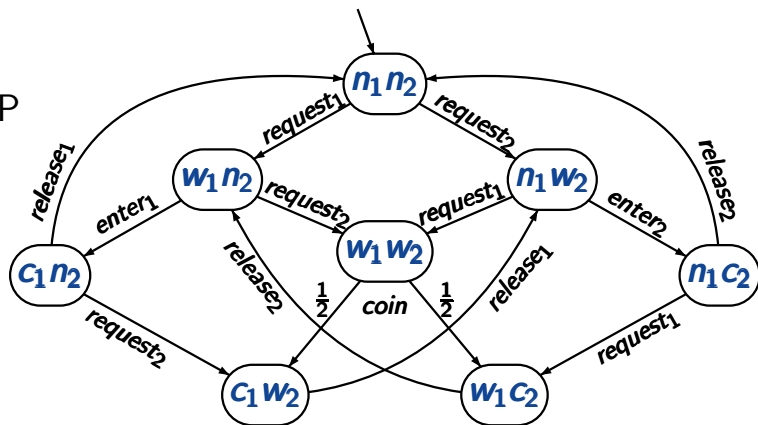
- 2 concurrent processes P_1, P_2 with 3 phases:
 - n_i noncritical actions of process P_i
 - w_i waiting phase of process P_i
 - c_i critical section of process P_i
- competition of both processes are waiting

- 2 concurrent processes P_1, P_2 with 3 phases:
 - n_i noncritical actions of process P_i
 - w_i waiting phase of process P_i
 - c_i critical section of process P_i
- competition of both processes are waiting
- resolved by a randomized arbiter who tosses a coin

Randomized mutual exclusion protocol

- interleaving of the request operations
- competition if both processes are waiting
- randomized arbiter tosses a coin if both are waiting

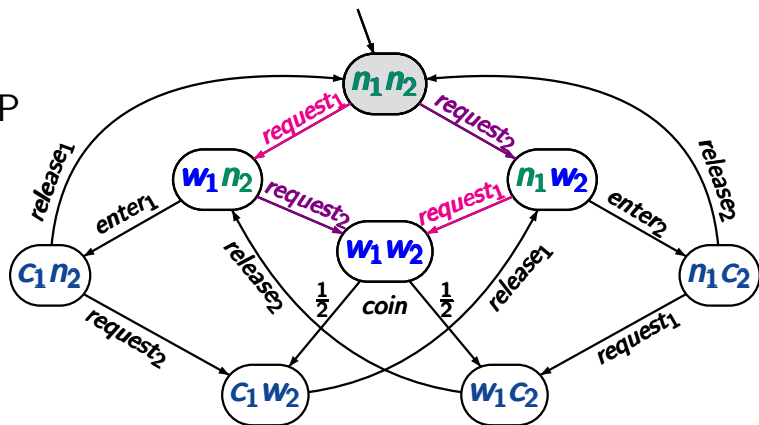
MDP



Randomized mutual exclusion protocol

- interleaving of the request operations
- competition if both processes are waiting
- randomized arbiter tosses a coin if both are waiting

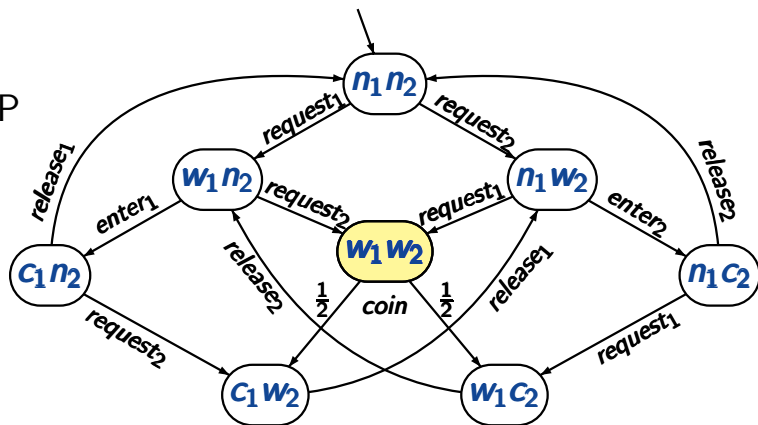
MDP



Randomized mutual exclusion protocol

- interleaving of the request operations
- **competition** if both processes are waiting
- randomized arbiter tosses a coin if both are waiting

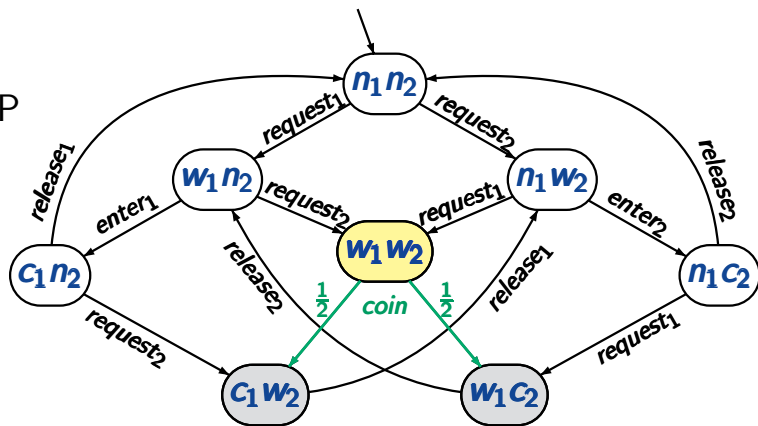
MDP



Randomized mutual exclusion protocol

- interleaving of the request operations
- competition if both processes are waiting
- **randomized arbiter** tosses a coin if both are waiting

MDP

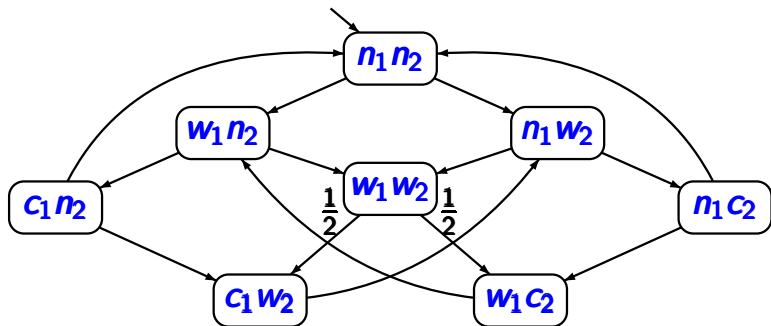


Properties of the randomized MUTEX

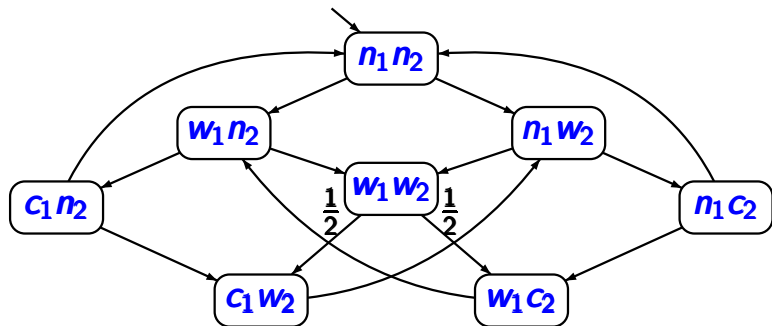
PMC-73

Properties of the randomized MUTEX

PMC-73



safety: the processes are never simultaneously in their critical section

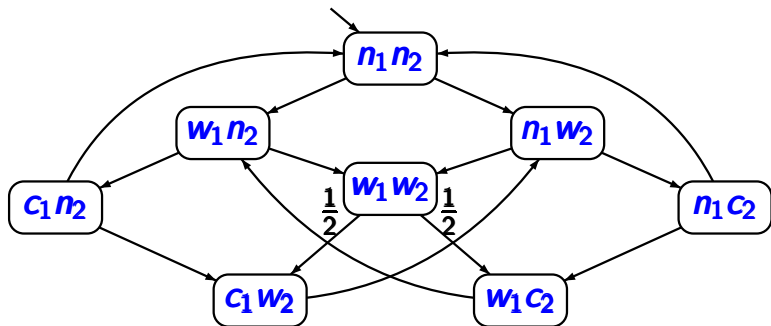


safety: the processes are never simultaneously in their critical section

holds on all paths as state $\langle c_1, c_2 \rangle$ is unreachable

Properties of the randomized MUTEX

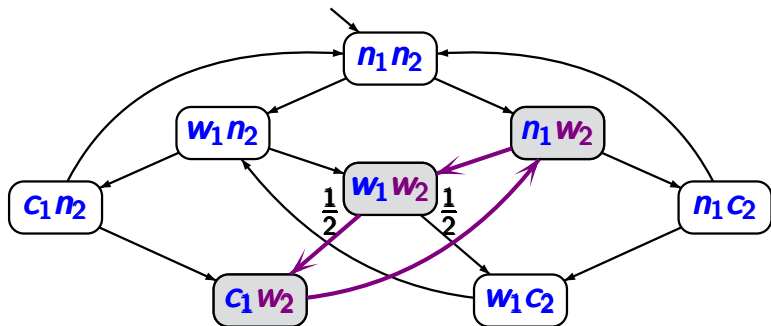
PMC-73



liveness: each waiting process will eventually enter its critical section

Properties of the randomized MUTEX

PMC-73

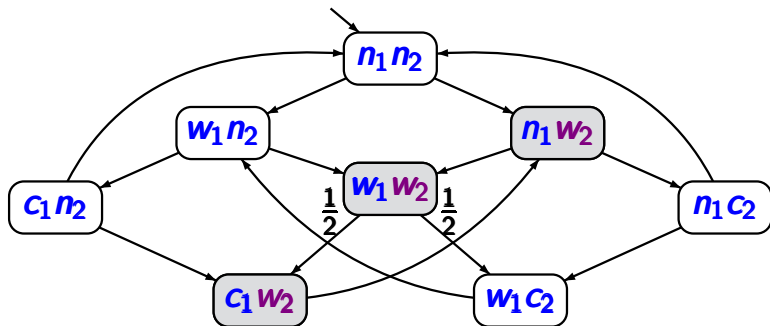


liveness: each waiting process will eventually enter its critical section

does not hold on all paths, but **almost surely**

Properties of the randomized MUTEX

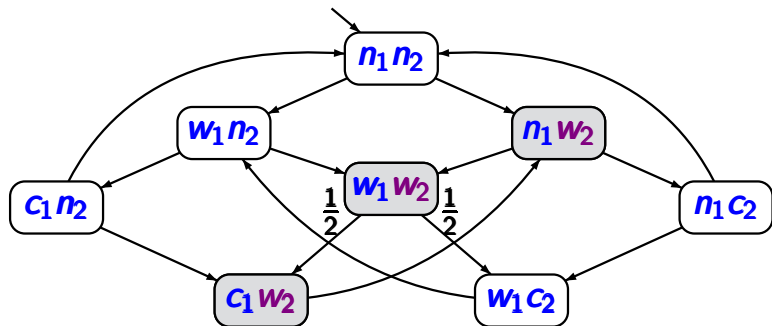
PMC-73



Suppose process 2 is waiting.

what is the probability that process 2 enters its critical section within the next 3 steps ?

Properties of the randomized MUTEX



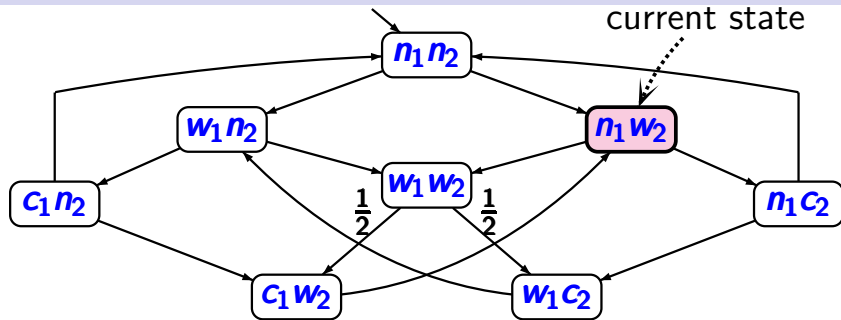
Suppose process **2** is **waiting**.

what is the probability that process **2** enters its critical section within the next **3** steps ?

... depends ...

Randomized mutual exclusion protocol

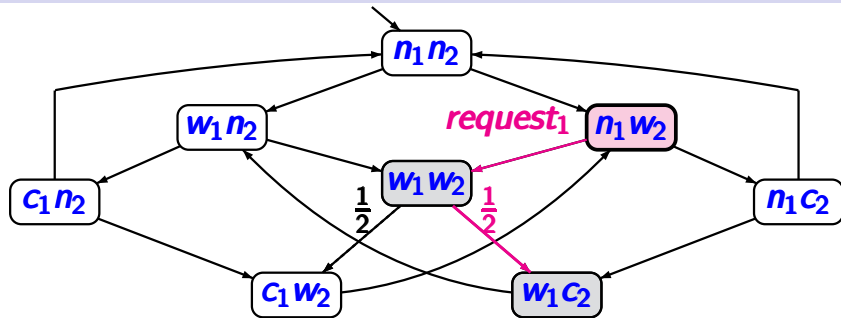
PMC-74



what is the probability that process 2 enters its critical section within the next 3 steps ?

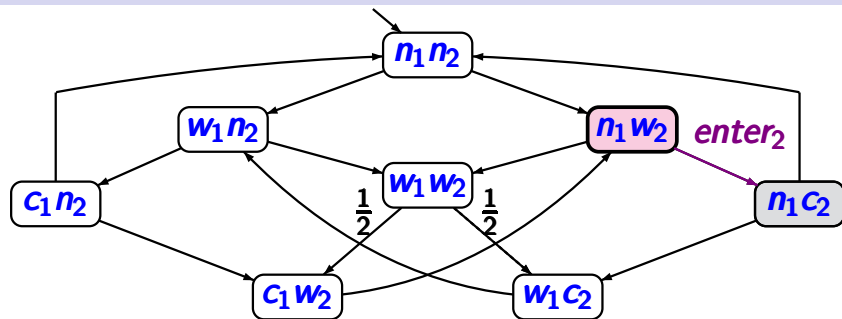
Randomized mutual exclusion protocol

PMC-74



what is the probability that process 2 enters its critical section within the next 3 steps ?

probability $\frac{1}{2}$ for the schedulers that choose process 1 in state $\langle n_1, w_2 \rangle$



what is the probability that process 2 enters its critical section within the next 3 steps ?

probability $\frac{1}{2}$ for the schedulers that choose process 1 in state $\langle n_1, w_2 \rangle$

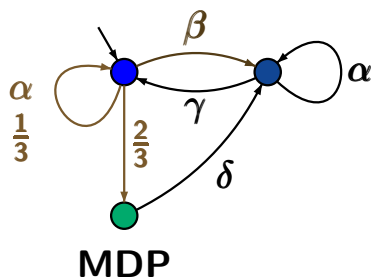
probability 1 for the schedulers that choose process 2 in $\langle n_1, w_2 \rangle$

- requires resolving the nondeterminism by schedulers

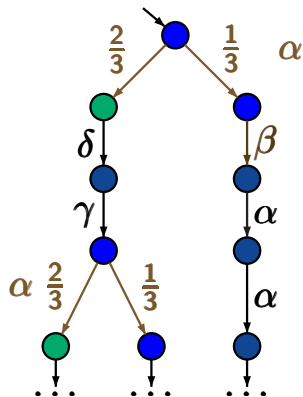
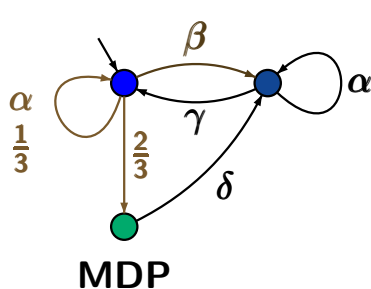
- requires resolving the nondeterminism by schedulers
- a scheduler is a function $D : S^* \longrightarrow Act$ s.t.
action $D(s_0 \dots s_n)$ is enabled in state s_n

- requires resolving the nondeterminism by schedulers
- a scheduler is a function $D : S^* \longrightarrow Act$ s.t.
action $D(s_0 \dots s_n)$ is enabled in state s_n
- each scheduler induces an infinite Markov chain

- requires resolving the nondeterminism by schedulers
- a scheduler is a function $D : S^* \rightarrow Act$ s.t. action $D(s_0 \dots s_n)$ is enabled in state s_n
- each scheduler induces an infinite Markov chain




- requires resolving the nondeterminism by schedulers
- a scheduler is a function $D : S^* \rightarrow Act$ s.t. action $D(s_0 \dots s_n)$ is enabled in state s_n
- each scheduler induces an infinite Markov chain



- requires resolving the nondeterminism by schedulers
- a scheduler is a function $D : S^* \rightarrow Act$ s.t. action $D(s_0 \dots s_n)$ is enabled in state s_n
- each scheduler induces an infinite Markov chain

yields a notion of probability measure \Pr^D
on measurable sets of infinite paths



part 1: Markov chains
probabilistic computation tree logic
(PCTL/PCTL*)

part 2: Markov decision processes (MDP)
PCTL/PCTL* over MDP
partial order reduction
fairness



- syntax of state and path formulas as for **PCTL*** over Markov chains
- probability operator $\mathbb{P}_I(\dots)$ ranges over all schedulers

state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

state formulas:

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

path formulas:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \mathbf{U} \varphi_2$$

given an MDP \mathcal{M} , define by structural induction:

- a satisfaction relation \models for states \mathbf{s} in \mathcal{M} and **PCTL*** state formulas
- a satisfaction relation \models for infinite paths π in \mathcal{M} and **PCTL*** path formulas

$s \models \text{true}$

$s \models a$ iff $a \in L(s)$

$s \models \Phi_1 \wedge \Phi_2$ iff $s \models \Phi_1$ and $s \models \Phi_2$

$s \models \neg\Phi$ iff $s \not\models \Phi$

$s \models \mathbb{P}_I(\varphi)$ iff for all schedulers D :

$\Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \} \in I$

$$s \models \text{true}$$

$$s \models a \quad \text{iff} \quad a \in L(s)$$

$$s \models \Phi_1 \wedge \Phi_2 \quad \text{iff} \quad s \models \Phi_1 \text{ and } s \models \Phi_2$$

$$s \models \neg\Phi \quad \text{iff} \quad s \not\models \Phi$$

$$s \models \mathbb{P}_I(\varphi) \quad \text{iff} \quad \text{for all schedulers } D:$$

$$\Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \} \in I$$



prob. measure in the Markov chain induced by D

$$s \models \text{true}$$

$$s \models a \quad \text{iff} \quad a \in L(s)$$

$$s \models \Phi_1 \wedge \Phi_2 \quad \text{iff} \quad s \models \Phi_1 \text{ and } s \models \Phi_2$$

$$s \models \neg\Phi \quad \text{iff} \quad s \not\models \Phi$$

$$s \models \mathbb{P}_I(\varphi) \quad \text{iff} \quad \text{for all schedulers } D:$$

$$\Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \} \in I$$



prob. measure in the Markov chain induced by D

semantics of path formulas as for Markov chains

given: MDP $\mathcal{M} = (\mathcal{S}, Act, P, AP, L, s_0)$

PCTL* state formula ϕ

task: check whether $\mathcal{M} \models \phi$

given: MDP $\mathcal{M} = (\mathcal{S}, Act, P, AP, L, s_0)$

PCTL* state formula ϕ

task: check whether $\mathcal{M} \models \phi$

main procedure as for PCTL* over Markov chains:

recursively compute the satisfaction sets

$$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$$

for all sub-state formulas Ψ of ϕ

given: MDP $\mathcal{M} = (\mathcal{S}, Act, P, AP, L, s_0)$

PCTL* state formula ϕ

task: check whether $\mathcal{M} \models \phi$

main procedure as for PCTL* over Markov chains:

recursively compute the satisfaction sets

$$Sat(\Psi) = \{s \in \mathcal{S} : s \models \Psi\}$$

for all sub-state formulas Ψ of ϕ

treatment of the propositional logic fragment: ✓

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$ or $\mathbb{P}_{< p}(\varphi)$

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$ or $\mathbb{P}_{< p}(\varphi)$

- compute the maximal probabilities for φ

$$\Pr_{\max}^{\mathcal{M}}(s, \varphi) = \sup_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$ or $\mathbb{P}_{< p}(\varphi)$

- compute the maximal probabilities for φ

$$\Pr_{\max}^M(s, \varphi) = \max_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$

- compute the maximal probabilities for φ

$$\Pr_{\max}^M(s, \varphi) = \max_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

- return $\{s \in S : \Pr_{\max}^M(s, \varphi) \leq p\}$

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$

- compute the maximal probabilities for φ

$$\Pr_{\max}^M(s, \varphi) = \max_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

- return $\{s \in S : \Pr_{\max}^M(s, \varphi) \leq p\}$

lower probability bounds $\mathbb{P}_{\geq p}(\varphi)$ or $\mathbb{P}_{> p}(\varphi)$

analogous, but minimal probabilities for φ

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$ or $\mathbb{P}_{< p}(\varphi)$

compute the maximal probabilities for φ

$$\Pr_{\max}^M(s, \varphi) = \max_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

special case $\varphi = \diamond \Psi$

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$ or $\mathbb{P}_{< p}(\varphi)$

compute the maximal probabilities for φ

$$\Pr_{\max}^M(s, \varphi) = \max_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

special case $\varphi = \diamond \Psi$

compute $\Pr_{\max}^M(s, \diamond \Psi)$ by solving a linear program

↑
maximal reachability probabilities

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$ or $\mathbb{P}_{< p}(\varphi)$

compute the maximal probabilities for φ

$$\Pr_{\max}^M(s, \varphi) = \max_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

special case $\varphi = \diamond \Psi$

compute $\Pr_{\max}^M(s, \diamond \Psi)$ by solving a linear program

general case:

upper probability bounds $\mathbb{P}_{\leq p}(\varphi)$ or $\mathbb{P}_{< p}(\varphi)$

compute the maximal probabilities for φ

$$\Pr_{\max}^M(s, \varphi) = \max_D \Pr^D \{ \pi \in \text{Paths}(s) : \pi \models \varphi \}$$

for all states s

special case $\varphi = \diamond \Psi$

compute $\Pr_{\max}^M(s, \diamond \Psi)$ by solving a **linear program**

general case:

via **DRA** for φ and maximal reachability probabilities in the product

given: MDP \mathcal{M} with state space S
set $T \subseteq S$ of goal states

task: compute $x_s = \Pr_{\max}^{\mathcal{M}}(s, \diamond T) = \max_D \Pr^D(s, \diamond T)$

given: MDP \mathcal{M} with state space S
set $T \subseteq S$ of goal states

task: compute $x_s = \Pr_{\max}^{\mathcal{M}}(s, \diamond T) = \max_D \Pr^D(s, \diamond T)$

The vector $(x_s)_{s \in S}$ is the least solution in $[0, 1]$
of the equation system

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

given: MDP \mathcal{M} with state space S
set $T \subseteq S$ of goal states

task: compute $x_s = \Pr_{\max}^{\mathcal{M}}(s, \diamond T) = \max_D \Pr^D(s, \diamond T)$

The vector $(x_s)_{s \in S}$ is the **least** solution in $[0, 1]$
of the equation system

$$x_s = 1 \text{ if } s \in T$$

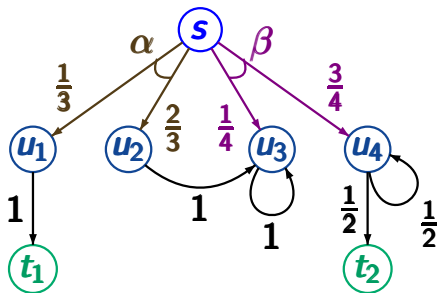
$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

Example: maximal reachability probabilities

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$



$$T = \{t_1, t_2\}$$

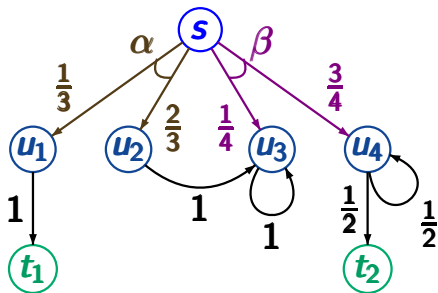
Example: maximal reachability probabilities

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\}$$



$$T = \{t_1, t_2\}$$

Example: maximal reachability probabilities

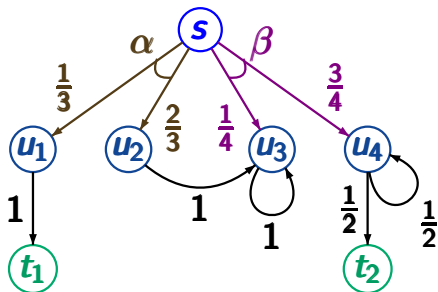
The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\}$$

$$x_{u_1} = x_{t_1}$$



$$T = \{t_1, t_2\}$$

Example: maximal reachability probabilities

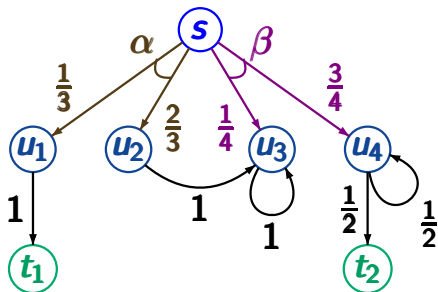
The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\}$$

$$x_{u_1} = x_{t_1} = 1$$



$$T = \{t_1, t_2\}$$

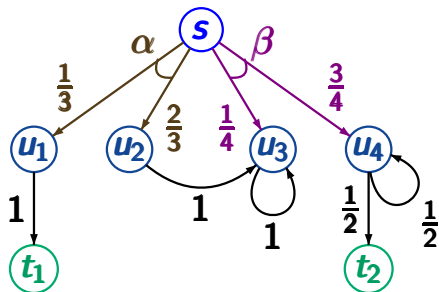
Example: maximal reachability probabilities

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\}$$



$$x_{u_1} = x_{t_1} = 1$$

$$x_{u_2} = x_{u_3}$$

$$T = \{t_1, t_2\}$$

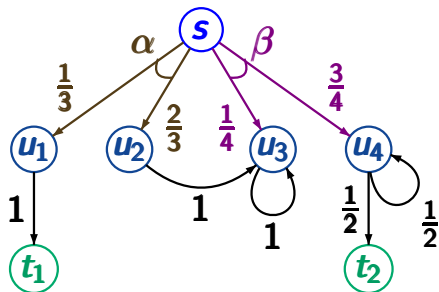
Example: maximal reachability probabilities

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the **least** solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\}$$



$$x_{u_1} = x_{t_1} = 1$$

$$x_{u_2} = x_{u_3} = 0$$

$$T = \{t_1, t_2\}$$

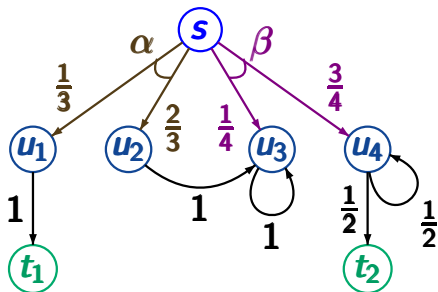
Example: maximal reachability probabilities

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\}$$



$$x_{u_1} = x_{t_1} = 1$$

$$x_{u_2} = x_{u_3} = 0$$

$$x_{t_2} = 1$$

$$T = \{t_1, t_2\}$$

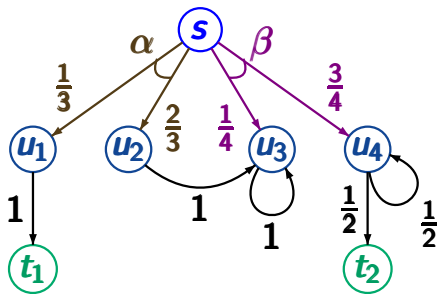
Example: maximal reachability probabilities

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\}$$



$$x_{u_1} = x_{t_1} = 1$$

$$x_{u_2} = x_{u_3} = 0$$

$$x_{t_2} = 1$$

$$\begin{aligned} x_{u_4} &= \frac{1}{2}x_{u_4} + \frac{1}{2}x_{t_2} \\ &= \frac{1}{2}x_{u_4} + \frac{1}{2} = 1 \end{aligned}$$

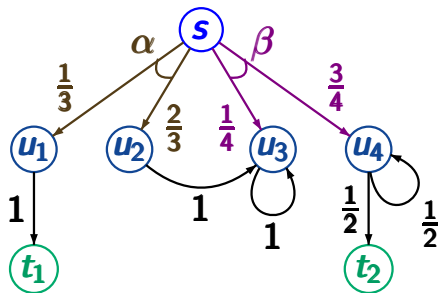
Example: maximal reachability probabilities

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the least solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

$$x_s = \max \left\{ \frac{1}{3}x_{u_1} + \frac{2}{3}x_{u_2}, \frac{1}{4}x_{u_3} + \frac{3}{4}x_{u_4} \right\} = \frac{3}{4}$$



$$x_{u_1} = x_{t_1} = 1$$

$$x_{u_2} = x_{u_3} = 0$$

$$x_{t_2} = 1$$

$$\begin{aligned} x_{u_4} &= \frac{1}{2}x_{u_4} + \frac{1}{2}x_{t_2} \\ &= \frac{1}{2}x_{u_4} + \frac{1}{2} = 1 \end{aligned}$$

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the **least** solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

and the **unique** solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = 0 \text{ if } T \text{ is not reachable from } s$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ else}$$

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the **least** solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

and the **unique** solution in $[0, 1]$ of

$$x_s = 1 \text{ iff } s \in S^1$$

$$x_s = 0 \text{ iff } s \in S^0 = \{s : s \not\vdash \exists \diamond T\}$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ else}$$

graph algorithms

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the **least** solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

and the **unique** solution in $[0, 1]$ of

$$x_s = 1 \text{ iff } s \in S^1$$

$$x_s = 0 \text{ iff } s \in S^0 = \{s : s \not\vdash \exists \diamond T\}$$

$$x_s \geq \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'}$$

$$\begin{array}{l} \text{if } s \in S^? \\ \alpha \in \text{Act}(s) \end{array}$$

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the **least** solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

and the **unique** solution in $[0, 1]$ of

$$x_s = 1 \text{ iff } s \in S^1$$

$$x_s = 0 \text{ iff } s \in S^0 = \{s : s \not\vdash \exists \diamond T\}$$

$$x_s \geq \sum_{s' \in S^?} P(s, \alpha, s') \cdot x_{s'} + P(s, \alpha, S^1) \quad \text{if } s \in S^? \\ \alpha \in \text{Act}(s)$$

where $\sum_{s \in S^?} x_s$ is minimal

The vector $(x_s)_{s \in S}$ where $x_s = \Pr_{\max}^M(s, \diamond T)$ is the **least** solution in $[0, 1]$ of

$$x_s = 1 \text{ if } s \in T$$

$$x_s = \max \left\{ \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} : \alpha \in \text{Act}(s) \right\} \text{ if } s \notin T$$

and the **unique** solution in $[0, 1]$ of

$$x_s = 1 \text{ iff } s \in S^1$$

$$x_s = 0 \text{ iff } s \in S^0 = \{s : s \not\vdash \exists \diamond T\}$$

$$x_s \geq \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'} + P(s, \alpha, S^1) \quad \text{if } s \in S^? \\ \alpha \in \text{Act}(s)$$

where $\sum_{s \in S} x_s$ is minimal

linear program

given: MDP $\mathcal{M} = (S, P, \dots)$

prefix-independent limit property E for paths

task: compute $\Pr_{\max}^{\mathcal{M}}(s, E)$

Maximal probabilities for limit properties

given: MDP $\mathcal{M} = (\mathcal{S}, P, \dots)$

prefix-independent limit property E for paths

task: compute $\Pr_{\max}^{\mathcal{M}}(s, E)$

$$\max_D \Pr^D \{ \pi \in Paths(s) : \pi \in E \}$$

Maximal probabilities for limit properties

given: MDP $\mathcal{M} = (S, P, \dots)$

prefix-independent limit property E for paths

task: compute $\Pr_{\max}^{\mathcal{M}}(s, E)$



i.e., there exists subsets T_1, \dots, T_k of S s.t.
for all paths π in \mathcal{M} :

$$\pi \models E \text{ iff } \exists i \in \{1, \dots, k\}. \inf(\pi) = T_i$$

where $\inf(s_0 s_1 s_2 \dots) = \{t \in S : \exists i \geq 0. s_i = t\}$

Let $\mathcal{M} = (S, Act, P, AP, L)$ be an MDP.

An *end component* of \mathcal{M} is a strongly connected sub-MDP

Let $\mathcal{M} = (S, Act, P, AP, L)$ be an MDP.

An *end component* of \mathcal{M} is a strongly connected sub-MDP, i.e., a pair (T, A) where $\emptyset \neq T \subseteq S$ and $A : T \rightarrow 2^{Act}$ s.t.

Let $\mathcal{M} = (S, \text{Act}, P, AP, L)$ be an MDP.

An *end component* of \mathcal{M} is a strongly connected sub-MDP, i.e., a pair (T, A) where $\emptyset \neq T \subseteq S$ and $A : T \rightarrow 2^{\text{Act}}$ s.t.

- (1) enabledness of selected actions
- (2) closed under probabilistic branching
- (3) the underlying graph is strongly connected

Let $\mathcal{M} = (S, \text{Act}, P, AP, L)$ be an MDP.

An *end component* of \mathcal{M} is a strongly connected sub-MDP, i.e., a pair (T, A) where $\emptyset \neq T \subseteq S$ and $A : T \rightarrow 2^{\text{Act}}$ s.t.

- (1) enabledness of selected actions
 $\emptyset \neq A(t) \subseteq \text{Act}(t)$ for all $t \in T$
- (2) closed under probabilistic branching
- (3) the underlying graph is strongly connected

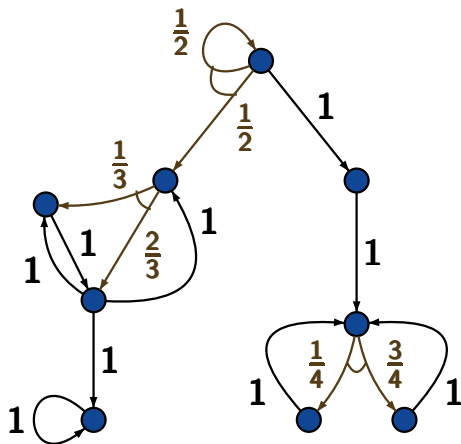
Let $\mathcal{M} = (S, \text{Act}, P, AP, L)$ be an MDP.

An *end component* of \mathcal{M} is a strongly connected sub-MDP, i.e., a pair (T, A) where $\emptyset \neq T \subseteq S$ and $A : T \rightarrow 2^{\text{Act}}$ s.t.

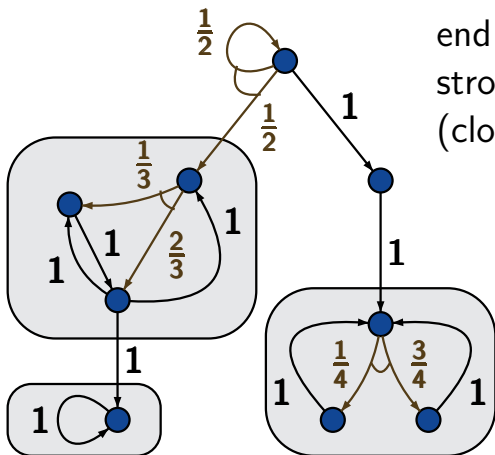
- (1) enabledness of selected actions
 $\emptyset \neq A(t) \subseteq \text{Act}(t)$ for all $t \in T$
- (2) closed under probabilistic branching
 $\forall t \in T \forall \alpha \in A(t). (P(t, \alpha, u) > 0 \longrightarrow u \in T)$
- (3) the underlying graph is strongly connected

Example: end components

PMC-80



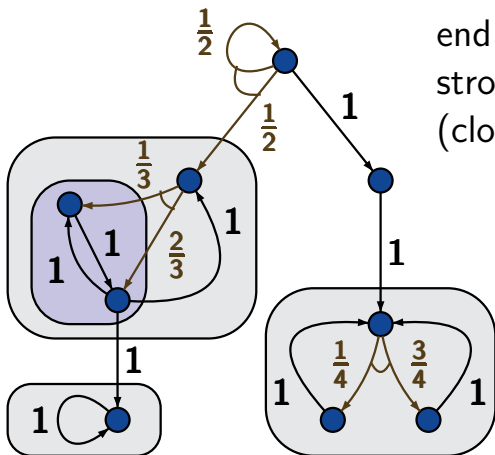
Example: end components



end component:
strongly connected sub-MDP
(closed under prob. branching)

Example: end components

PMC-80

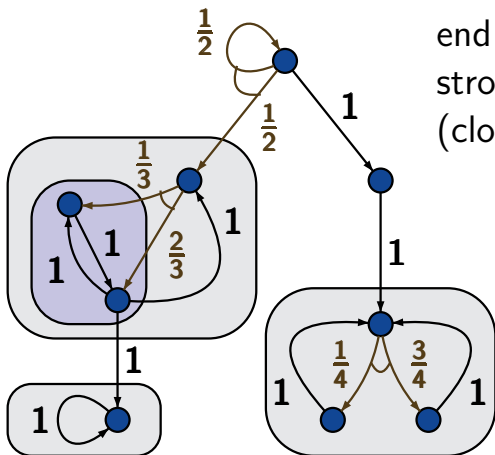


end component:
strongly connected sub-MDP
(closed under prob. branching)

Limiting behavior of MDPs

PMC-80

For all schedulers D , almost surely an **end component** will be reached and all its states visited infinitely often



end component:
strongly connected sub-MDP
(closed under prob. branching)

For all schedulers D , almost surely an **end component** will be reached and all its states visited infinitely often



i.e., for all schedulers D and states s :

$$\Pr^D \left\{ \pi \in \text{Paths}(s) : \text{inf}(\pi) \text{ constitutes an end component} \right\} = 1$$

For all schedulers D , almost surely an end component will be reached and all its states visited infinitely often



i.e., for all schedulers D and states s :

$$\Pr^D \left\{ \pi \in \text{Paths}(s) : \text{inf}(\pi) \text{ constitutes an end component} \right\} = 1$$

Let E be a limit property and $T_1, \dots, T_k \subseteq S$ s.t.

$$\pi \models E \quad \text{iff} \quad \exists i \geq 0. \text{inf}(\pi) = T_i$$

For all schedulers D , almost surely an **end component** will be reached and all its states visited infinitely often



i.e., for all schedulers D and states s :

$$\Pr^D \left\{ \pi \in \text{Paths}(s) : \text{inf}(\pi) \text{ constitutes an end component} \right\} = 1$$

Let E be a limit property and $T_1, \dots, T_k \subseteq S$ s.t.

$$\pi \models E \quad \text{iff} \quad \exists i \geq 0. \text{inf}(\pi) = T_i$$

Then: $\Pr_{\max}(s, E) = \Pr_{\max}(s, \diamond T)$ where

$$T = \bigcup \{ T_i : T_i \text{ constitutes an end component} \}$$

Let E be a Rabin condition $\bigvee_{1 \leq i \leq k} \diamond \square \neg L_i \wedge \square \diamond U_i$.

Let E be a Rabin condition $\bigvee_{1 \leq i \leq k} \diamond \square \neg L_i \wedge \square \diamond U_i$. Then:

$$\Pr_{\max}(s, E) = \Pr_{\max}(s, \diamond \text{acc}EC)$$

Let E be a Rabin condition $\bigvee_{1 \leq i \leq k} \diamond \square \neg L_i \wedge \square \diamond U_i$. Then:

$$\Pr_{\max}(s, E) = \Pr_{\max}(s, \diamond \text{acc}EC)$$



union of all end components T that “meet E ”, i.e.,
 $\exists i \in \{1, \dots, k\}. T \cap L_i = \emptyset$ and $T \cap U_i \neq \emptyset$

Let E be a Rabin condition $\bigvee_{1 \leq i \leq k} \diamond \square \neg L_i \wedge \square \diamond U_i$. Then:

$$\Pr_{\max}(s, E) = \Pr_{\max}(s, \diamond \text{accMEC})$$



$\bigcup_{1 \leq i \leq k}$ maximal end components T in $\mathcal{M} \setminus L_i$
s.t. $T \cap U_i \neq \emptyset$

Let E be a Rabin condition $\bigvee_{1 \leq i \leq k} \diamond \square \neg L_i \wedge \square \diamond U_i$. Then:

$$\Pr_{\max}(s, E) = \Pr_{\max}(s, \diamond \text{accMEC})$$



$\bigcup_{1 \leq i \leq k}$ maximal end components T in $\mathcal{M} \setminus L_i$
s.t. $T \cap U_i \neq \emptyset$

model checking algorithm for Rabin condition E :

1. compute the maximal end components
2. check which of them fulfills E
3. compute maximal reachability probabilities (linear program)

Summary: PCTL* model checking for MDP

given: MDP $\mathcal{M} = (S, Act, P, \dots)$
PCTL* star formula $\mathbb{P}_{\leq p}(\varphi)$

task: compute $Sat(\mathbb{P}_{\leq p}(\varphi))$

Summary: PCTL* model checking for MDP

given: MDP $\mathcal{M} = (S, Act, P, \dots)$

PCTL* star formula $\mathbb{P}_{\leq p}(\varphi)$

task: compute $Sat(\mathbb{P}_{\leq p}(\varphi))$

method: compute $x_s = \Pr_{\max}^{\mathcal{M}}(s, \varphi)$ via reduction to the probabilistic reachability problem

Summary: PCTL* model checking for MDP

given: MDP $\mathcal{M} = (\mathcal{S}, Act, P, \dots)$

PCTL* star formula $\mathbb{P}_{\leq p}(\varphi)$

task: compute $Sat(\mathbb{P}_{\leq p}(\varphi))$

method: compute $x_s = \Pr_{\max}^{\mathcal{M}}(s, \varphi)$ via reduction
to the probabilistic reachability problem



using DRA \mathcal{A} for φ and
linear program for $\mathcal{M} \times \mathcal{A}$

MDP \mathcal{M}

PCTL* path formula φ

MDP \mathcal{M}

PCTL* path formula φ



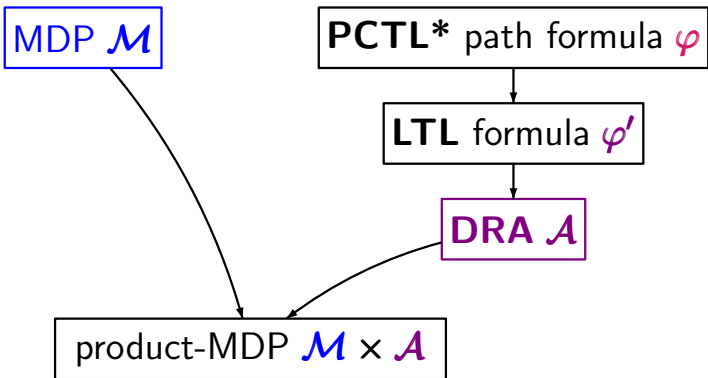
LTL formula φ'

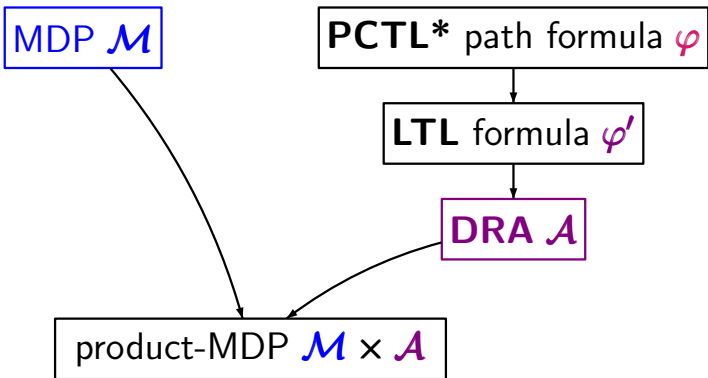
MDP \mathcal{M}

PCTL* path formula φ

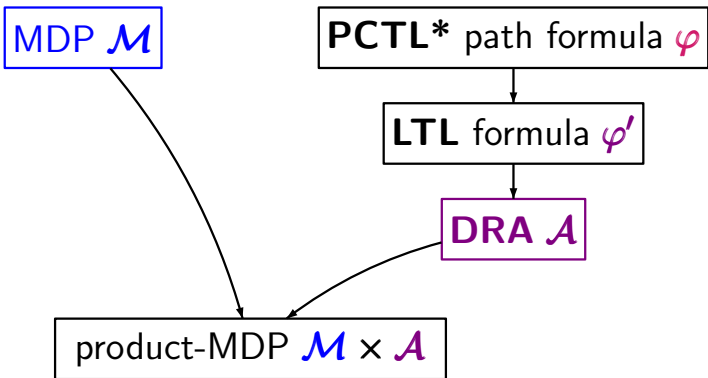
LTL formula φ'

DRA \mathcal{A}



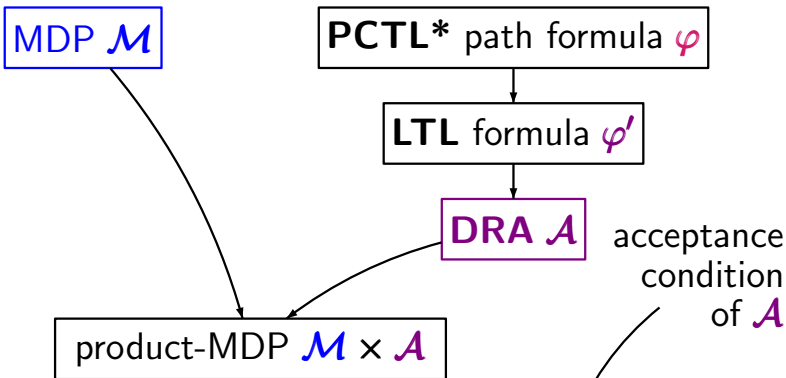


$$\Pr_{\max}^{\mathcal{M}}(s, \varphi) = \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \mathit{init}_s \rangle, \bigvee_i (\neg \diamond \square L_i \wedge \square \diamond U_i))$$



$$\Pr_{\max}^{\mathcal{M}}(s, \varphi) = \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \mathit{init}_s \rangle, \bigvee_i (\neg \diamond \square L_i \wedge \square \diamond U_i))$$

initial state in the product, if \mathcal{M} starts in s ,
 i.e., $\mathit{init}_s = \delta(q_0, L(s))$



$$\Pr_{\max}^{\mathcal{M}}(s, \varphi) = \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \mathit{init}_s \rangle, \bigvee_i (\neg \diamond \square L_i \wedge \square \diamond U_i))$$

initial state in the product, if \mathcal{M} starts in s ,
 i.e., $\mathit{init}_s = \delta(q_0, L(s))$

MDP \mathcal{M}

PCTL* path formula φ

LTL formula φ'

DRA \mathcal{A}

acceptance
condition
of \mathcal{A}

product-MDP $\mathcal{M} \times \mathcal{A}$

$$\begin{aligned}\Pr_{\max}^{\mathcal{M}}(s, \varphi) &= \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \mathit{init}_s \rangle, \bigvee_i (\neg \diamond \square L_i \wedge \square \diamond U_i)) \\ &= \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \mathit{init}_s \rangle, \diamond \mathit{accEC})\end{aligned}$$

MDP \mathcal{M}

PCTL* path formula φ

LTL formula φ'

DRA \mathcal{A}

acceptance
condition
of \mathcal{A}

product-MDP $\mathcal{M} \times \mathcal{A}$

$$\begin{aligned}\Pr_{\max}^{\mathcal{M}}(s, \varphi) &= \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \mathit{init}_s \rangle, \bigvee_i (\neg \diamond \square L_i \wedge \square \diamond U_i)) \\ &= \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \mathit{init}_s \rangle, \diamond \mathit{accEC})\end{aligned}$$

union of all end components C such that
 $\exists i. C \cap L_i = \emptyset$ and $C \cap U_i \neq \emptyset$

Complexity of PCTL/PCTL* model checking

PMC-94

	PCTL	PCTL*
Markov chain		
Markov decision process		

Complexity of PCTL/PCTL* model checking

PMC-94

	PCTL	PCTL*
Markov chain	graph algorithms + linear equation systems	
Markov decision process	graph algorithms + linear program	

Complexity of PCTL/PCTL* model checking

PMC-94

	PCTL	PCTL*
Markov chain	graph algorithms + <i>PTIME</i>	linear equation systems <i>PSPACE</i> -complete [VARDI/WOLPER'86]
Markov decision process	graph algorithms +	linear program

Complexity of PCTL/PCTL* model checking

PMC-94

	PCTL	PCTL*
Markov chain	graph algorithms + <i>PTIME</i>	linear equation systems <i>PSPACE</i> -complete [VARDI/WOLPER'86]
Markov decision process	graph algorithms + <i>PTIME</i>	linear program <i>2EXP</i> -complete [COURCOUBETIS/YANNAKAKIS'88]

Complexity of PCTL/PCTL* model checking

PMC-94

	PCTL	PCTL*
Markov chain	graph algorithms + <i>PTIME</i>	linear equation systems <i>PSPACE</i> -complete [VARDI/WOLPER'86]
Markov decision process	graph algorithms + <i>PTIME</i>	linear program <i>2EXP</i> -complete [COURCOUBETIS/YANNAKAKIS'88]

tools: e.g., PRISM (Oxford), MRMC (Aachen), LIQUOR (Dresden), ...

part 1: Markov chains
probabilistic computation tree logic
(PCTL/PCTL*)

part 2: Markov decision processes (MDP)
PCTL/PCTL* over MDP
partial order reduction
fairness



several techniques to combat the
state explosion problem

- **symbolic model checking** with variants of BDDs
e.g., in **PRISM** [Kwiatkowska/Norman/Parker]
- **state aggregation** with bisimulation
e.g., in **MRMC** [Katoen et al]
- **abstraction-refinement**
e.g., in **RAPTURE** [d'Argenio/Jeannet/Jensen/Larsen]
PASS [Hermanns/Wachter/Zhang]
- **partial order reduction**
e.g., in **LiQuor** [Baier/Ciesinski/Größer]
⋮

- **symbolic model checking** with variants of BDDs
e.g., in **PRISM** [Kwiatkowska/Norman/Parker]
- **state aggregation** with bisimulation
e.g., in **MRMC** [Katoen et al]
- **abstraction-refinement**
e.g., in **RAPTURE** [d'Argenio/Jeannet/Jensen/Larsen]
PASS [Hermanns/Wachter/Zhang]
- **partial order reduction**
e.g., in **LiQuor** [Baier/Ciesinski/Größer]
⋮

technique for reducing the state space of concurrent systems [Godefroid, Peled, Valmari, ca. 1990]

technique for reducing the state space of concurrent systems [Godefroid, Peled, Valmari, ca. 1990]

- attempts to analyze a sub-system by identifying “redundant interleavings”
- explores representatives of paths that agree up to the order of independent actions

technique for reducing the state space of concurrent systems [Godefroid, Peled, Valmari, ca. 1990]

- attempts to analyze a sub-system by identifying “redundant interleavings”
- explores representatives of paths that agree up to the order of independent actions

e.g., $\underbrace{x := x + y}_{\text{action } \alpha} \parallel \underbrace{z := z + 3}_{\text{action } \beta}$

has the same effect as $\alpha; \beta$ or $\beta; \alpha$

Partial order reduction

PMC-POR-02

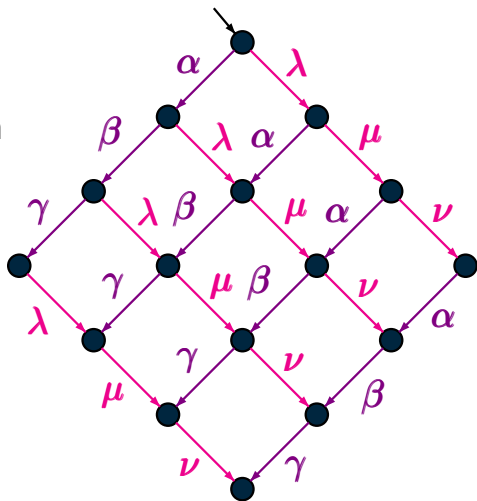
concurrent execution
of processes P_1 , P_2

- no communication
- no competition

transition system
for $P_1 \parallel P_2$ where

$$P_1 = \alpha; \beta; \gamma$$

$$P_2 = \lambda; \mu; \nu$$



Partial order reduction

PMC-POR-02

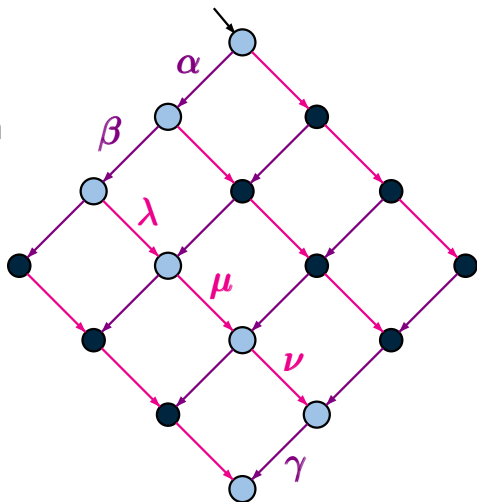
concurrent execution
of processes P_1 , P_2

- no communication
- no competition

transition system
for $P_1 \parallel P_2$ where

$$P_1 = \alpha; \beta; \gamma$$

$$P_2 = \lambda; \mu; \nu$$



idea: explore just **1** path as representative for all paths

given: transition system $\mathcal{T} = (\mathcal{S}, \mathit{Act}, \rightarrow, \dots)$

task: generate a sub-system \mathcal{T}_r by choosing appropriate
action sets $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

given: transition system $\mathcal{T} = (\mathcal{S}, \text{Act}, \rightarrow, \dots)$

task: generate a sub-system \mathcal{T}_r by choosing appropriate action sets $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$ s.t.

For each path π in \mathcal{T} there exists a path π_r in \mathcal{T}_r
s.t. $\pi \equiv_{st} \pi_r$

↑
stutter-equivalence, i.e.,
their traces agree up to repetition of state-labels

given: transition system $\mathcal{T} = (\mathcal{S}, \text{Act}, \rightarrow, \dots)$

task: generate a sub-system \mathcal{T}_r by choosing appropriate action sets $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$ s.t.

For each path π in \mathcal{T} there exists a path π_r in \mathcal{T}_r
s.t. $\pi \equiv_{st} \pi_r$

↑
stutter-equivalence, i.e.,
their traces agree up to repetition of state-labels

Hence: \mathcal{T} and \mathcal{T}_r satisfy the same stutter-invariant linear-time properties, e.g., $\text{LTL}_{\setminus \circ}$ formulas

given: transition system $\mathcal{T} = (\mathcal{S}, \text{Act}, \rightarrow, \dots)$

task: generate a sub-system \mathcal{T}_r by choosing appropriate action sets $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$ s.t.

For each path π in \mathcal{T} there exists a path π_r in \mathcal{T}_r
s.t. $\pi \equiv_{st} \pi_r$

probabilistic case: generate a sub-MDP \mathcal{M}_r from \mathcal{M} s.t.

\mathcal{M}_r and \mathcal{M} have the same **extremal probabilities**
for stutter-invariant linear-time properties

given: transition system $\mathcal{T} = (\mathcal{S}, \text{Act}, \rightarrow, \dots)$

task: generate a sub-system \mathcal{T}_r by choosing appropriate action sets $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$ s.t.

For each path π in \mathcal{T} there exists a path π_r in \mathcal{T}_r
s.t. $\pi \equiv_{st} \pi_r$

probabilistic case: generate a sub-MDP \mathcal{M}_r from \mathcal{M} s.t.

For all schedulers D for \mathcal{M} there is a scheduler D_r for \mathcal{M}_r s.t. for all measurable, stutter-invariant events E :

$$\Pr_{\mathcal{M}}^D(E) = \Pr_{\mathcal{M}_r}^{D_r}(E)$$

Actions α and β are called independent in a transition system \mathcal{T} iff:

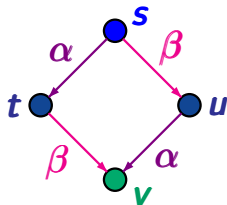
whenever $s \xrightarrow{\alpha} t$ and $s \xrightarrow{\beta} u$ then

- (1) α is enabled in u
- (2) β is enabled in t
- (3) if $u \xrightarrow{\alpha} v$ and $t \xrightarrow{\beta} w$ then $v = w$

Actions α and β are called independent in a transition system \mathcal{T} iff:

whenever $s \xrightarrow{\alpha} t$ and $s \xrightarrow{\beta} u$ then

- (1) α is enabled in u
- (2) β is enabled in t
- (3) if $u \xrightarrow{\alpha} v$ and $t \xrightarrow{\beta} w$ then $v = w$



Let $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ be a MDP and $\alpha, \beta \in \text{Act}$.

α and β are independent in \mathcal{M} if for each state s
s.t. $\alpha, \beta \in \text{Act}(s)$:

- (1) if $P(s, \alpha, t) > 0$ then $\beta \in \text{Act}(t)$
- (2) if $P(s, \beta, u) > 0$ then $\alpha \in \text{Act}(u)$
- (3) ...

Let $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ be a MDP and $\alpha, \beta \in \text{Act}$.

α and β are independent in \mathcal{M} if for each state s
s.t. $\alpha, \beta \in \text{Act}(s)$:

- (1) if $P(s, \alpha, t) > 0$ then $\beta \in \text{Act}(t)$
- (2) if $P(s, \beta, u) > 0$ then $\alpha \in \text{Act}(u)$
- (3) for all states w :

$$P(s, \alpha\beta, w) = P(s, \beta\alpha, w)$$

Let $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ be a MDP and $\alpha, \beta \in \text{Act}$.

α and β are independent in \mathcal{M} if for each state s
s.t. $\alpha, \beta \in \text{Act}(s)$:

- (1) if $P(s, \alpha, t) > 0$ then $\beta \in \text{Act}(t)$
- (2) if $P(s, \beta, u) > 0$ then $\alpha \in \text{Act}(u)$
- (3) for all states w :

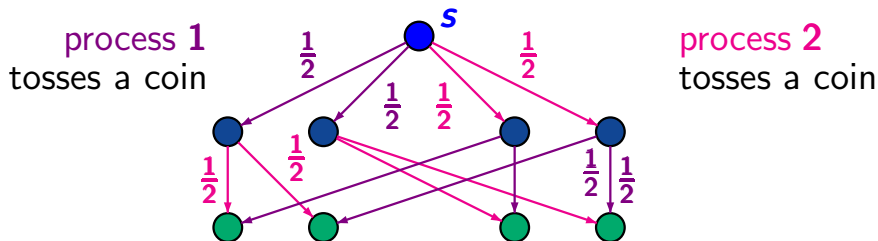
$$P(s, \alpha\beta, w) = P(s, \beta\alpha, w)$$

$$\sum_{t \in \mathcal{S}} P(s, \alpha, t) \cdot P(t, \beta, w)$$

$$\sum_{u \in \mathcal{S}} P(s, \beta, u) \cdot P(u, \alpha, w)$$

α and β are independent in \mathcal{M} if for each state s s.t. $\alpha, \beta \in \text{Act}(s)$:

- (1) if $P(s, \alpha, t) > 0$ then $\beta \in \text{Act}(t)$
- (2) if $P(s, \beta, u) > 0$ then $\alpha \in \text{Act}(u)$
- (3) for all states w : $P(s, \alpha\beta, w) = P(s, \beta\alpha, w)$



idea: use Peled's conditions for the ample sets

idea: use Peled's conditions for the ample sets

$$(A0) \quad \emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$$

idea: use Peled's conditions for the ample sets

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

(A1) stutter condition: if $\mathit{ample}(s) \neq \mathit{Act}(s)$ then all actions $\alpha \in \mathit{ample}(s)$ are stutter actions



i.e., have no visible effect on the labels of the states

idea: use Peled's conditions for the ample sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition: if $\text{ample}(s) \neq \text{Act}(s)$ then all actions $\alpha \in \text{ample}(s)$ are stutter actions

(A2) dependency condition:

For each path $s \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} \xrightarrow{\beta}$ in \mathcal{M} s.t. β is dependent on some action in $\text{ample}(s)$, there exists $i \in \{1, \dots, n\}$ with $\alpha_i \in \text{ample}(s)$

idea: use Peled's conditions for the ample sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition: if $\text{ample}(s) \neq \text{Act}(s)$ then all actions $\alpha \in \text{ample}(s)$ are stutter actions

(A2) dependency condition:

For each path $s \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} \xrightarrow{\beta}$ in \mathcal{M} s.t. β is dependent on some action in $\text{ample}(s)$, there exists $i \in \{1, \dots, n\}$ with $\alpha_i \in \text{ample}(s)$

Hence: if $\alpha \in \text{ample}(s)$ and $\beta \in \text{Act}(s) \setminus \text{ample}(s)$ then α and β are independent

idea: use Peled's conditions for the ample sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition: if $\text{ample}(s) \neq \text{Act}(s)$ then all actions $\alpha \in \text{ample}(s)$ are stutter actions

(A2) dependency condition: ...

(A3) cycle condition:

for each cycle $s_0 s_1 \dots s_n$ in \mathcal{M}_r and each action $\alpha \in \bigcap_{1 \leq i \leq n} \text{Act}(s_i)$ we have: $\alpha \in \bigcup_{1 \leq i \leq n} \text{ample}(s_i)$

idea: use Peled's conditions for the ample sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition: if $\text{ample}(s) \neq \text{Act}(s)$ then all actions $\alpha \in \text{ample}(s)$ are stutter actions

(A2) dependency condition: ...

(A3) cycle condition:

for each cycle $s_0 s_1 \dots s_n$ in \mathcal{M}_r and each action $\alpha \in \bigcap_{1 \leq i \leq n} \text{Act}(s_i)$ we have: $\alpha \in \bigcup_{1 \leq i \leq n} \text{ample}(s_i)$

By (A0)-(A3): for all paths π in \mathcal{M} there is a path π_r in \mathcal{M}_r with $\pi \equiv_{st} \pi_r$

idea: use Peled's conditions for the ample sets

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

(A1) stutter condition: if $\mathit{ample}(s) \neq \mathit{Act}(s)$ then all actions $\alpha \in \mathit{ample}(s)$ are stutter actions

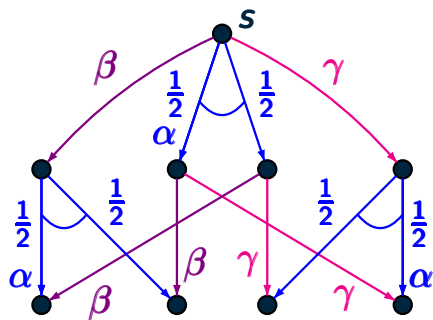
(A2) dependency condition: ...

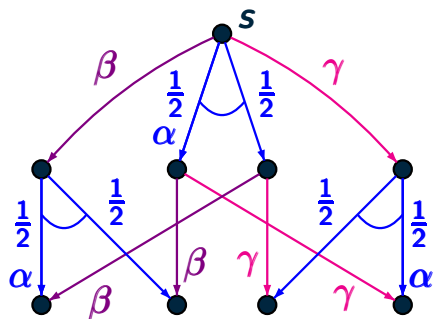
(A3) end component condition:

for each end component T in \mathcal{M}_r and each action $\alpha \in \bigcap_{t \in T} \mathit{Act}(t)$ we have: $\alpha \in \bigcup_{t \in T} \mathit{ample}(t)$

By (A0)-(A3): for almost all paths π in \mathcal{M} there is a path π_r in \mathcal{M}_r with $\pi \equiv_{st} \pi_r$

Peled's conditions (A0)-(A3) are **not sufficient**
to preserve **maximal reachability probabilities**

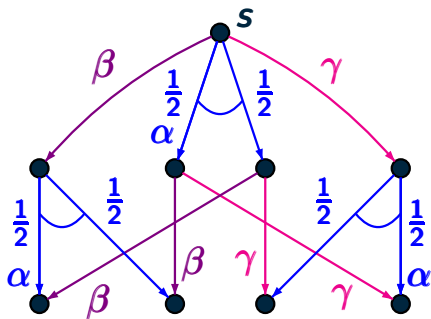




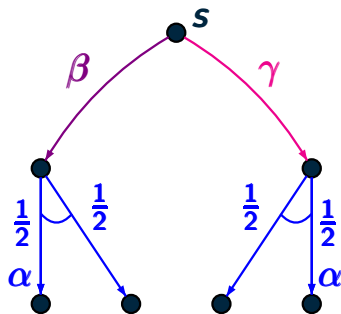
- α independent from β and γ
- α , β and γ are stutter actions

Partial order reduction for MDPs

PMC-POR-08



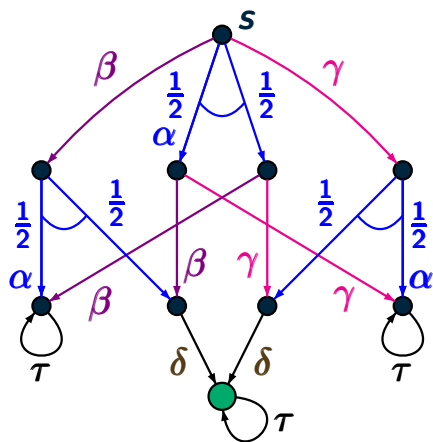
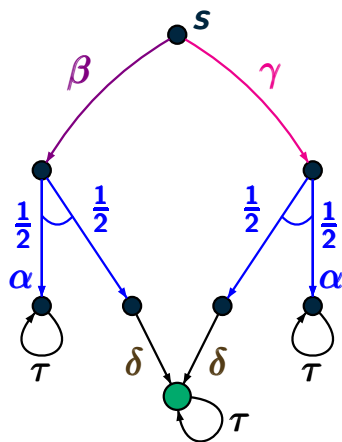
original MDP \mathcal{M}



reduced MDP \mathcal{M}_r

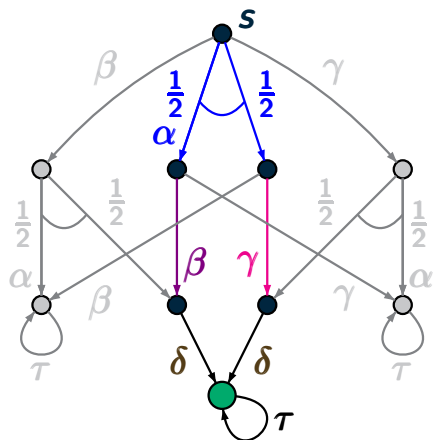
Partial order reduction for MDPs

PMC-POR-08

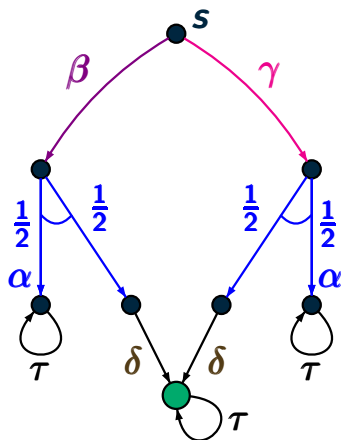
original MDP \mathcal{M} reduced MDP \mathcal{M}_r

Partial order reduction for MDPs

PMC-POR-08

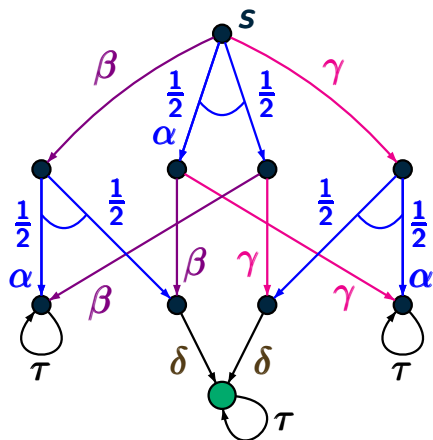


original MDP \mathcal{M}

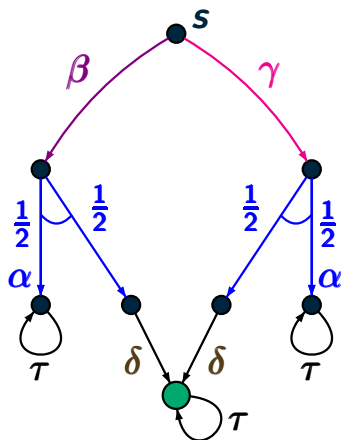


reduced MDP \mathcal{M}_r

$$\Pr_{\max}^{\mathcal{M}}(s, \diamond \text{green}) = 1$$



original MDP \mathcal{M}



reduced MDP \mathcal{M}_r

$$\Pr_{\max}^{\mathcal{M}}(s, \Diamond \text{green}) = 1 < \frac{1}{2} = \Pr_{\max}^{\mathcal{M}_r}(s, \Diamond \text{green})$$

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following **branching condition**

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following branching condition

(A4) if there is a path $s \xrightarrow{\alpha_1} \xrightarrow{\alpha_2} \dots \xrightarrow{\alpha_n} \xrightarrow{\beta}$ in \mathcal{M}_r s.t.

- $\alpha_1, \dots, \alpha_n, \beta \notin \text{ample}(s)$ and
- β is probabilistic

then $|\text{ample}(s)| = 1$

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following branching condition

(A4) $|\text{ample}(s)| = 1$ or $\text{ample}(s) = \text{Act}(s)$

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following branching condition

(A4) $|\mathit{ample}(s)| = 1$ or $\mathit{ample}(s) = \mathit{Act}(s)$

If (A0)-(A4) hold then \mathcal{M} and \mathcal{M}_r have the same **extremal probabilities** for all $\text{LTL}_{\setminus \text{O}}$ formulas.

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following branching condition

(A4) $|\text{ample}(s)| = 1$ or $\text{ample}(s) = \text{Act}(s)$

If (A0)-(A4) hold then \mathcal{M} and \mathcal{M}_r satisfy the same $\text{CTL}_{\setminus O}$ formulas [Gerth et al, 1995]

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following branching condition

(A4) $|\mathit{ample}(s)| = 1$ or $\mathit{ample}(s) = \mathit{Act}(s)$

If (A0)-(A4) hold then \mathcal{M} and \mathcal{M}_r satisfy the same $\mathbf{PCTL}_{\setminus O}$ formulas ?

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

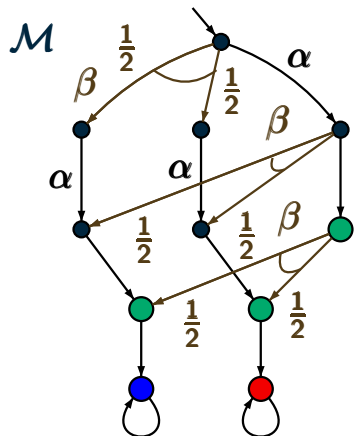
by the following branching condition

(A4) $|\text{ample}(s)| = 1$ or $\text{ample}(s) = \text{Act}(s)$

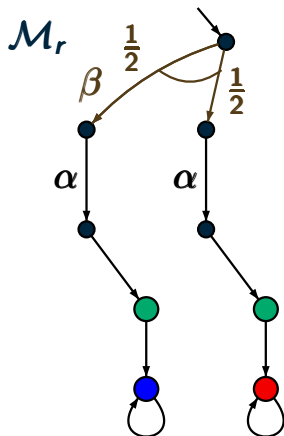
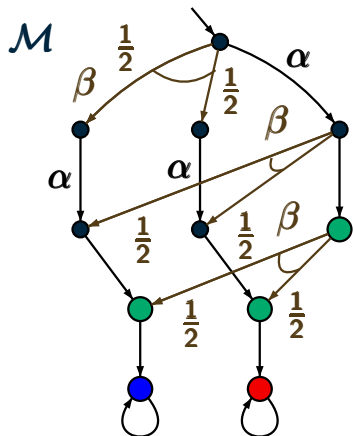
If (A0)-(A4) hold then \mathcal{M} and \mathcal{M}_r satisfy the same **PCTL_∅** formulas ?

no

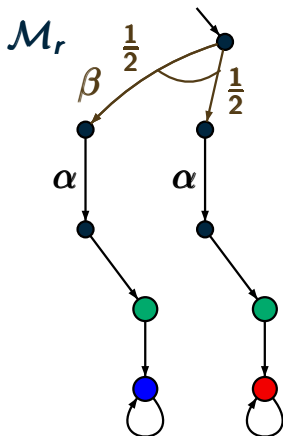
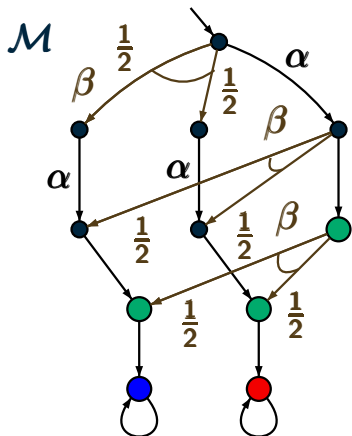
(A0)-(A4) not sufficient for PCTL



(A0)-(A4) not sufficient for PCTL

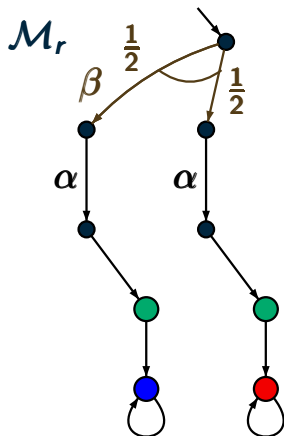
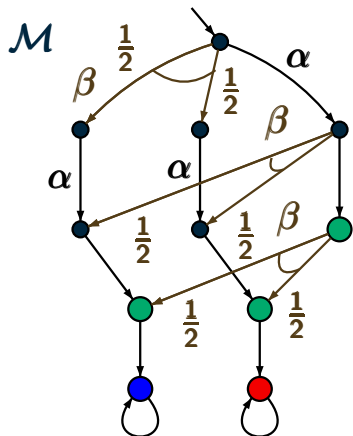


(A0)-(A4) not sufficient for PCTL



(A0)-(A4) hold

(A0)-(A4) not sufficient for PCTL



(A0)-(A4) hold, but $\mathcal{M} \not\models \Phi$ and $\mathcal{M}_r \models \Phi$ where

$$\Phi = \mathbb{P}_{=1} \left(\square (\text{green} \rightarrow (\mathbb{P}_{=1}(\diamond \text{blue}) \vee \mathbb{P}_{=1}(\diamond \text{red}))) \right)$$

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \text{ample}(s) \subseteq \text{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following branching condition

(A4') $\text{ample}(s) = \text{Act}(s)$ or $\text{ample}(s) = \{\alpha\}$ for
some nonprobabilistic action α

extend Peled's conditions for the ample-sets

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle or end component condition

by the following branching condition

(A4') $\mathit{ample}(s) = \mathit{Act}(s)$ or $\mathit{ample}(s) = \{\alpha\}$ for
some nonprobabilistic action α

If (A0)-(A3) and (A4') hold then \mathcal{M} and \mathcal{M}_r are
bisimilar and satisfy the same $\mathbf{PCTL}^*_{\setminus \bigcirc}$ formulas

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i

$\mathit{Act}_i(s)$ = action set of process \mathcal{P}_i enabled in state s

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

(A1) stutter condition

(A2) dependency condition

(A3) cycle/end component condition

(A4) branching condition

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i

- (A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$ ← local
- (A1) stutter condition ← local
- (A2) dependency condition
- (A3) cycle/end component condition
- (A4) branching condition ← local

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i

(A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$

← local

(A1) stutter condition

← local

(A2) dependency condition

← global in \mathcal{M}

(A3) cycle/end component condition

← global in \mathcal{M}_r

(A4) branching condition

← local

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i
- checks local conditions (A0), (A1) and (A4)
- realizes stronger conditions than (A2) and (A3)

- | | | |
|---|---|---------------------------|
| (A0) $\emptyset \neq \mathit{ample}(s) \subseteq \mathit{Act}(s)$ | ← | local |
| (A1) stutter condition | ← | local |
| (A2) dependency condition | ← | global in \mathcal{M} |
| (A3) cycle/end component condition | ← | global in \mathcal{M}_r |
| (A4) branching condition | ← | local |

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i
- checks local conditions (A0), (A1) and (A4)
- realizes **stronger conditions** than (A2) and (A3)

(A2) dependency condition

(A3) cycle condition

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i
- checks local conditions (A0), (A1) and (A4)
- realizes **stronger conditions** than (A2) and (A3)

(A2) dependency condition

(A3) cycle condition

if DFS detects a backward edge $t \rightarrow s$
then $\mathit{ample}(s) = \mathit{Act}(s)$

suppose \mathcal{M} is an MDP for $\mathcal{P}_1 \parallel \dots \parallel \mathcal{P}_n$

DFS-based on-the-fly generation of \mathcal{M}_r

- tries to define $\mathit{ample}(s) = \mathit{Act}_i(s)$ for some i
- checks local conditions (A0), (A1) and (A4)
- realizes **stronger conditions** than (A2) and (A3)

(A2) dependency condition

replace with (A2) with a global dependency condition
on the **control flow graphs** for $\mathcal{P}_1, \dots, \mathcal{P}_n$

(A3) cycle condition

if DFS detects a backward edge $t \rightarrow s$
then $\mathit{ample}(s) = \mathit{Act}(s)$

- part 1: Markov chains
probabilistic computation tree logic
(PCTL/PCTL*)
- part 2: Markov decision processes (MDP)
PCTL/PCTL* over MDP
partial order reduction
MDP with **fairness**



- extend Markov chains by nondeterminism
- modelling asynchronous distributed systems by interleaving

- extend Markov chains by nondeterminism
- modelling asynchronous distributed systems by interleaving

↑

verification of liveness properties
(qualitative or quantitative)
often requires fairness assumptions

- extend Markov chains by nondeterminism
- modelling asynchronous distributed systems by interleaving

↑
verification of liveness properties
(qualitative or quantitative)
often requires fairness assumptions

e.g., strong process fairness:

$\square \diamond$ process P is enabled \longrightarrow $\square \diamond$ actions of process P are taken

- extend Markov chains by **nondeterminism**
- modelling asynchronous distributed systems by **interleaving**

↑
verification of **liveness properties**
(qualitative or quantitative)
often requires **fairness assumptions**

general case: fairness assumptions impose restrictions on the resolution of nondeterminism to **rule out unrealistic behaviors**

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$

A fairness assumption \mathcal{F} for \mathcal{M} is a conjunction of limit properties of the form:

unconditional fairness $\Box \Diamond V$

strong fairness $\Box \Diamond U \rightarrow \Box \Diamond V$

weak fairness $\Diamond \Box U \rightarrow \Box \Diamond V$

where $U, V \subseteq \mathcal{S}$

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$

A fairness assumption \mathcal{F} for \mathcal{M} is a conjunction of limit properties of the form:

unconditional fairness $\Box\Diamond V$

strong fairness $\Box\Diamond U \rightarrow \Box\Diamond V$

weak fairness $\Diamond\Box U \rightarrow \Box\Diamond V$

where $U, V \subseteq \mathcal{S}$



here: just state-based fairness conditions

action-based fairness conditions can be added, e.g.,

$\Box\Diamond \text{enabled}(A) \rightarrow \Box\Diamond \text{taken}(A)$ where $A \subseteq \text{Act}$

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$

A fairness assumption \mathcal{F} for \mathcal{M} is a conjunction of limit properties of the form:

unconditional fairness $\Box \Diamond V$

strong fairness $\Box \Diamond U \rightarrow \Box \Diamond V$

weak fairness $\Diamond \Box U \rightarrow \Box \Diamond V$

where $U, V \subseteq \mathcal{S}$

Scheduler D for \mathcal{M} is called \mathcal{F} -fair iff

$$\Pr^D(\mathbf{s}, \mathcal{F}) = 1 \quad \text{for all reachable states } \mathbf{s}$$

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ and
a fairness assumption \mathcal{F} for \mathcal{M}

scheduler D is called \mathcal{F} -fair iff $\Pr^D(\mathbf{s}, \mathcal{F}) = 1$
for all states $\mathbf{s} \in \mathcal{S}$

\mathcal{F} is realizable iff there exists a \mathcal{F} -fair scheduler

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ and
a fairness assumption \mathcal{F} for \mathcal{M}

scheduler D is called \mathcal{F} -fair iff $\Pr^D(s, \mathcal{F}) = 1$
for all states $s \in \mathcal{S}$

\mathcal{F} is realizable iff there exists a \mathcal{F} -fair scheduler
iff $s \models \exists \diamond \text{FairMEC}$ for all $s \in \mathcal{S}$

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ and
a fairness assumption \mathcal{F} for \mathcal{M}

scheduler D is called \mathcal{F} -fair iff $\Pr^D(s, \mathcal{F}) = 1$
for all states $s \in \mathcal{S}$

\mathcal{F} is realizable iff there exists a \mathcal{F} -fair scheduler
iff $s \models \exists \diamond \text{FairMEC}$ for all $s \in \mathcal{S}$

↑
union of all maximal end components that contain
a sub-component T where \mathcal{F} holds

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ and
a fairness assumption \mathcal{F} for \mathcal{M}
e.g., $\mathcal{F} = \Box \Diamond U \rightarrow \Box \Diamond V$

scheduler D is called \mathcal{F} -fair iff $\Pr^D(s, \mathcal{F}) = 1$
for all states $s \in \mathcal{S}$

\mathcal{F} is realizable iff there exists a \mathcal{F} -fair scheduler
iff $s \models \exists \Diamond \text{FairMEC}$ for all $s \in \mathcal{S}$

union of all maximal end components that contain
a sub-component T where \mathcal{F} holds, i.e.,

$$U \cap T = \emptyset \text{ or } V \cap T \neq \emptyset$$

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ and
a fairness assumption \mathcal{F} for \mathcal{M}
e.g., $\mathcal{F} = \square\lozenge U \rightarrow \square\lozenge V$

scheduler D is called \mathcal{F} -fair iff $\Pr^D(s, \mathcal{F}) = 1$
for all states $s \in \mathcal{S}$

\mathcal{F} is realizable iff there exists a \mathcal{F} -fair scheduler
iff $s \models \exists\lozenge \text{FairMEC}$ for all $s \in \mathcal{S}$

poly-time algorithm for computing *FairMEC*:

... recursive computation of maximal
end components in sub-MDPs ...

- syntax of state and path formulas as before

- syntax of state and path formulas as before
- semantics as for standard **PCTL*** over MDP, but:

$$s \models_{\mathcal{F}} \mathbb{P}_I(\varphi) \quad \text{iff} \quad \text{for all } \mathcal{F}\text{-fair schedulers } D:$$
$$\Pr^D(s, \varphi) \in I$$

- syntax of state and path formulas as before
- semantics as for standard **PCTL*** over MDP, but:

$$s \models_{\mathcal{F}} \mathbb{P}_I(\varphi) \quad \text{iff} \quad \text{for all } \mathcal{F}\text{-fair schedulers } D: \\ \Pr^D(s, \varphi) \in I$$

simple cases: e.g., if \mathcal{F} is realizable then

$$s \models_{\mathcal{F}} \mathbb{P}_{\leq p}(\diamond b) \quad \text{iff} \quad s \models \mathbb{P}_{\leq p}(\diamond b)$$

- syntax of state and path formulas as before
- semantics as for standard **PCTL*** over MDP, but:

$$s \models_{\mathcal{F}} \mathbb{P}_I(\varphi) \quad \text{iff} \quad \text{for all } \mathcal{F}\text{-fair schedulers } D: \\ \Pr^D(s, \varphi) \in I$$

simple cases: e.g., if \mathcal{F} is realizable then

$$s \models_{\mathcal{F}} \mathbb{P}_{\leq p}(\diamond b) \quad \text{iff} \quad s \models \mathbb{P}_{\leq p}(\diamond b)$$

but $s \models_{\mathcal{F}} \mathbb{P}_{\geq p}(\diamond b) \quad \text{iff} \quad s \not\models \mathbb{P}_{\geq p}(\diamond b)$ is possible

- syntax of state and path formulas as before
- semantics as for standard **PCTL*** over MDP, but:

$$s \models_{\mathcal{F}} \mathbb{P}_I(\varphi) \quad \text{iff} \quad \text{for all } \mathcal{F}\text{-fair schedulers } D: \\ \Pr^D(s, \varphi) \in I$$

simple cases: e.g., if \mathcal{F} is realizable then

$$s \models_{\mathcal{F}} \mathbb{P}_{\leq p}(\diamond b) \quad \text{iff} \quad s \models \mathbb{P}_{\leq p}(\diamond b)$$

but $s \models_{\mathcal{F}} \mathbb{P}_{\geq p}(\diamond b) \quad \text{iff} \quad s \not\models \mathbb{P}_{\geq p}(\diamond b)$ is possible

$$s \models_{\mathcal{F}} \mathbb{P}_{\geq p}(\diamond b) \quad \text{iff} \\ s \models \mathbb{P}_{\leq 1-p}(\neg b \mathbf{U} \text{FairMEC}(\neg b))$$

given: MDP $\mathcal{M} = (S, Act, P, AP, L, s_0)$

fairness assumption \mathcal{F}

PCTL* state formula ϕ

task: check whether $\mathcal{M} \models_{\mathcal{F}} \phi$

given: MDP $\mathcal{M} = (\mathcal{S}, Act, P, AP, L, s_0)$

fairness assumption \mathcal{F}

PCTL* state formula ϕ

task: check whether $\mathcal{M} \models_{\mathcal{F}} \phi$

main procedure as for standard **PCTL***:

recursively compute the satisfaction sets

$$Sat_{\mathcal{F}}(\Psi) = \{s \in \mathcal{S} : s \models_{\mathcal{F}} \Psi\}$$

for all sub-state formulas Ψ of ϕ

given: MDP $\mathcal{M} = (\mathcal{S}, Act, P, AP, L, s_0)$

fairness assumption \mathcal{F}

PCTL* state formula ϕ

task: check whether $\mathcal{M} \models_{\mathcal{F}} \phi$

main procedure as for standard **PCTL***:

recursively compute the satisfaction sets

$$Sat_{\mathcal{F}}(\Psi) = \{s \in \mathcal{S} : s \models_{\mathcal{F}} \Psi\}$$

for all sub-state formulas Ψ of ϕ

treatment of the propositional logic fragment: ✓

- given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ with
realizable fairness assumption \mathcal{F}
PCTL* star formula $\mathbb{P}_{\leq p}(\varphi)$
- task: compute $\text{Sat}_{\mathcal{F}}(\mathbb{P}_{\leq p}(\varphi))$

given: MDP $\mathcal{M} = (\mathcal{S}, \text{Act}, P, \dots)$ with
realizable fairness assumption \mathcal{F}
PCTL* star formula $\mathbb{P}_{\leq p}(\varphi)$

task: compute $\text{Sat}_{\mathcal{F}}(\mathbb{P}_{\leq p}(\varphi))$

method: compute

$$x_s = \max_{D \text{ is fair}} \text{Pr}^D(s, \varphi)$$

via reduction to the probabilistic reachability problem

↑
using **DRA** \mathcal{A} for φ and
linear program for $\mathcal{M} \times \mathcal{A}$

MDP \mathcal{M} with
fairness \mathcal{F}

PCTL* path formula φ

MDP \mathcal{M} with
fairness \mathcal{F}

PCTL* path formula φ



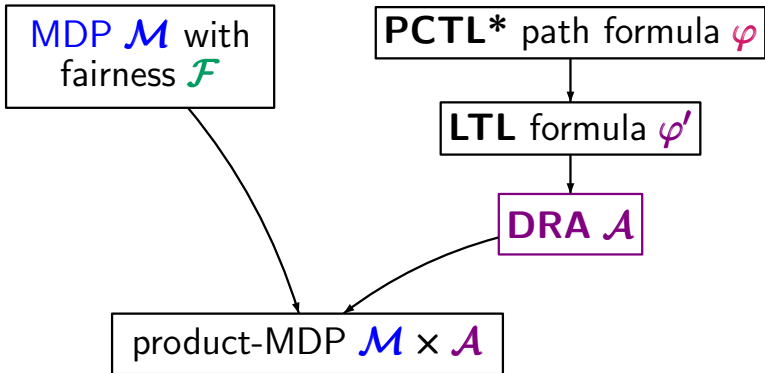
LTL formula φ'

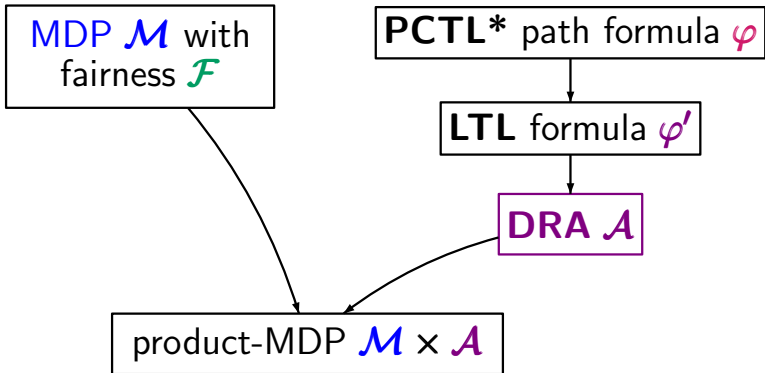
MDP \mathcal{M} with
fairness \mathcal{F}

PCTL* path formula φ

LTL formula φ'

DRA \mathcal{A}

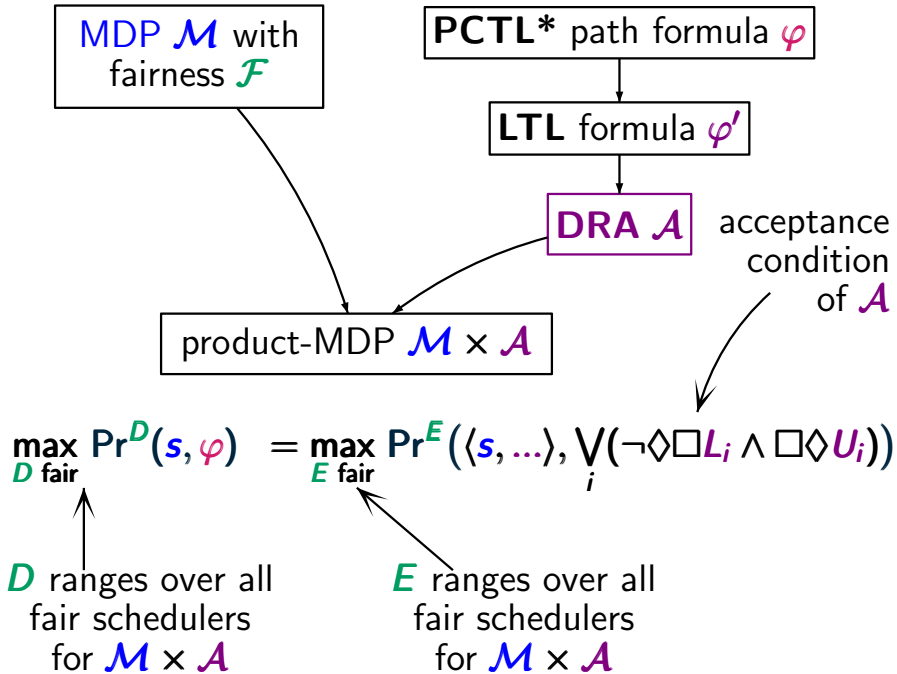


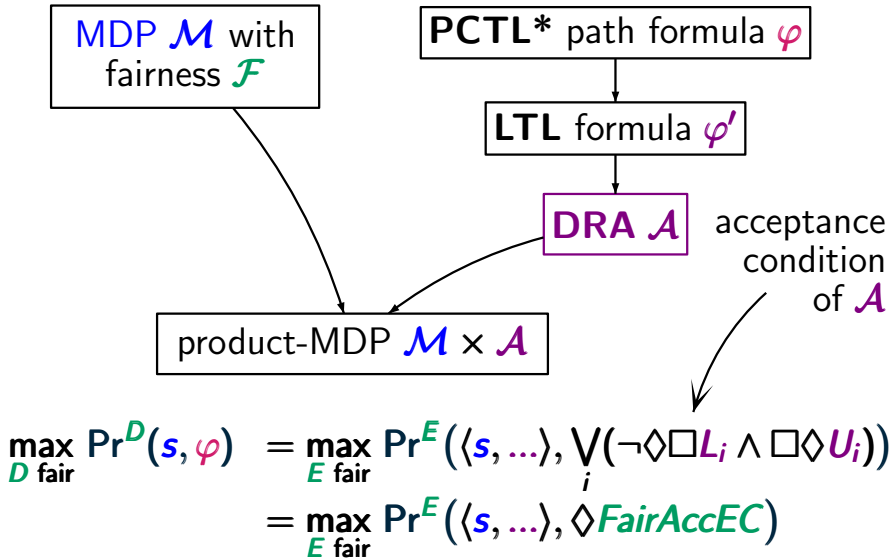


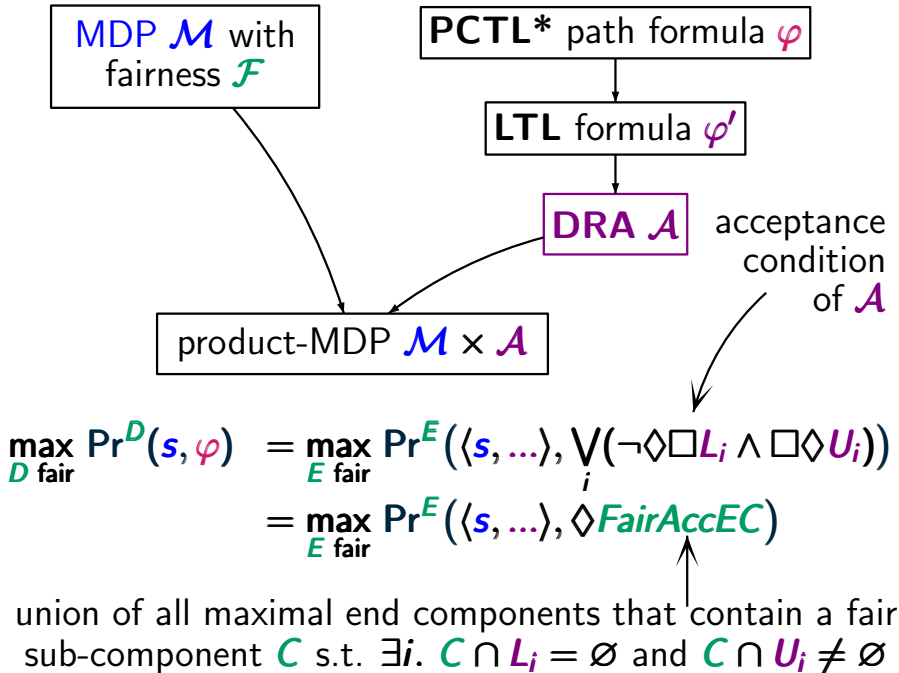
$$\max_{D \text{ fair}} \Pr^D(s, \varphi) = \max_{E \text{ fair}} \Pr^E(\langle s, \dots \rangle, \bigvee_i (\neg \diamond \square L_i \wedge \square \diamond U_i))$$

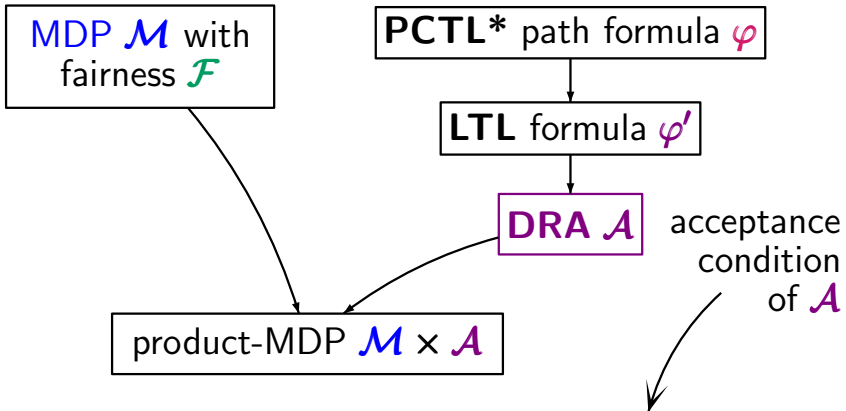
D ranges over all fair schedulers for $\mathcal{M} \times \mathcal{A}$

E ranges over all fair schedulers for $\mathcal{M} \times \mathcal{A}$









$$\begin{aligned}
 \max_{D \text{ fair}} \Pr^D(s, \varphi) &= \max_{E \text{ fair}} \Pr^E(\langle s, \dots \rangle, \bigvee_i (\neg \diamond \square L_i \wedge \square \diamond U_i)) \\
 &= \max_{E \text{ fair}} \Pr^E(\langle s, \dots \rangle, \diamond \text{FairAccEC}) \\
 &= \Pr_{\max}^{\mathcal{M} \times \mathcal{A}}(\langle s, \dots \rangle, \diamond \text{FairAccEC})
 \end{aligned}$$

- model checking for **systems with discrete probabilities**
 - * techniques for verifying **non-probabilistic systems** (graph algorithms, automata, ...)
 - * numerical methods for solving
 - linear equation systems** (Markov chains)
 - linear programs** (MDP)

- model checking for **systems with discrete probabilities**
 - * techniques for verifying **non-probabilistic systems** (graph algorithms, automata, ...)
 - * numerical methods for solving
 - linear equation systems** (Markov chains)
 - linear programs** (MDP)
- to combat the state explosion problem
 - * symbolic MTBDD-based **PRISM** [Kwiatk. et al]
 - * partial order reduction **LiQuor** [Baier et al]
 - * abstraction, bisimulation **MRMC** [Katoen et al]
 - refinement **RAPTURE** [d'Argenio et al]
 - ⋮ **PASS** [Hermanns et al]

Probabilistic Model Checking

Christel Baier

Technical University Dresden

Joost-Pieter Katoen

RWTH Aachen

David Parker

University of Oxford

Model Checking Continuous-Time Markov Chains

Joost-Pieter Katoen
RWTH Aachen