Symblicit algorithms for optimal strategy synthesis in monotonic Markov decision processes

Aaron Bohy[1] Véronique Bruyère[1] Jean-François Raskin[2]

[1]Université de Mons [2]Université Libre de Bruxelles

Cassting meeting
May 2014

Overview (1/2)

Motivations:

- Markov decision processes with large state spaces
- Explicit enumeration exhausts the memory
- Symbolic representations like MTBDDs are useful
- No easy use of (MT)BDDs for solving linear systems

[1] R. Wimmer, B. Braitling, B. Becker, E. M. Hahn, P. Crouzen, H. Hermanns, A. Dhama, and O. E. Theel. Symblicit calculation of long-run averages for concurrent probabilistic systems. In *QEST*, pages 27-36. IEEE Computer Society, 2010.

Overview (1/2)

Motivations:

- Markov decision processes with large state spaces
- Explicit enumeration exhausts the memory
- Symbolic representations like MTBDDs are useful
- No easy use of (MT)BDDs for solving linear systems

Recent contributions of [WBB+10][1]:

- **Symblicit** algorithm
  - Mixes **symb**olic and exp**licit** data structures
- Expected mean-payoff in Markov decision processes
- Using (MT)BDDs

[1] R. Wimmer, B. Braitling, B. Becker, E. M. Hahn, P. Crouzen, H. Hermanns, A. Dhama, and O. E. Theel. Symblicit calculation of long-run averages for concurrent probabilistic systems. In *QEST*, pages 27-36. IEEE Computer Society, 2010.

Overview (2/2)

Our contributions:

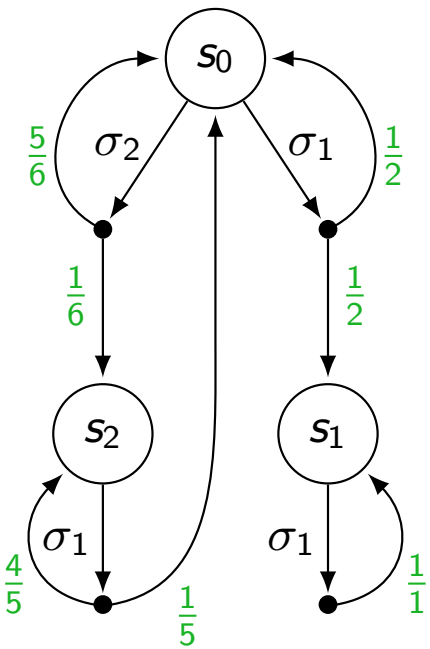
- New structure of **pseudo-antichain** (extension of antichains)
  $\Rightarrow$ Pseudo-antichain based symblicit algorithms
- **Monotonic** Markov decision processes
- **Two quantitative settings**:
  - Expected mean-payoff
  - Stochastic shortest path (focus of this talk)
- **Two applications**:
  - Automated planning
  - LTL synthesis

Full paper available on ArXiv: abs/1402.1076

Table of contents

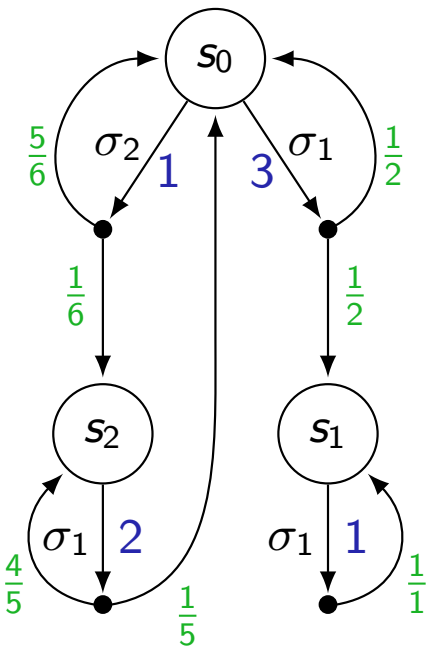
Table of contents

Markov decision processes (MDPs)



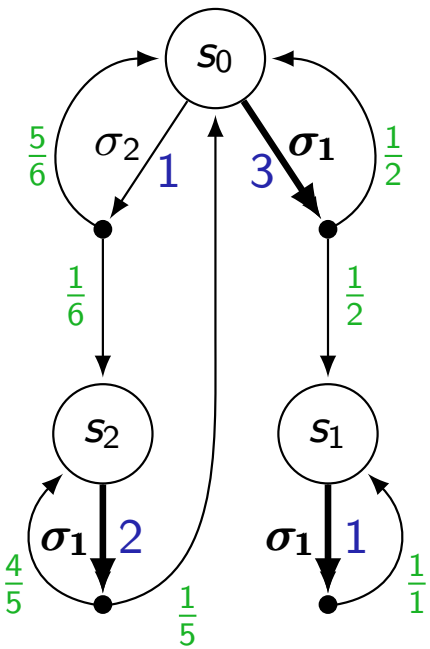
- $M = (S, \Sigma, \mathcal{P})$ where:
  - S is a finite set of *states*
  - Σ is a finite set of *actions*
  - $\mathcal{P} : S \times \Sigma \to \mathit{Dist}(S)$ is a *stochastic transition function*

Markov decision processes (MDPs)



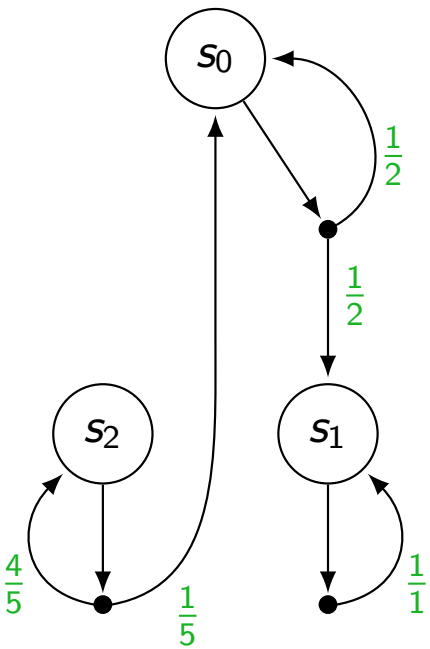
- $M = (S, \Sigma, \mathcal{P})$ where:
  - S is a finite set of *states*
  - Σ is a finite set of *actions*
  - $\mathcal{P} : S \times \Sigma \to \mathit{Dist}(S)$ is a *stochastic transition function*
- Cost function $c : S \times \Sigma \to \mathbb{R}_{>0}$

Markov decision processes (MDPs)



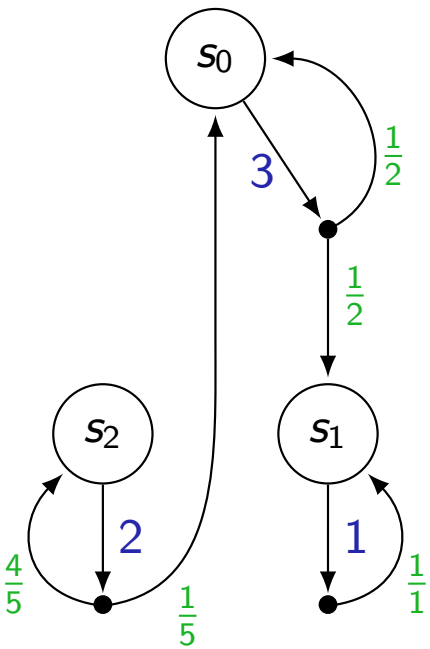
- $M = (S, \Sigma, \mathcal{P})$ where:
  - S is a finite set of *states*
  - Σ is a finite set of *actions*
  - $\mathcal{P} : S \times \Sigma \to Dist(S)$ is a *stochastic transition function*
- Cost function $c : S \times \Sigma \to \mathbb{R}_{>0}$
- (Memoryless) strategy $\lambda : S \to \Sigma$

Markov chains (MCs)



- MDP $(S, \Sigma, \mathcal{P})$ with $\mathcal{P} : S \times \Sigma \to \mathit{Dist}(S)$
  $+$ strategy $\lambda : S \to \Sigma$
  $\Rightarrow$ *induced* MC $(S, \mathcal{P}_\lambda)$ with $\mathcal{P}_\lambda : S \to \mathit{Dist}(S)$

Markov chains (MCs)



- MDP $(S, \Sigma, \mathcal{P})$ with $\mathcal{P} : S \times \Sigma \to \mathit{Dist}(S)$

  $+$ strategy $\lambda : S \to \Sigma$

  $\Rightarrow$ *induced* MC $(S, \mathcal{P}_\lambda)$ with $\mathcal{P}_\lambda : S \to \mathit{Dist}(S)$

- Cost function $c : S \times \Sigma \to \mathbb{R}_{>0}$

  $+$ strategy $\lambda : S \to \Sigma$

  $\Rightarrow$ *induced* cost function $c_\lambda : S \to \mathbb{R}_{>0}$

Expected truncated sum

- Let $M_\lambda = (S, \mathcal{P}_\lambda)$ with cost function c_λ
- Let $G \subseteq S$ be a set of *goal states*

Expected truncated sum

- Let $M_\lambda = (S, \mathcal{P}_\lambda)$ with cost function c_λ
- Let $G \subseteq S$ be a set of *goal states*

- $\mathrm{TS}_G(\rho = s_0 s_1 s_2 \dots) = \sum_{i=0}^{n-1} c_\lambda(s_i)$, with n first index s.t. $s_n \in G$

Expected truncated sum

- Let $M_\lambda = (S, \mathcal{P}_\lambda)$ with cost function c_λ
- Let $G \subseteq S$ be a set of *goal states*

- $\mathrm{TS}_G(\rho = s_0 s_1 s_2 \dots) = \sum_{i=0}^{n-1} c_\lambda(s_i)$, with n first index s.t. $s_n \in G$

- $\mathbb{E}_\lambda^{\mathrm{TS}_G}(s) = \sum_\rho \mathcal{P}_\lambda(\rho)\mathrm{TS}_G(\rho)$, with $\rho = s_0 s_1 \dots s_n$ s.t.
  $s_0 = s, s_n \in G$ and $s_0, \dots, s_{n-1} \notin G$

Stochastic shortest path (SSP)

- Let $M = (S, \Sigma, \mathcal{P})$ with cost function c
- Let $G \subseteq S$ be a set of *goal states*
- λ^* is *optimal* if $\mathbb{E}_{\lambda^*}^{\mathsf{TS}_G}(s) = \inf_{\lambda \in \Lambda} \mathbb{E}_{\lambda}^{\mathsf{TS}_G}(s)$

Stochastic shortest path (SSP)

- Let $M = (S, \Sigma, \mathcal{P})$ with cost function c
- Let $G \subseteq S$ be a set of *goal states*
- λ^* is *optimal* if $\mathbb{E}^{\mathsf{TS}_G}_{\lambda^*}(s) = \inf_{\lambda \in \Lambda} \mathbb{E}^{\mathsf{TS}_G}_{\lambda}(s)$
- SSP problem: compute an optimal strategy λ^*
- Complexity and strategies [BT96]:
  - Polynomial time via linear programming
  - Memoryless optimal strategies exist

Table of contents

Ingredients

- Strategy iteration algorithm [How60, BT96]
  - Generates a sequence of **monotonically improving strategies**
  - 2 phases:
    - strategy evaluation by solving a linear system
    - strategy improvement at each state
  - Stops as soon as no more improvement can be made
  - Returns the optimal strategy along with its value function

Ingredients

- Strategy iteration algorithm [How60, BT96]
  - Generates a sequence of **monotonically improving strategies**
  - 2 phases:
    - strategy evaluation by solving a linear system
    - strategy improvement at each state
  - Stops as soon as no more improvement can be made
  - Returns the optimal strategy along with its value function
- Bisimulation lumping [LS91, Buc94, KS60]
  - Applies to MCs
  - Gathers states which behave equivalently
  - Produces a **bisimulation quotient** (hopefully) smaller
  - Interested in the *largest* bisimulation $\sim_L$

Symblicit algorithm

- Mix of symbolic and explicit data structures

Algo 1 Symblicit(MDP $M^{\mathcal{S}}$, Cost function $c^{\mathcal{S}}$, Goal states $G^{\mathcal{S}}$)

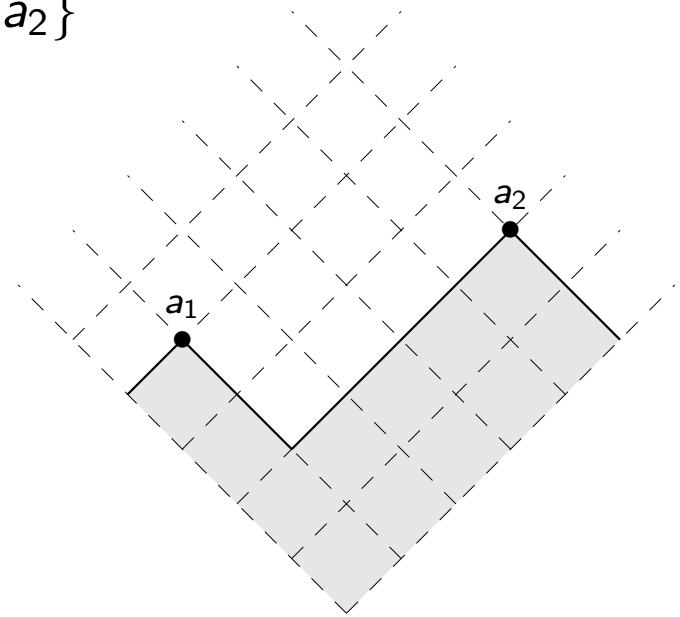
$n := 0, \lambda_n^{\mathcal{S}} :=$ InitialStrategy$(M^{\mathcal{S}}, G^{\mathcal{S}})$
repeat
  $(M_{\lambda_n}^{\mathcal{S}}, c_{\lambda_n}^{\mathcal{S}}) :=$ InducedMCAndCost$(M^{\mathcal{S}}, c^{\mathcal{S}}, \lambda_n^{\mathcal{S}})$
  $(M_{\lambda_n,\sim_L}^{\mathcal{S}}, c_{\lambda_n,\sim_L}^{\mathcal{S}}) :=$ Lump$(M_{\lambda_n}^{\mathcal{S}}, c_{\lambda_n}^{\mathcal{S}})$
  $(M_{\lambda_n,\sim_L}, c_{\lambda_n,\sim_L}) :=$ Explicit$(M_{\lambda_n,\sim_L}^{\mathcal{S}}, c_{\lambda_n,\sim_L}^{\mathcal{S}})$
  $v_n :=$ SolveLinearSystem$(M_{\lambda_n,\sim_L}, c_{\lambda_n,\sim_L})$
  $v_n^{\mathcal{S}} :=$ Symbolic(v_n)
  $\lambda_{n+1}^{\mathcal{S}} :=$ ImproveStrategy$(M^{\mathcal{S}}, \lambda_n^{\mathcal{S}}, v_n^{\mathcal{S}})$
  $n := n + 1$
until $\lambda_n^{\mathcal{S}} = \lambda_{n-1}^{\mathcal{S}}$
return $(\lambda_{n-1}^{\mathcal{S}}, v_{n-1}^{\mathcal{S}})$

Key: $\mathcal{S}$ in superscript denotes symbolic representations

Table of contents

Antichains

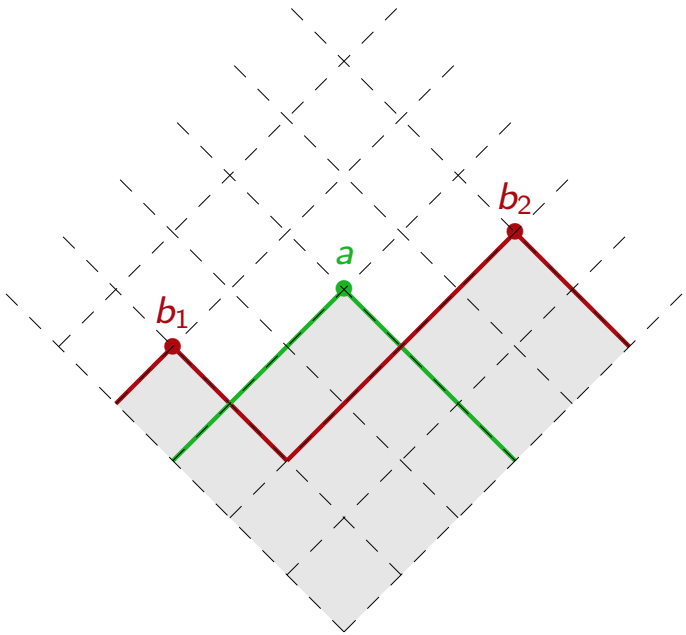
- Let $(S, \preceq)$ be a semilattice with greatest lower bound $\sqcap$
- A set $\alpha \subseteq S$ is an **antichain** if $\forall s, s' \in \alpha$, $s \not\preceq s'$ and $s' \not\preceq s$
- The *closure* of α is $\downarrow\alpha = \{s \in S \mid \exists a \in \alpha, s \preceq a\}$
- Example: $\alpha = \{a_1, a_2\}$

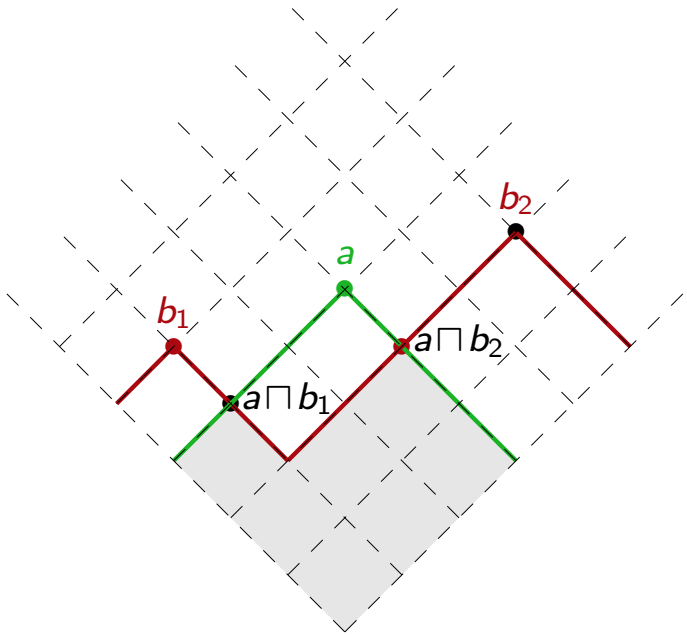


- **Canonical representations** of **closed sets** L by their maximal elements $\lceil L \rceil$ (*unique*)

Operations on antichains

- Given two antichains $\alpha, \beta \subseteq S$
- $\downarrow\alpha \cup \downarrow\beta$ and $\downarrow\alpha \cap \downarrow\beta$ are **closed**:

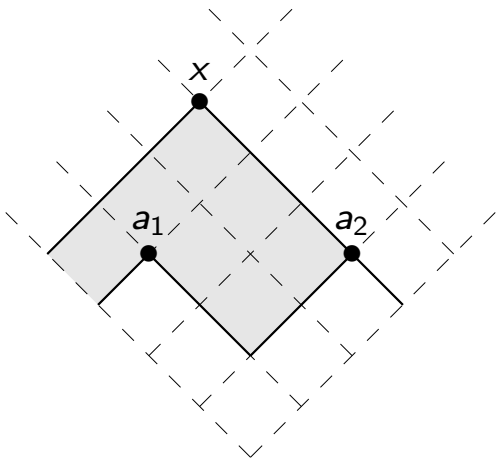




- But $\downarrow\alpha \setminus \downarrow\beta$ is **not** necessarily **closed**

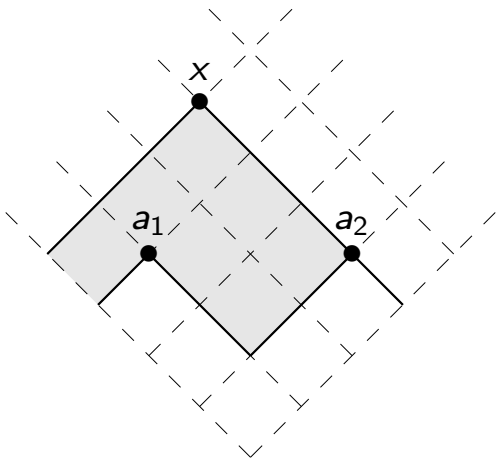
Pseudo-elements

- Let $(S, \preceq)$ be a semilattice with greatest lower bound $\sqcap$
- A **pseudo-element** is a pair (x, α) where $x \in S$ and $\alpha \subseteq S$ is an antichain such that $x \notin \downarrow\alpha$
- The *pseudo-closure* of (x, α) is $\updownarrow(x, \alpha) = \{s \in S \mid s \preceq x \text{ and } s \notin \downarrow\alpha\}$
  $= \downarrow\{x\} \setminus \downarrow\alpha$
- Example: (x, α) with $\alpha = \{a_1, a_2\}$



Pseudo-elements

- Let $(S, \preceq)$ be a semilattice with greatest lower bound $\sqcap$
- A **pseudo-element** is a pair (x, α) where $x \in S$ and $\alpha \subseteq S$ is an antichain such that $x \notin \downarrow\alpha$
- The *pseudo-closure* of (x, α) is $\updownarrow(x, \alpha) = \{s \in S \mid s \preceq x \text{ and } s \notin \downarrow\alpha\}$
  $= \downarrow\{x\} \setminus \downarrow\alpha$
- Example: (x, α) with $\alpha = \{a_1, a_2\}$



- (x, α) is in **canonical form** if $\forall a \in \alpha, a \preceq x$ (*unique*)

Pseudo-antichains

- A **pseudo-antichain** A is a set $\{(x_i, \alpha_i) \mid i \in I\}$ of pseudo-elements
- The *pseudo-closure* of A is $\updownarrow A = \bigcup_{i \in I} \updownarrow(x_i, \alpha_i)$

Pseudo-antichains

- A **pseudo-antichain** A is a set $\{(x_i, \alpha_i) \mid i \in I\}$ of pseudo-elements
- The *pseudo-closure* of A is $\updownarrow A = \bigcup_{i \in I} \updownarrow (x_i, \alpha_i)$
- A is a *PA-representation* of $\updownarrow A$ (*not unique*)
- A is **simplified** if:
  - $\forall i \in I, (x_i, \alpha_i)$ is in canonical form, and
  - A does not contain unnecessary pseudo-elements

Pseudo-antichains

- A **pseudo-antichain** A is a set $\{(x_i, \alpha_i) \mid i \in I\}$ of pseudo-elements
- The *pseudo-closure* of A is $\updownarrow A = \bigcup_{i \in I} \updownarrow (x_i, \alpha_i)$
- A is a *PA-representation* of $\updownarrow A$ (*not unique*)
- A is **simplified** if:
  - $\forall i \in I, (x_i, \alpha_i)$ is in canonical form, and
  - A does not contain unnecessary pseudo-elements
- Remark: **any** set can be PA-represented

Operations

- Efficient computations of pseudo-closures of pseudo-antichains w.r.t. the union, intersection and difference
- Example (for the intersection):
  $\updownarrow(x, \{a\}) \cap \updownarrow(y, \{b\}) = \updownarrow(x \sqcap y, \{a, b\})$

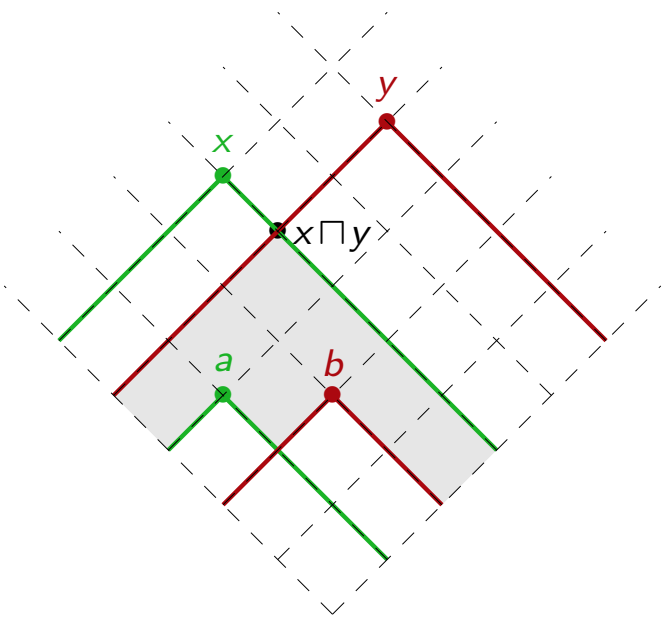


Table of contents

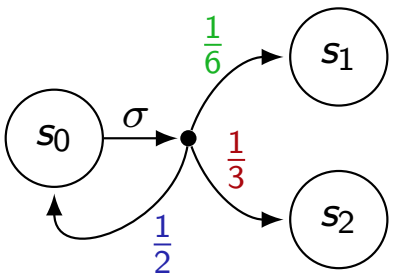
New definition of MDPs

- Not.: $\mathcal{F}_{\text{tot}}(X, Y)$ is the set of total functions from X to Y

New definition of MDPs

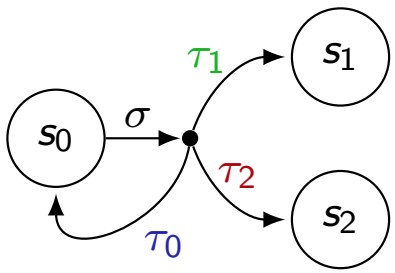
- Not.: $\mathcal{F}_{\text{tot}}(X, Y)$ is the set of total functions from X to Y
- MDP $M = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ where:
  - T set of stochastic actions s.t. $\Sigma \cap T = \emptyset$
  - $\mathcal{E} : S \times \Sigma \to \mathcal{F}_{\text{tot}}(T, S)$ **successor function**
  - $\mathcal{D} : S \times \Sigma \to \mathit{Dist}(T)$ **stochastic function**

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



$\mathcal{P}(s_0, \sigma)(s_0) = \frac{1}{2}$
$\mathcal{P}(s_0, \sigma)(s_1) = \frac{1}{6}$
$\mathcal{P}(s_0, \sigma)(s_2) = \frac{1}{3}$

$M = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$



$\mathcal{E}(s_0, \sigma)(\tau_0) = s_0$
$\mathcal{E}(s_0, \sigma)(\tau_1) = s_1$
$\mathcal{E}(s_0, \sigma)(\tau_2) = s_2$

$\mathcal{D}(s_0, \sigma)(\tau_0) = \frac{1}{2}$
$\mathcal{D}(s_0, \sigma)(\tau_1) = \frac{1}{6}$
$\mathcal{D}(s_0, \sigma)(\tau_2) = \frac{1}{3}$

Monotonic MDPs

- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice

Monotonic MDPs

- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
  - $\preceq$ is *compatible* with $\mathcal{E}$, i.e. $\forall s, s' \in S$

s s'

Monotonic MDPs

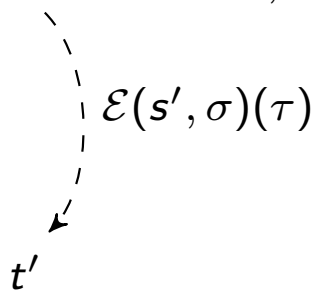
- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
  - $\preceq$ is *compatible* with $\mathcal{E}$, i.e. $\forall s, s' \in S$

$$s \qquad \preceq \qquad s'$$

Monotonic MDPs

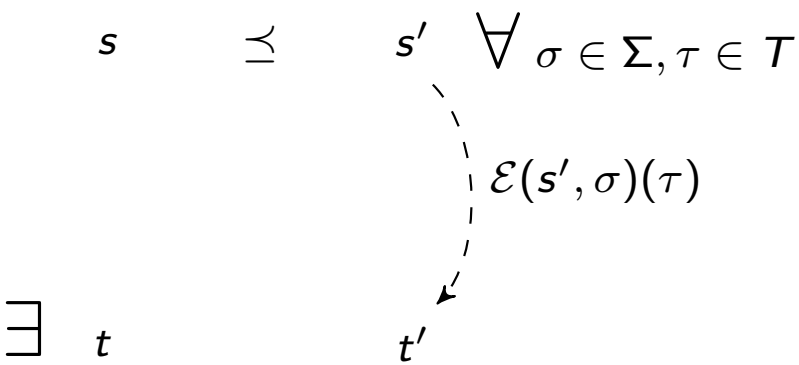
- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
  - $\preceq$ is *compatible* with $\mathcal{E}$, i.e. $\forall s, s' \in S$

$$s \preceq s' \quad \forall\, \sigma \in \Sigma, \tau \in T$$



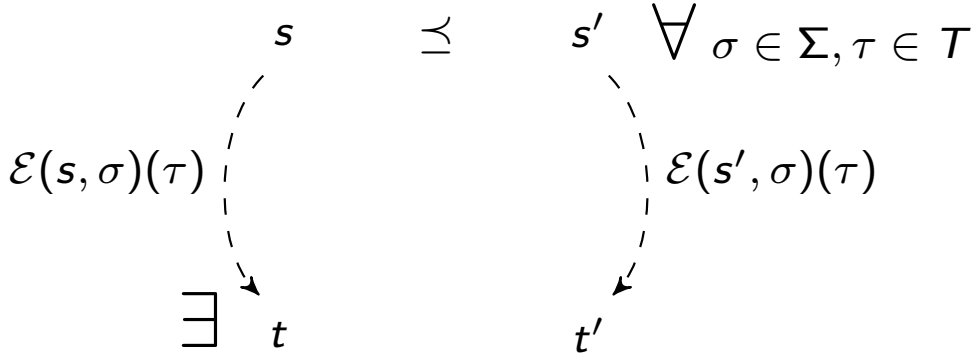
Monotonic MDPs

- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
  - $\preceq$ is *compatible* with $\mathcal{E}$, i.e. $\forall s, s' \in S$



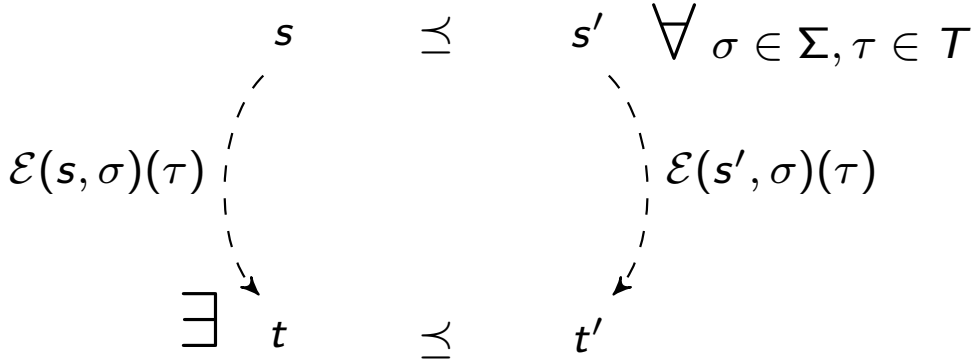
Monotonic MDPs

- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
  - $\preceq$ is *compatible* with $\mathcal{E}$, i.e. $\forall s, s' \in S$



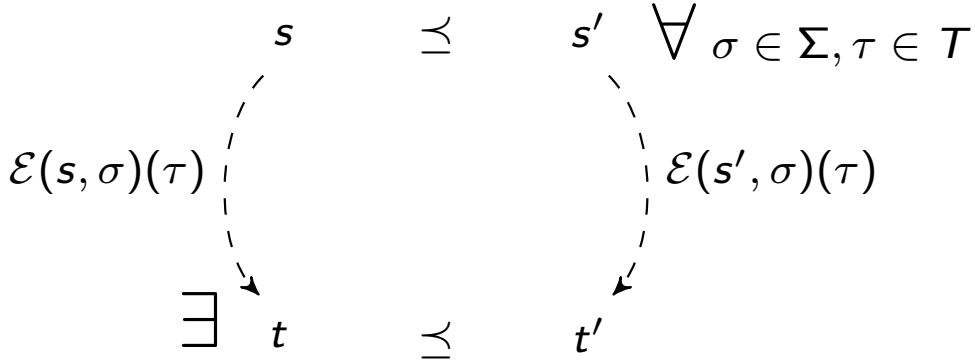
Monotonic MDPs

- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
  - $\preceq$ is *compatible* with $\mathcal{E}$, i.e. $\forall s, s' \in S$



Monotonic MDPs

- A *monotonic* MDP is an MDP $M_{\preceq} = (S, \Sigma, T, \mathcal{E}, \mathcal{D})$ s.t.:
  - S is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
  - $\preceq$ is *compatible* with $\mathcal{E}$, i.e. $\forall s, s' \in S$

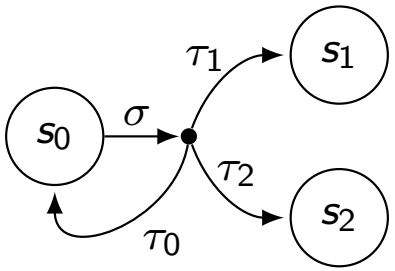


- Remark:
  - All MDPs can be seen monotonic
  - Interested in MDPs built on state spaces *already* equipped with a **natural partial order**

$\mathrm{Pre}_{\sigma,\tau}$ operator

Given $L \subseteq S, \sigma \in \Sigma$ and $\tau \in T$, let

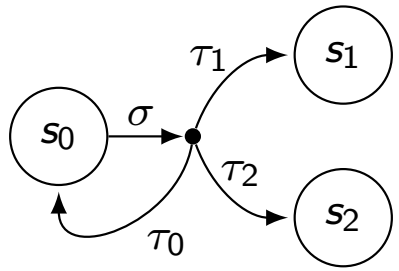
$$\mathrm{Pre}_{\sigma,\tau}(L) = \{s \in S \mid \mathcal{E}(s,\sigma)(\tau) \in L\}$$



$\mathrm{Pre}_{\sigma,\tau}$ operator

Given $L \subseteq S, \sigma \in \Sigma$ and $\tau \in T$, let

$$\mathrm{Pre}_{\sigma,\tau}(L) = \{s \in S \mid \mathcal{E}(s,\sigma)(\tau) \in L\}$$

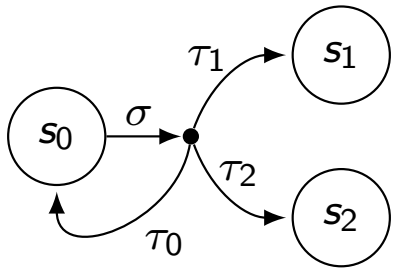


Lemma. For all closed set $L \subseteq S$, $\mathrm{Pre}_{\sigma,\tau}(L)$ is closed

$\mathrm{Pre}_{\sigma,\tau}$ operator

Given $L \subseteq S, \sigma \in \Sigma$ and $\tau \in T$, let

$$\mathrm{Pre}_{\sigma,\tau}(L) = \{s \in S \mid \mathcal{E}(s,\sigma)(\tau) \in L\}$$



Lemma. For all closed set $L \subseteq S$, $\mathrm{Pre}_{\sigma,\tau}(L)$ is closed

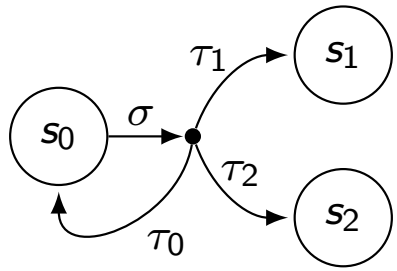
Proposition. Let $A = \{(x_i, \alpha_i) \mid i \in I\}$ be a pseudo-antichain. Then,

- $\mathrm{Pre}_{\sigma,\tau}(\updownarrow A) = \bigcup_{i \in I} \mathrm{Pre}_{\sigma,\tau}(\updownarrow(x_i, \alpha_i))$

$\mathrm{Pre}_{\sigma,\tau}$ operator

Given $L \subseteq S, \sigma \in \Sigma$ and $\tau \in T$, let

$$\mathrm{Pre}_{\sigma,\tau}(L) = \{s \in S \mid \mathcal{E}(s,\sigma)(\tau) \in L\}$$



Lemma. For all closed set $L \subseteq S$, $\mathrm{Pre}_{\sigma,\tau}(L)$ is closed

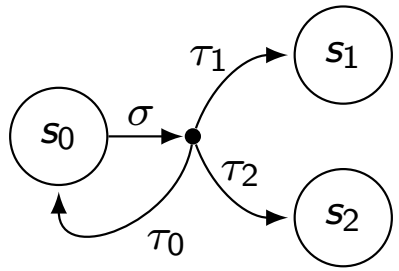
Proposition. Let $A = \{(x_i, \alpha_i) \mid i \in I\}$ be a pseudo-antichain. Then,

- $\mathrm{Pre}_{\sigma,\tau}(\updownarrow A) = \bigcup_{i \in I} \mathrm{Pre}_{\sigma,\tau}(\updownarrow(x_i, \alpha_i))$
- $\mathrm{Pre}_{\sigma,\tau}(\updownarrow(x, \alpha)) = \bigcup_{x' \in \lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil} \updownarrow(x', \lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\alpha) \rceil)$

$\mathrm{Pre}_{\sigma,\tau}$ operator

Given $L \subseteq S, \sigma \in \Sigma$ and $\tau \in T$, let

$$\mathrm{Pre}_{\sigma,\tau}(L) = \{s \in S \mid \mathcal{E}(s,\sigma)(\tau) \in L\}$$



Lemma. For all closed set $L \subseteq S$, $\mathrm{Pre}_{\sigma,\tau}(L)$ is closed

Proposition. Let $A = \{(x_i, \alpha_i) \mid i \in I\}$ be a pseudo-antichain. Then,

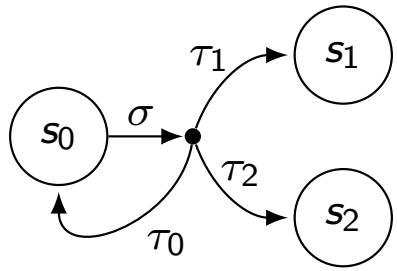
- $\mathrm{Pre}_{\sigma,\tau}(\updownarrow A) = \bigcup_{i \in I} \mathrm{Pre}_{\sigma,\tau}(\updownarrow(x_i, \alpha_i))$
- $\mathrm{Pre}_{\sigma,\tau}(\updownarrow(x, \alpha)) = \bigcup_{x' \in \lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil} \updownarrow(x', \lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\alpha) \rceil)$

Assumption 1. There exists an algorithm taking any state $x \in S$ in input and returning $\lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil$

$\mathrm{Pre}_{\sigma,\tau}$ operator

Given $L \subseteq S, \sigma \in \Sigma$ and $\tau \in T$, let

$$\mathrm{Pre}_{\sigma,\tau}(L) = \{s \in S \mid \mathcal{E}(s,\sigma)(\tau) \in L\}$$



Lemma. For all closed set $L \subseteq S$, $\mathrm{Pre}_{\sigma,\tau}(L)$ is closed

Proposition. Let $A = \{(x_i, \alpha_i) \mid i \in I\}$ be a pseudo-antichain. Then,

- $\mathrm{Pre}_{\sigma,\tau}(\updownarrow A) = \bigcup_{i \in I} \mathrm{Pre}_{\sigma,\tau}(\updownarrow(x_i, \alpha_i))$
- $\mathrm{Pre}_{\sigma,\tau}(\updownarrow(x, \alpha)) = \bigcup_{x' \in \lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil} \updownarrow(x', \lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\alpha) \rceil)$

Assumption 1. There exists an algorithm taking any state $x \in S$ in input and returning $\lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil$

$\Rightarrow$ The computation of $\mathrm{Pre}_{\sigma,\tau}(\updownarrow A)$ does not treat the whole $\updownarrow A$

Symbolic representations

- Representations of equivalence relations
  - Example: $s \sim_\lambda s'$ iff $\lambda(s) = \lambda(s')$
- Each equivalence class is PA-represented

Symbolic representations

- Representations of equivalence relations
  - Example: $s \sim_\lambda s'$ iff $\lambda(s) = \lambda(s')$
- Each equivalence class is PA-represented
- Coarsest equivalence relations
- PA-representations as *compact* as possible

Symbolic representations

- Representations of equivalence relations
  - Example: $s \sim_\lambda s'$ iff $\lambda(s) = \lambda(s')$
- Each equivalence class is PA-represented
- Coarsest equivalence relations
- PA-representations as *compact* as possible

Assumption 2. Given $s \in S$, $\sigma \in \Sigma$ and $\tau \in T$, $\mathcal{E}(s, \sigma)(\tau)$ and $\mathcal{D}(s, \sigma)(\tau)$ can be computed on-the-fly

Table of contents

STRIPS

A *STRIPS* is a tuple (P, I, M, O) where

- P is a finite set of *propositional variables*
- $I \subseteq P$ is a subset of *initial* variables
- $M \subseteq P$ is a subset of *goal* variables

STRIPS

A *STRIPS* is a tuple (P, I, M, O) where

- P is a finite set of *propositional variables*
- $I \subseteq P$ is a subset of *initial* variables
- $M \subseteq P$ is a subset of *goal* variables
- O is a finite set of *operators* $o = (\gamma, (\alpha, \delta))$ s.t.
  - $\gamma \subseteq P$ is the *guard* of o
  - (α, δ), with $\alpha, \delta \subseteq P$, is the *effect* of o

STRIPS

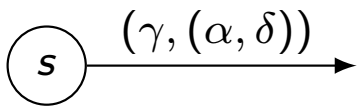
A *STRIPS* is a tuple (P, I, M, O) where

- P is a finite set of *propositional variables*
- $I \subseteq P$ is a subset of *initial* variables
- $M \subseteq P$ is a subset of *goal* variables
- O is a finite set of *operators* $o = (\gamma, (\alpha, \delta))$ s.t.
  - $\gamma \subseteq P$ is the *guard* of o
  - (α, δ), with $\alpha, \delta \subseteq P$, is the *effect* of o

STRIPS

A *STRIPS* is a tuple (P, I, M, O) where

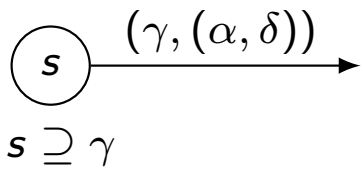
- P is a finite set of *propositional variables*
- $I \subseteq P$ is a subset of *initial* variables
- $M \subseteq P$ is a subset of *goal* variables
- O is a finite set of *operators* $o = (\gamma, (\alpha, \delta))$ s.t.
  - $\gamma \subseteq P$ is the *guard* of o
  - (α, δ), with $\alpha, \delta \subseteq P$, is the *effect* of o



STRIPS

A *STRIPS* is a tuple (P, I, M, O) where

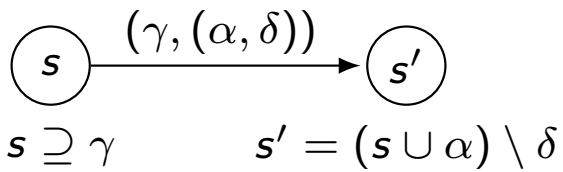
- P is a finite set of *propositional variables*
- $I \subseteq P$ is a subset of *initial* variables
- $M \subseteq P$ is a subset of *goal* variables
- O is a finite set of *operators* $o = (\gamma, (\alpha, \delta))$ s.t.
  - $\gamma \subseteq P$ is the *guard* of o
  - (α, δ), with $\alpha, \delta \subseteq P$, is the *effect* of o



STRIPS

A *STRIPS* is a tuple (P, I, M, O) where

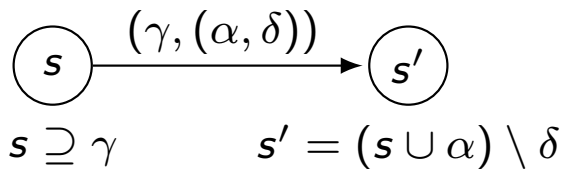
- P is a finite set of *propositional variables*
- $I \subseteq P$ is a subset of *initial* variables
- $M \subseteq P$ is a subset of *goal* variables
- O is a finite set of *operators* $o = (\gamma, (\alpha, \delta))$ s.t.
  - $\gamma \subseteq P$ is the *guard* of o
  - (α, δ), with $\alpha, \delta \subseteq P$, is the *effect* of o



STRIPS

A *STRIPS* is a tuple (P, I, M, O) where

- P is a finite set of *propositional variables*
- $I \subseteq P$ is a subset of *initial* variables
- $M \subseteq P$ is a subset of *goal* variables
- O is a finite set of *operators* $o = (\gamma, (\alpha, \delta))$ s.t.
  - $\gamma \subseteq P$ is the *guard* of o
  - (α, δ), with $\alpha, \delta \subseteq P$, is the *effect* of o



Planning from STRIPS [FN72]

- Find a sequence of operators leading from the initial state I to a goal state $s \supseteq M$

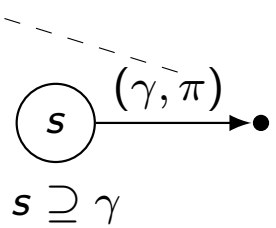
Stochastic STRIPS

- Extension of STRIPS with **stochastic aspects** [BL00]
- Operator $o = (\gamma, \pi) \in O$ s.t.
  - γ is the guard of o
  - $\pi : 2^P \times 2^P \to [0, 1]$ is a probability distribution on the pairs (α, δ)

Stochastic STRIPS

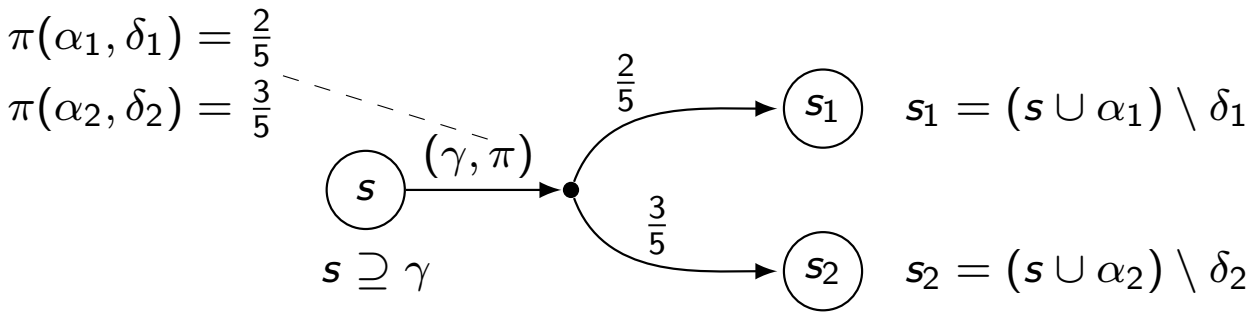
- Extension of STRIPS with **stochastic aspects** [BL00]
- Operator $o = (\gamma, \pi) \in O$ s.t.
  - γ is the guard of o
  - $\pi : 2^P \times 2^P \to [0, 1]$ is a probability distribution on the pairs (α, δ)

$\pi(\alpha_1, \delta_1) = \frac{2}{5}$
$\pi(\alpha_2, \delta_2) = \frac{3}{5}$



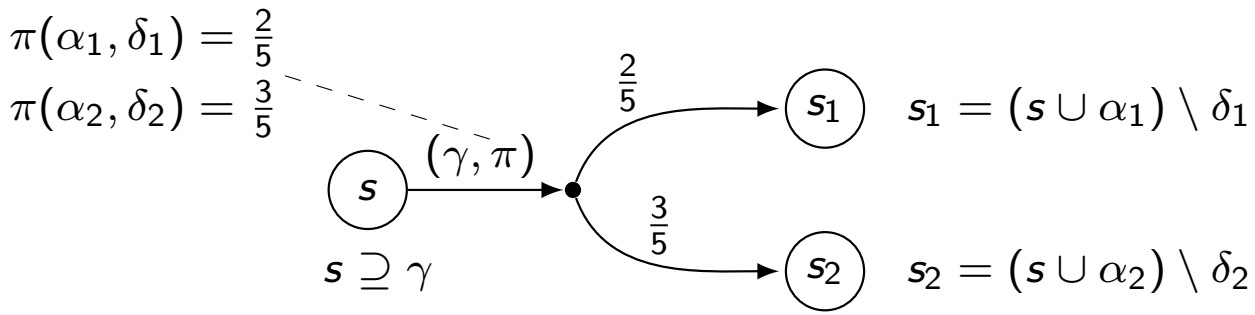
Stochastic STRIPS

- Extension of STRIPS with **stochastic aspects** [BL00]
- Operator $o = (\gamma, \pi) \in O$ s.t.
  - γ is the guard of o
  - $\pi : 2^P \times 2^P \to [0, 1]$ is a probability distribution on the pairs (α, δ)



Stochastic STRIPS

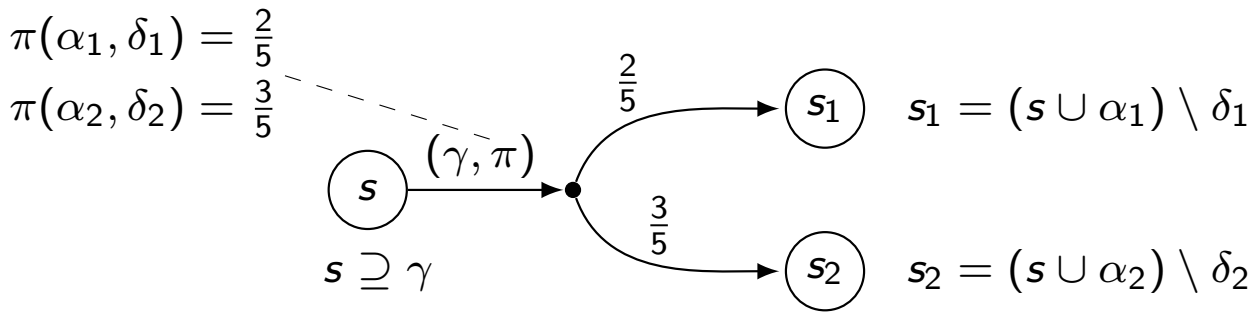
- Extension of STRIPS with **stochastic aspects** [BL00]
- Operator $o = (\gamma, \pi) \in O$ s.t.
  - γ is the guard of o
  - $\pi : 2^P \times 2^P \to [0, 1]$ is a probability distribution on the pairs (α, δ)



- $\rightsquigarrow$ Markov decision process

Stochastic STRIPS

- Extension of STRIPS with **stochastic aspects** [BL00]
- Operator $o = (\gamma, \pi) \in O$ s.t.
  - γ is the guard of o
  - $\pi : 2^P \times 2^P \to [0, 1]$ is a probability distribution on the pairs (α, δ)



- $\rightsquigarrow$ Markov decision process

- Cost function $C : O \to \mathbb{R}_{>0}$
- Planning from stochastic STRIPS
  - Minimize the truncated sum up to a state $s \supseteq M$ from I
  - $\rightsquigarrow$ Stochastic shortest path problem

Symblicit algorithm (1/2)

- M_S is **monotonic**
  - $(S, \supseteq)$ is a semilattice with greatest lower bound $\sqcap = \cup$
  - $\supseteq$ is compatible with $\mathcal{E}$

$$s \qquad \supseteq \qquad s'$$

Symblicit algorithm (1/2)

- M_S is **monotonic**
  - $(S, \supseteq)$ is a semilattice with greatest lower bound $\sqcap = \cup$
  - $\supseteq$ is compatible with $\mathcal{E}$

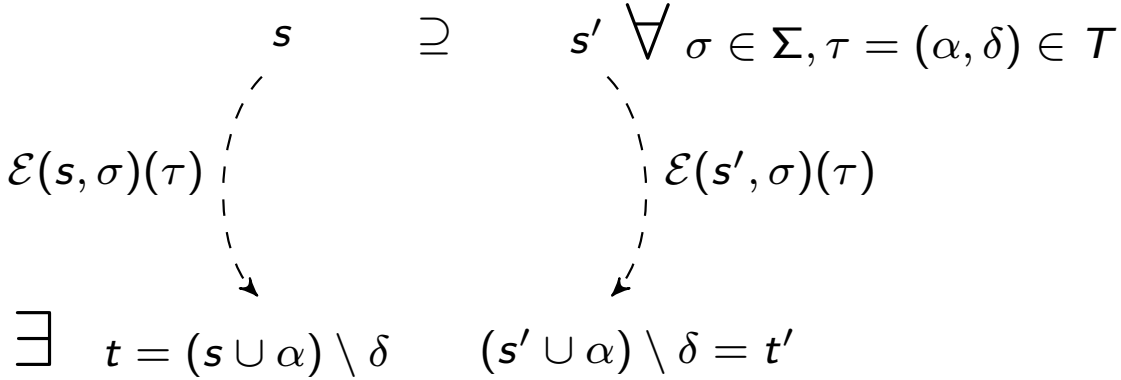
$$s \quad \supseteq \quad s' \quad \forall\, \sigma \in \Sigma, \tau = (\alpha, \delta) \in T$$

$$\mathcal{E}(s', \sigma)(\tau)$$

$$(s' \cup \alpha) \setminus \delta = t'$$

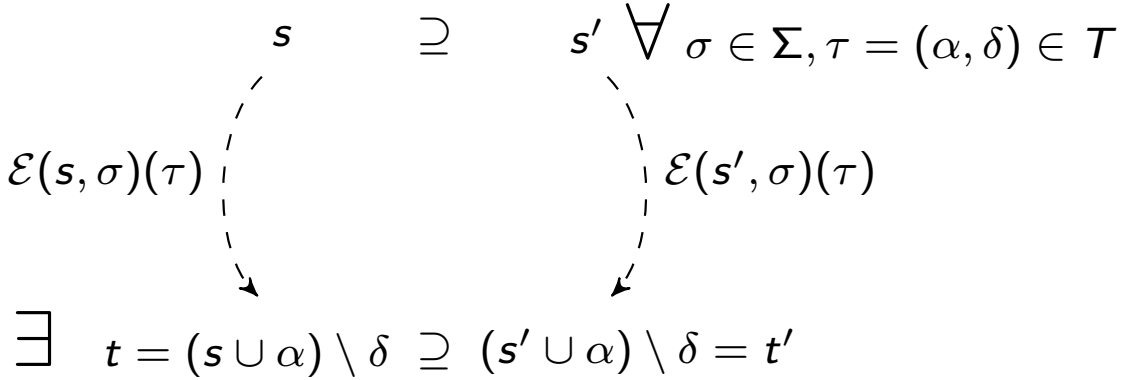
Symblicit algorithm (1/2)

- M_S is **monotonic**
  - $(S, \supseteq)$ is a semilattice with greatest lower bound $\sqcap = \cup$
  - $\supseteq$ is compatible with $\mathcal{E}$



Symblicit algorithm (1/2)

- M_S is **monotonic**
  - $(S, \supseteq)$ is a semilattice with greatest lower bound $\sqcap = \cup$
  - $\supseteq$ is compatible with $\mathcal{E}$

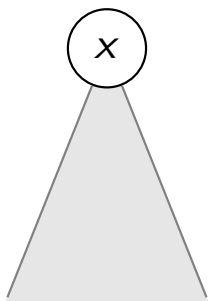


Symblicit algorithm (2/2)

- Assumption 1. There exists an algorithm taking any state $x \in S$ in input and returning $\lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil$:

Algo 2 $\mathrm{Pre}(x, \sigma = (\gamma, \pi), \tau = (\alpha, \delta))$

1: **if** $x \cap \delta \neq \emptyset$ **then**
2: return $\emptyset$
3: **else**
4: return $\{\gamma \cup (x \setminus \alpha)\}$

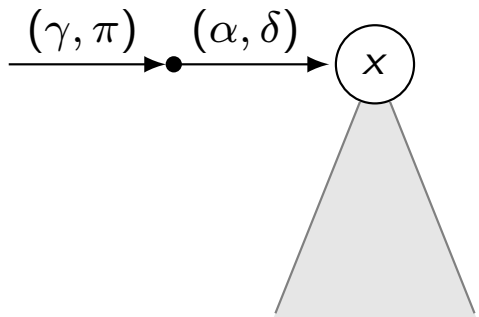


Symblicit algorithm (2/2)

- Assumption 1. There exists an algorithm taking any state $x \in S$ in input and returning $\lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil$:

Algo 2 $\mathrm{Pre}(x, \sigma = (\gamma, \pi), \tau = (\alpha, \delta))$

1: **if** $x \cap \delta \neq \emptyset$ **then**
2: return $\emptyset$
3: **else**
4: return $\{\gamma \cup (x \setminus \alpha)\}$

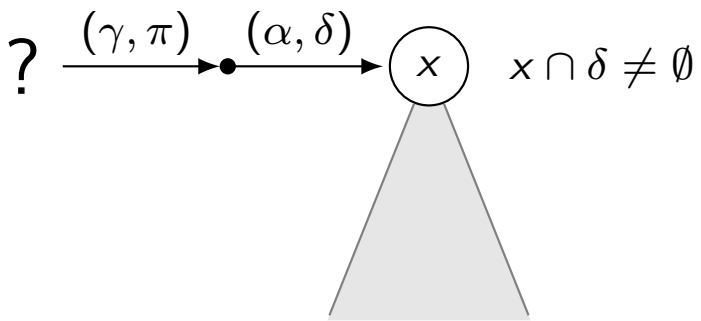


Symblicit algorithm (2/2)

- Assumption 1. There exists an algorithm taking any state $x \in S$ in input and returning $\lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil$:

Algo 2 $\mathrm{Pre}(x, \sigma = (\gamma, \pi), \tau = (\alpha, \delta))$

1: **if** $x \cap \delta \neq \emptyset$ **then**
2: return $\emptyset$
3: **else**
4: return $\{\gamma \cup (x \setminus \alpha)\}$

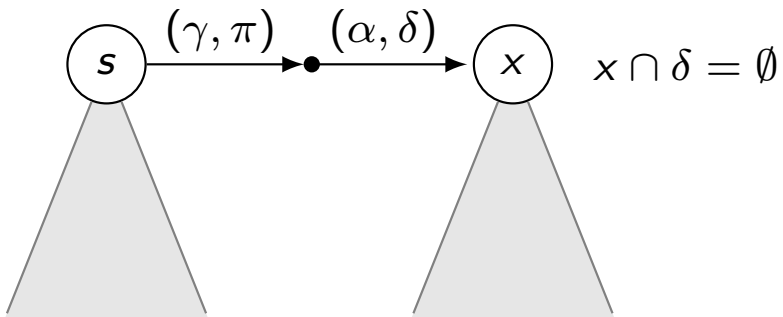


Symblicit algorithm (2/2)

- Assumption 1. There exists an algorithm taking any state $x \in S$ in input and returning $\lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil$:

Algo 2 $\mathrm{Pre}(x, \sigma = (\gamma, \pi), \tau = (\alpha, \delta))$

1: **if** $x \cap \delta \neq \emptyset$ **then**
2: return $\emptyset$
3: **else**
4: return $\{\gamma \cup (x \setminus \alpha)\}$

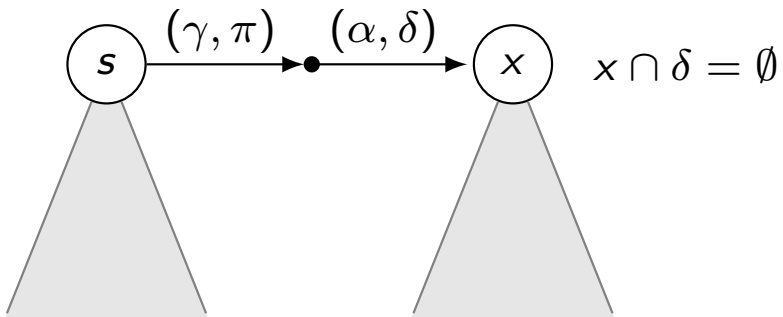


Symblicit algorithm (2/2)

- Assumption 1. There exists an algorithm taking any state $x \in S$ in input and returning $\lceil \mathrm{Pre}_{\sigma,\tau}(\downarrow\{x\}) \rceil$:

Algo 2 $\mathrm{Pre}(x, \sigma = (\gamma, \pi), \tau = (\alpha, \delta))$

1: **if** $x \cap \delta \neq \emptyset$ **then**
2: return $\emptyset$
3: **else**
4: return $\{\gamma \cup (x \setminus \alpha)\}$



- Assumption 2. Given $s \in S$, $\sigma = (\gamma, \pi) \in \Sigma$ and $\tau = (\alpha, \delta) \in T$, $\mathcal{E}(s, \sigma)(\tau)$ and $\mathcal{D}(s, \sigma)(\tau)$ can be computed on-the-fly:
  - $\mathcal{E}(s, \sigma)(\tau) = (s \cup \alpha) \setminus \delta$
  - $\mathcal{D}(s, \sigma)(\tau) = \pi(\tau)$

Experimental results

	example	$\mathbb{E}_\lambda^{\mathsf{TS}_G}$	$\lvert M_S \rvert$	PA #it	PA $\lvert \sim_L \rvert$	PA time	PA mem	Explicit time	Explicit mem
Monkey	(3, 2)	35.75	4096	4	23	0.2	16.0	60.6	1626
	(3, 3)	35.75	65536	5	43	1.6	17.3		> 4000
	(3, 4)	35.75	1048576	6	57	17.8	21.7		> 4000
	(3, 5)	36.00	16777216	7	88	272.1	37.5		> 4000
	(5, 2)	35.75	65536	4	31	0.5	16.6	20316.2	2343
	(5, 3)	35.75	4194304	5	56	8.2	19.5		> 4000
	(5, 4)	35.75	268435456	6	97	196.8	31.3		> 4000
	(5, 5)	36.00	17179869184	7	152	7098.4	81.3		> 4000
Moats and castles	(2, 5)	32.22	4096	3	49	1.8	17.3	133.7	1202
	(2, 6)	32.22	16384	3	66	11.7	19.3	2966.8	1706
	(3, 3)	59.00	4096	3	84	15.3	20.2	149.6	1205
	(3, 4)	52.00	32768	3	219	150.8	30.7	14660.7	1611
	(3, 5)	48.33	262144	3	357	740.2	49.1		> 4000
	(3, 6)	48.33	2097152	3	595	11597.7	145.8		> 4000
	(4, 2)	96.89	4096	3	132	43.7	26.5	173.6	1211
	(4, 3)	78.67	65536	3	464	1594.5	82.2		> 4000

Expected mean-payoff with LTL synthesis

Comparison with an MTBDD based symblicit algorithm [VE13]

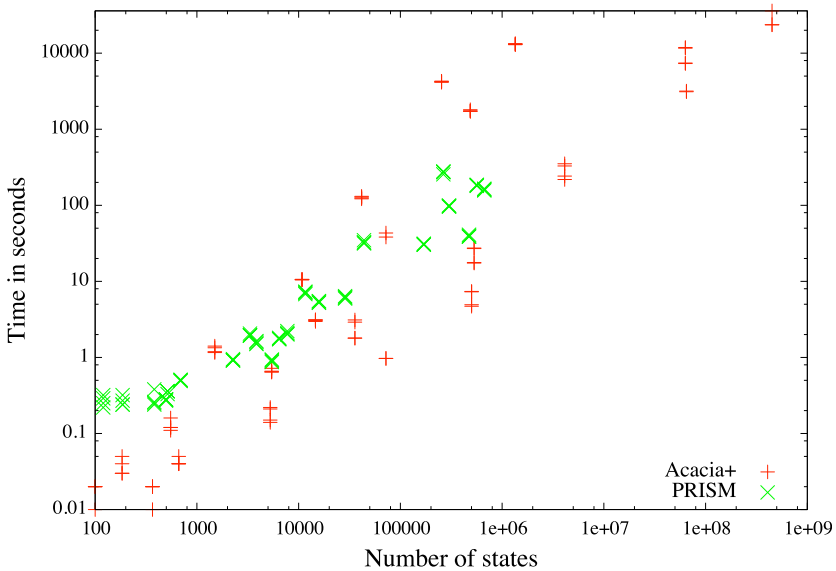


Figure : Execution time

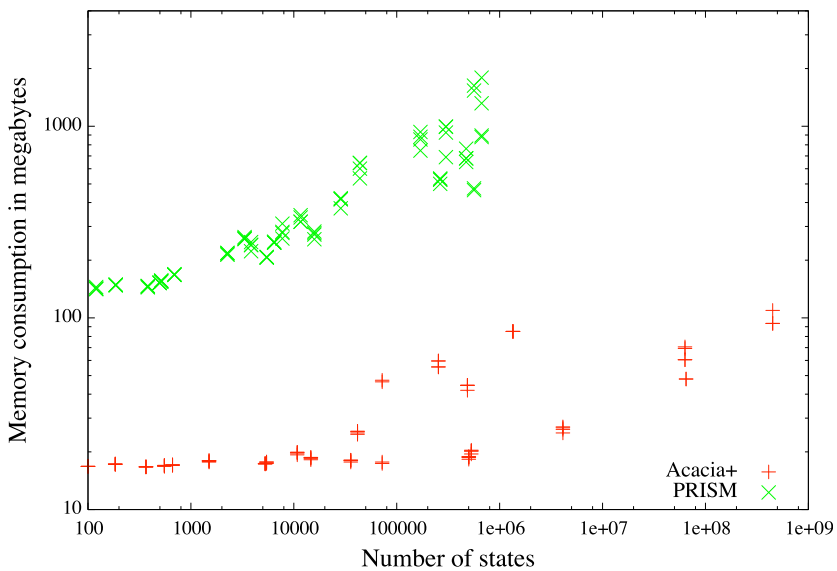


Figure : Memory consumption

$\Rightarrow$ Monotonic MDPs are better handled by pseudo-antichains

Table of contents

Conclusion and future work

Summary:

- New data structure of **pseudo-antichains**
- Symblicit algorithms in monotonic MDPs with a **natural partial order**
- Expected mean-payoff and stochastic shortest path
- Promising experimental results

Conclusion and future work

Summary:

- New data structure of **pseudo-antichains**
- Symblicit algorithms in monotonic MDPs with a **natural partial order**
- Expected mean-payoff and stochastic shortest path
- Promising experimental results

Future work:

- Implementation of a MTBDD based symblicit algorithm for the stochastic shortest path
- Apply pseudo-antichains in other contexts (e.g. model-checking or synthesis of non-stochastic models)

Thank you!

Questions?

References I

Aaron Bohy, Véronique Bruyère, Emmanuel Filiot, and Jean-François Raskin.
Synthesis from LTL specifications with mean-payoff objectives.
In Nir Piterman and Scott A. Smolka, editors, *TACAS*, volume 7795 of *Lecture Notes in Computer Science*, pages 169–184. Springer, 2013.

Avrim L Blum and John C Langford.
Probabilistic planning in the graphplan framework.
In *Recent Advances in AI Planning*, pages 319–332. Springer, 2000.

D. P. Bertsekas and J. N. Tsitsiklis.
Neuro-Dynamic Programming.
Anthropological Field Studies. Athena Scientific, 1996.

Peter Buchholz.
Exact and ordinary lumpability in finite Markov chains.
Journal of applied probability, pages 59–75, 1994.

Luca de Alfaro.
Computing minimum and maximum reachability times in probabilistic systems.
In Jos C. M. Baeten and Sjouke Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 1999.

Richard E Fikes and Nils J Nilsson.
STRIPS: A new approach to the application of theorem proving to problem solving.
Artificial intelligence, 2(3):189–208, 1972.

References II


Ronald A. Howard.
Dynamic Programming and Markov Processes.
John Wiley and Sons, 1960.


John G. Kemeny and J. L. Snell.
Finite Markov Chains.
Van Nostrand Company, Inc, 1960.


Kim G. Larsen and Arne Skou.
Bisimulation through probabilistic testing.
Inf. Comput., 94(1):1–28, 1991.


Christian Von Essen.
personal communication, 20-11-2013.


R. Wimmer, B. Braitling, B. Becker, E. M. Hahn, P. Crouzen, H. Hermanns, A. Dhama, and O. E. Theel.
Symblicit calculation of long-run averages for concurrent probabilistic systems.
In *QEST*, pages 27–36. IEEE Computer Society, 2010.

LTL synthesis with mean-payoff objectives

- Synthesis from LTL specifications with MP objectives [BBFR13]
- Reduction to a 2-player safety game $(\mathcal{G}, \alpha)$ with a partial order
- Winning strategies represented by $\mathsf{Win}_1(\mathcal{G}, \alpha)$

LTL synthesis with mean-payoff objectives

- Synthesis from LTL specifications with MP objectives [BBFR13]
- Reduction to a 2-player safety game $(\mathcal{G}, \alpha)$ with a partial order
- Winning strategies represented by $\mathsf{Win}_1(\mathcal{G}, \alpha)$

- Goal: compute the **best strategy against a stochastic opponent**
- Replace the second player by a probability distribution in $\mathsf{Win}_1(\mathcal{G}, \alpha)$
- $\Rightarrow$ Monotonic MDP that satisfies Assumptions 1 and 2
- Symblicit algorithm for the expected mean-payoff problem
- Implementation in Acacia+