Symblicit algorithms for optimal strategy synthesis in monotonic Markov decision processes

Aaron Bohy\textsuperscript{1} Véronique Bruyère\textsuperscript{1} Jean-François Raskin\textsuperscript{2}

\textsuperscript{1}Université de Mons \quad \textsuperscript{2}Université Libre de Bruxelles

SYNT 2014
3rd workshop on Synthesis
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

---

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 *symbolic* and *explicit* data structures
• Expected mean-payoff in Markov decision processes
• Using (MT)BDDs

---

Overview (2/2)

Our motivations:

- Antichains sometimes outperform BDDs (e.g. [WDHR06, DR07])
- Use **antichains** instead of (MT)BDDs in symblicit algorithms
Overview (2/2)

Our motivations:

- Antichains sometimes outperform BDDs (e.g. [WDHR06, DR07])
- Use **antichains** instead of (MT)BDDs in symblicit algorithms

Our contributions:

- New structure of **pseudo-antichain** (extension of antichains)
  - Closed under negation
- **Monotonic** Markov decision processes
- **Two quantitative settings:**
  - Stochastic shortest path (focus of this talk)
  - Expected mean-payoff
- **Two applications:**
  - Automated planning
  - LTL synthesis

Full paper available on ArXiv: abs/1402.1076
Table of contents

Definitions

Symblicit approach

Antichains and pseudo-antichains

Monotonic Markov decision processes

Applications

Conclusion and future work
Table of contents

Definitions

Symblicit approach

Antichains and pseudo-antichains

Monotonic MDPs

Applications

Conclusion and future work
Markov decision processes (MDPs)

- $M = (S, \Sigma, \mathcal{P})$ where:
  - $S$ is a finite set of *states*
  - $\Sigma$ is a finite set of *actions*
  - $\mathcal{P} : S \times \Sigma \rightarrow \text{Dist}(S)$ is a *stochastic transition function*
Markov decision processes (MDPs)

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

- $M = (S, \Sigma, P)$ where:
  - $S$ is a finite set of states
  - $\Sigma$ is a finite set of actions
  - $P : S \times \Sigma \rightarrow \text{Dist}(S)$ is a stochastic transition function

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

- (Memoryless) strategy $\lambda : S \rightarrow \Sigma$
Markov chains (MCs)

- MDP \((S, \Sigma, \mathcal{P})\) with \(\mathcal{P} : S \times \Sigma \rightarrow \text{Dist}(S)\) + strategy \(\lambda : S \rightarrow \Sigma\) \(\Rightarrow\) induced MC \((S, \mathcal{P}_\lambda)\) with \(\mathcal{P}_\lambda : S \rightarrow \text{Dist}(S)\)
Markov chains (MCs)

- MDP \((S, \Sigma, P)\) with \(P : S \times \Sigma \rightarrow Dist(S)\) + strategy \(\lambda : S \rightarrow \Sigma\) \(\Rightarrow \) induced MC \((S, P_\lambda)\) with \(P_\lambda : S \rightarrow Dist(S)\)

- Cost function \(c : S \times \Sigma \rightarrow \mathbb{R}_{>0}\) + strategy \(\lambda : S \rightarrow \Sigma\) \(\Rightarrow \) induced cost function \(c_\lambda : S \rightarrow \mathbb{R}_{>0}\)
Expected truncated sum

- Let $M_\lambda = (S, P_\lambda)$ with cost function $c_\lambda$
- Let $G \subseteq S$ be a set of goal states
Expected truncated sum

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

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

• Let $M_\lambda = (S, P_\lambda)$ with cost function $c_\lambda$
• Let $G \subseteq S$ be a set of goal states

$TS_G(\rho = s_0s_1s_2 \ldots) = \sum_{i=0}^{n-1} c_\lambda(s_i)$, with $n$ first index s.t. $s_n \in G$

$E^{TS_G}_\lambda(s) = \sum_\rho P_\lambda(\rho)TS_G(\rho)$, with $\rho = s_0s_1 \ldots s_n$ s.t. $s_0 = s, s_n \in G$ and $s_0, \ldots, s_{n-1} \not\in G$
Stochastic shortest path (SSP)

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

- Let $M = (S, \Sigma, \mathcal{P})$ with cost function $c$
- Let $G \subseteq S$ be a set of goal states

$\lambda^*$ is *optimal* if $\mathbb{E}_{\lambda^*}^{TS_G}(s) = \inf_{\lambda \in \Lambda} \mathbb{E}_{\lambda}^{TS_G}(s)$

- SSP problem: compute an optimal strategy $\lambda^*$

- Complexity and strategies [BT96]:
  - Polynomial time via linear programming
  - Memoryless optimal strategies exist
Table of contents

Definitions

Symblicit approach

Antichains and pseudo-antichains

Monotonic Markov decision processes

Applications

Conclusion and future work
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

**Algorithm 1** Symblicit(MDP $M^S$, Cost function $c^S$, Goal states $G^S$)

1: $n := 0$, $\lambda_n^S := \text{InitialStrategy}(M^S, G^S)$
2: repeat
3: $(M_{\lambda_n}^S, c_{\lambda_n}^S) := \text{InducedMCAndCost}(M^S, c^S, \lambda_n^S)$
4: $(M_{\lambda_n, \sim_L}^S, c_{\lambda_n, \sim_L}^S) := \text{Lump}(M_{\lambda_n}^S, c_{\lambda_n}^S)$
5: $(M_{\lambda_n, \sim_L}^S, c_{\lambda_n, \sim_L}^S) := \text{Explicit}(M_{\lambda_n, \sim_L}^S, c_{\lambda_n, \sim_L}^S)$
6: $v_n := \text{SolveLinearSystem}(M_{\lambda_n, \sim_L}^S, c_{\lambda_n, \sim_L}^S)$
7: $v_n^S := \text{Symbolic}(v_n)$
8: $\lambda_{n+1}^S := \text{ImproveStrategy}(M^S, \lambda_n^S, v_n^S)$
9: $n := n + 1$
10: until $\lambda_n^S = \lambda_{n-1}^S$
11: return $(\lambda_{n-1}^S, v_{n-1}^S)$

Key: $S$ in superscript denotes symbolic representations
Table of contents

Definitions

Symblicit approach

Antichains and pseudo-antichains

Monotonic Markov decision processes

Applications

Conclusion and future work
Antichains

- Let \((S, \preceq)\) be a semilattice with greatest lower bound
- 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 \(\alpha\) is \(\downarrow \alpha = \{ s \in S \mid \exists a \in \alpha, s \preceq a \}\)
- Example: \(\alpha = \{a_1, a_2\}\)
Antichains

- Let \((S, \preceq)\) be a semilattice with greatest lower bound
- 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 \(\alpha\) 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 by their maximal elements (*unique*)
- **Efficient computations** of closures of antichains w.r.t. union and intersection
- **But** antichains are not closed under negation
Pseudo-elements

- Let \((S, \preceq)\) be a semilattice with greatest lower bound.
- A **pseudo-element** is a pair \((x, \alpha)\) where \(x \in S\) and \(\alpha \subseteq S\) is an antichain such that \(x \not\in \downarrow \alpha\).
- The **pseudo-closure** of \((x, \alpha)\) is \(\uparrow(x, \alpha) = \{ s \in S \mid s \preceq x \text{ and } s \not\in \downarrow \alpha \} = \downarrow \{x\} \setminus \downarrow \alpha\).
- Example: \((x, \alpha)\) with \(\alpha = \{a_1, a_2\}\).

![Diagram showing pseudo-elements and related concepts]

\[x\]
\[a_1\]
\[a_2\]
Pseudo-elements

- Let \((S, \preceq)\) be a semilattice with greatest lower bound
- A **pseudo-element** is a pair \((x, \alpha)\) where \(x \in S\) and \(\alpha \subseteq S\) is an antichain such that \(x \not\in \downarrow \alpha\)
- The **pseudo-closure** of \((x, \alpha)\) is \(\uparrow(x, \alpha) = \{s \in S \mid s \preceq x \text{ and } s \not\in \downarrow \alpha\}\)
  \[= \downarrow \{x\} \setminus \downarrow \alpha\]
- Example: \((x, \alpha)\) with \(\alpha = \{a_1, a_2\}\)

- \((x, \alpha)\) 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 $\uparrow A = \bigcup_{i \in I} \uparrow(x_i, \alpha_i)$.
A pseudo-antichain $A$ is a set $\{ (x_i, \alpha_i) \mid i \in I \}$ of pseudo-elements.

The pseudo-closure of $A$ is $\uparrow A = \bigcup_{i \in I} \downarrow (x_i, \alpha_i)$.

$A$ is a $PA$-representation of $\uparrow A$ (not 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 $\uparrow A = \bigcup_{i \in I} \uparrow (x_i, \alpha_i)$.
- $A$ is a **PA-representation** of $\uparrow A$ (not unique).
- Any set can be PA-represented.
- **Efficient computations** of pseudo-closures of pseudo-antichains w.r.t. the union, intersection and negation.
Table of contents

Definitions

Symblicit approach

Antichains and pseudo-antichains

Monotonic Markov decision processes

Applications

Conclusion and future work
Monotonic properties

Intuition on a transition system (TS) \((S, \Sigma, \Delta)\) where:

- \(S\): set of states
- \(\Sigma\): set of actions
- \(\Delta\): transition function
Monotonic properties

Intuition on a transition system (TS) \((S, \Sigma, \Delta)\) where:

- \(S\): set of states
- \(\Sigma\): set of actions
- \(\Delta\): transition function

A **monotonic** TS is a TS \((S, \Sigma, \Delta)\) s.t.:
- \(S\) is equipped with a partial order \(\preceq\) s.t. \((S, \preceq)\) is a semilattice
Monotonic properties

Intuition on a transition system (TS) \((S, \Sigma, \Delta)\) where:

- \(S\): set of states
- \(\Sigma\): set of actions
- \(\Delta\): transition function

A **monotonic** TS is a TS \((S, \Sigma, \Delta)\) s.t.:

- \(S\) is equipped with a partial order \(\preceq\) s.t. \((S, \preceq)\) is a semilattice
- \(\preceq\) is **compatible** with \(\Delta\), i.e. \(\forall s, s' \in S\)
  \[
  s \preceq s'
  \]
Monotonic properties

Intuition on a transition system (TS) \((S, \Sigma, \Delta)\) where:

- \(S\): set of states
- \(\Sigma\): set of actions
- \(\Delta\): transition function

A **monotonic** TS is a TS \((S, \Sigma, \Delta)\) s.t.:

- \(S\) is equipped with a partial order \(\preceq\) s.t. \((S, \preceq)\) is a semilattice
- \(\preceq\) is **compatible** with \(\Delta\), i.e. \(\forall s, s' \in S\)

\[
\begin{align*}
  s & \preceq s' \\
  \forall \sigma \in \Sigma
\end{align*}
\]

\[
\Delta(s', \sigma) \rightarrow t'
\]
Monotonic properties

Intuition on a transition system (TS) \((S, \Sigma, \Delta)\) where:

- \(S\): set of states
- \(\Sigma\): set of actions
- \(\Delta\): transition function

A monotonic TS is a TS \((S, \Sigma, \Delta)\) s.t.:

- \(S\) is equipped with a partial order \(\preceq\) s.t. \((S, \preceq)\) is a semilattice
- \(\preceq\) is compatible with \(\Delta\), i.e. \(\forall s, s' \in S\)

\[
\begin{align*}
  s \preceq s' & \quad \forall \sigma \in \Sigma \\
  \Delta(s', \sigma) & \quad \exists t, t' 
\end{align*}
\]
Monotonic properties

Intuition on a transition system (TS) $(S, \Sigma, \Delta)$ where:

- $S$: set of states
- $\Sigma$: set of actions
- $\Delta$: transition function

A **monotonic** TS is a TS $(S, \Sigma, \Delta)$ s.t.:

- $S$ is equipped with a partial order $\preceq$ s.t. $(S, \preceq)$ is a semilattice
- $\preceq$ is compatible with $\Delta$, i.e. $\forall s, s' \in S$
  
  \[
  s \preceq s' \Rightarrow \exists t, t' \in S
  \]

  $\Delta(s, \sigma) \rightarrow t$

  $\Delta(s', \sigma) \rightarrow t'$
Monotonic properties

Intuition on a transition system (TS) \((S, \Sigma, \Delta)\) where:

- \(S\): set of states
- \(\Sigma\): set of actions
- \(\Delta\): transition function

A **monotonic** TS is a TS \((S, \Sigma, \Delta)\) s.t.:

- \(S\) is equipped with a partial order \(\preceq\) s.t. \((S, \preceq)\) is a semilattice
- \(\preceq\) is *compatible* with \(\Delta\), i.e. \(\forall s, s' \in S\)

\[
\begin{align*}
\Delta(s, \sigma) & \preceq \Delta(s', \sigma) \\
\exists t & \preceq t'
\end{align*}
\]
Monotonic Markov decision processes

Monotonic MDP:

- MDP s.t. its underlying TS is monotonic

Remark:

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

⇒ Pseudo-antichain based symblicit algorithm for monotonic MDPs
Table of contents

Definitions

Symblicit approach

Antichains and pseudo-antichains

Monotonic Markov decision processes

Applications

Conclusion and future work
A STRIPS is a tuple \((P, I, M, O)\) where

- \(P\) is a finite set of \emph{propositional variables}
- \(I \subseteq P\) is a subset of \emph{initial} variables
- \(M \subseteq P\) is a subset of \emph{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 \(o\)
  - \((\alpha, \delta)\), with \(\alpha, \delta \subseteq P\), is the *effect* of \(o\)
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 $o$
  - $(\alpha, \delta)$, with $\alpha, \delta \subseteq P$, is the effect of $o$
STRIPTS

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 \(o\)
  - \((\alpha, \delta)\), with \(\alpha, \delta \subseteq P\), is the *effect* of \(o\)
STRIPS

A \textit{STRIPS} is a tuple \((P, I, M, O)\) where

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

\[s \xrightarrow{(\gamma, (\alpha, \delta))} s'\]

\(s \supseteq \gamma\)

\(s' = (s \cup \alpha) \setminus \delta\)

\(\Rightarrow\) TS with monotonic properties
A STRIPS is a tuple \((P, I, M, O)\) where

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

\[s \xrightarrow{(\gamma, (\alpha, \delta))} s'\]

\(s \supseteq \gamma\)

\(s' = (s \cup \alpha) \setminus \delta\)

\(\Rightarrow\) TS with \textbf{monotonic} properties

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]
  • Probability distribution on the effects of operators

⇝ Monotonic Markov decision processes
Stochastic STRIPS

- Extension of STRIPS with **stochastic aspects** [BL00]
  - Probability distribution on the effects of operators

  \[\rightsquigarrow\] **Monotonic** Markov decision processes

- **Cost function** \( C : O \rightarrow \mathbb{R}_{>0} \)
- Planning from stochastic STRIPS
  - Minimize the expected truncated sum up to a state \( s \supseteq M \) from \( I \)

  \[\rightsquigarrow\] **Stochastic shortest path problem**
## Experimental results

| example    | $\mathbb{E}^{TS_G}_\lambda$ | $|M_S|$ | #it | $|\sim_L|$ | time | mem | Explicit time | 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

Results from [BBFR13]:

- Synthesis from LTL specifications with mean-payoff objectives
- Reduction to a 2-player safety game (SG)
  - between the system and its environment
  - equipped with a partial order (monotonic properties)
Expected mean-payoff with LTL synthesis

Results from [BBFR13]:

- Synthesis from LTL specifications with mean-payoff objectives
- Reduction to a 2-player safety game (SG)
  - between the system and its environment
  - equipped with a partial order (monotonic properties)

Goal: compute a **worst-case** winning strategy with **good expected performance**
Expected mean-payoff with LTL synthesis

Results from [BBFR13]:

- Synthesis from LTL specifications with mean-payoff objectives
- Reduction to a 2-player safety game (SG)
  - between the system and its environment
  - equipped with a partial order (monotonic properties)

Goal: compute a **worst-case** winning strategy with **good expected performance**

Idea:

- Replace the environment by a probability distribution in the SG restricted to winning states

\[\rightsquigarrow\textbf{Monotonic MDP}\]

- Symblicit algorithm for the expected mean-payoff problem
- Implementation in Acacia+
Experimental results

Comparison with an MTBDD based symblicit algorithm [VE13]

Figure: Execution time

⇒ Monotonic MDPs are better handled by pseudo-antichains

Figure: Memory consumption
Table of contents

Definitions

Symblicit approach

Antichains and pseudo-antichains

Monotonic Markov decision processes

Applications

Conclusion and future work
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 of probabilistic lossy channel systems)
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

Avrim L Blum and John C Langford.
Probabilistic planning in the graphplan framework.

D. P. Bertsekas and J. N. Tsitsiklis.
Neuro-Dynamic Programming.

Peter Buchholz.
Exact and ordinary lumpability in finite Markov chains.

Laurent Doyen and Jean-François Raskin.
Improved algorithms for the automata-based approach to model-checking.
In Orna Grumberg and Michael Huth, editors, TACAS, volume 4424 of Lecture Notes in

STRIPS: A new approach to the application of theorem proving to problem solving.
Ronald A. Howard.  
*Dynamic Programming and Markov Processes.*  

John G. Kemeny and J. L. Snell.  
*Finite Markov Chains.*  

Kim G. Larsen and Arne Skou.  
Bisimulation through probabilistic testing.  

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

Symblicit calculation of long-run averages for concurrent probabilistic systems.  

Martin De Wulf, Laurent Doyen, Thomas A. Henzinger, and Jean-François Raskin.  