Probabilistic Model Checking and Counterexample Generation

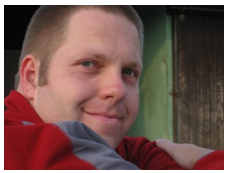
Erika Ábrahám

RWTH Aachen University, Germany

IFIP WG 2.2 Meeting
Munich, September 15-18, 2014

DFG CEBug project 2010-2013







How do probabilities and programming concepts relate?

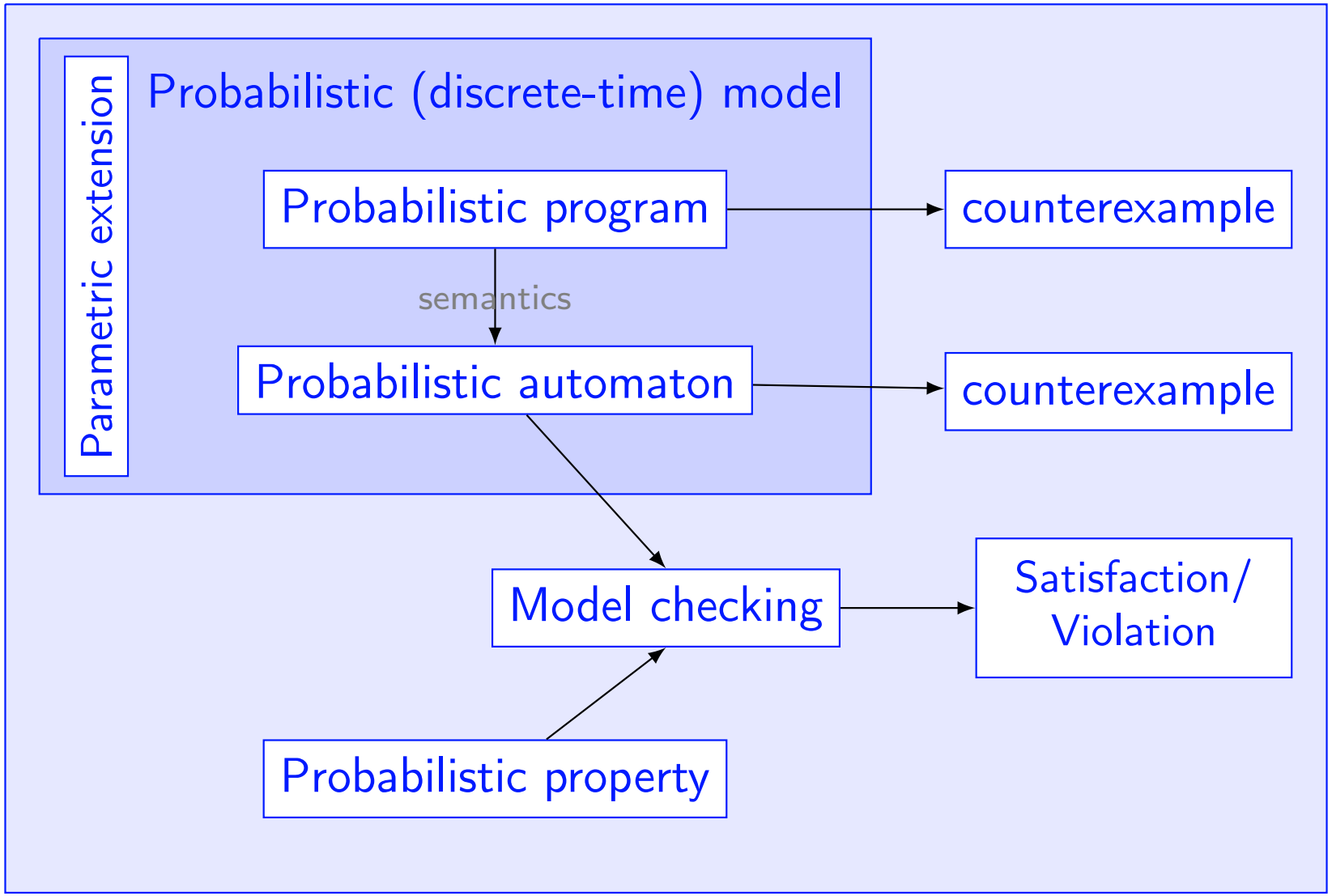
- Randomized algorithms
  - randomized network protocols, Google search algorithm,
  - cryptography,
  - ...
- Modeling and predicting complex/unreliable systems
  - message losses, processor failures,
  - waiting times, soft deadlines,
  - social networks, chemical plants, biological systems,
  - ...

How do probabilities and programming concepts relate?

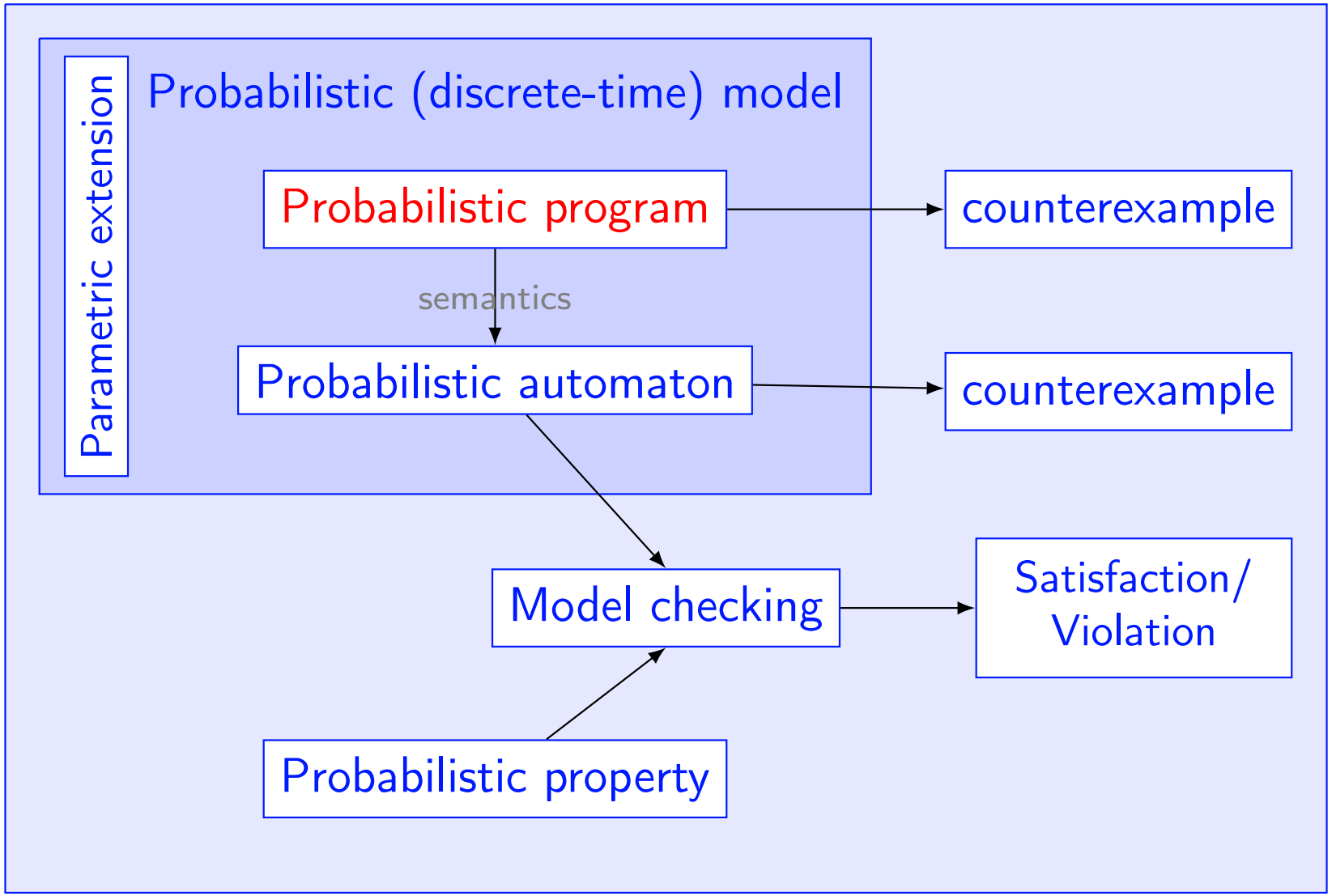
- Randomized algorithms
  - randomized network protocols, Google search algorithm,
  - cryptography,
  - ...
- Modeling and predicting complex/unreliable systems
  - message losses, processor failures,
  - waiting times, soft deadlines,
  - social networks, chemical plants, biological systems,
  - ...

Topic of this talk: Formal methods for modeling and analyzing such systems.

Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Example: Dueling Cowboys (slightly changed PRISM syntax)

Example: Dueling Cowboys (slightly changed PRISM syntax)

module Duel || Joe || Luke

turn: $\{\bot, \mathtt{J}, \mathtt{L}\}$ **init** $\bot$;

aliveJ, aliveL: **bool init** 1;

$$
\begin{array}{lll}
[\mathtt{start}] & \mathtt{turn} = \bot & \rightarrow 0.75: \mathtt{turn}' = \mathtt{L} + 0.25: \mathtt{turn}' = \mathtt{J}; \\
[\mathtt{luke}] & \mathtt{turn} = \mathtt{L} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.1: \mathtt{turn}' = \mathtt{J} + \\
 & & \quad\ 0.9: \mathtt{turn}' = \mathtt{J} \wedge \mathtt{aliveJ}' = 0; \\
[\mathtt{joe}] & \mathtt{turn} = \mathtt{J} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.4: \mathtt{turn}' = \mathtt{L} + \\
 & & \quad\ 0.6: \mathtt{turn}' = \mathtt{L} \wedge \mathtt{aliveL}' = 0; \\
 & & \\
[\tau] & \neg\mathtt{aliveJ} \vee \neg\mathtt{aliveL} & \rightarrow 1.0: \mathtt{skip};
\end{array}
$$

endmodule

Example: Dueling Cowboys (slightly changed PRISM syntax)

module Duel || Joe || Luke

turn: $\{\bot, \mathtt{J}, \mathtt{L}\}$ **init** $\bot$;

aliveJ, aliveL: **bool init** 1;

$$
\begin{array}{lll}
[\mathtt{start}]\ \mathtt{turn} = \bot & \rightarrow & 0.75: \mathtt{turn}' = \mathtt{L} + \ 0.25: \mathtt{turn}' = \mathtt{J}; \\
[\mathtt{luke}]\ \mathtt{turn} = \mathtt{L} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow & 0.1: \mathtt{turn}' = \mathtt{J} \ + \\
 & & 0.9: \mathtt{turn}' = \mathtt{J} \wedge \mathtt{aliveJ}' = 0; \\
[\mathtt{joe}]\ \mathtt{turn} = \mathtt{J} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow & 0.4: \mathtt{turn}' = \mathtt{L} \ + \\
 & & 0.6: \mathtt{turn}' = \mathtt{L} \wedge \mathtt{aliveL}' = 0; \\
 & & \\
[\tau]\ \neg\mathtt{aliveJ} \vee \neg\mathtt{aliveL} & \rightarrow & 1.0: \mathtt{skip};
\end{array}
$$

endmodule

Example: Dueling Cowboys (slightly changed PRISM syntax)

module Duel || Joe || Luke

turn: $\{\bot, \mathtt{J}, \mathtt{L}\}$ **init** $\bot$;

aliveJ, aliveL: **bool init** 1;

$$
\begin{array}{lll}
[\mathtt{start}] & \mathtt{turn} = \bot & \rightarrow 0.75: \mathtt{turn}' = \mathtt{L} + 0.25: \mathtt{turn}' = \mathtt{J}; \\
[\mathtt{luke}] & \mathtt{turn} = \mathtt{L} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.1: \mathtt{turn}' = \mathtt{J} + \\
 & & \quad\ 0.9: \mathtt{turn}' = \mathtt{J} \wedge \mathtt{aliveJ}' = 0; \\
[\mathtt{joe}] & \mathtt{turn} = \mathtt{J} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.4: \mathtt{turn}' = \mathtt{L} + \\
 & & \quad\ 0.6: \mathtt{turn}' = \mathtt{L} \wedge \mathtt{aliveL}' = 0; \\
 & & \\
[\tau] & \neg\mathtt{aliveJ} \vee \neg\mathtt{aliveL} & \rightarrow 1.0: \mathtt{skip};
\end{array}
$$

endmodule

Example: Dueling Cowboys (slightly changed PRISM syntax)

module Duel || Joe || Luke

turn: $\{\bot, \mathtt{J}, \mathtt{L}\}$ **init** $\bot$;

aliveJ, aliveL: **bool init** 1;

$$
\begin{array}{lll}
[\mathtt{start}] & \mathtt{turn} = \bot & \rightarrow 0.75: \mathtt{turn}' = \mathtt{L} + 0.25: \mathtt{turn}' = \mathtt{J}; \\
[\mathtt{luke}] & \mathtt{turn} = \mathtt{L} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.1: \mathtt{turn}' = \mathtt{J} + \\
 & & \quad\ 0.9: \mathtt{turn}' = \mathtt{J} \wedge \mathtt{aliveJ}' = 0; \\
[\mathtt{joe}] & \mathtt{turn} = \mathtt{J} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.4: \mathtt{turn}' = \mathtt{L} + \\
 & & \quad\ 0.6: \mathtt{turn}' = \mathtt{L} \wedge \mathtt{aliveL}' = 0; \\
 & & \\
[\tau] & \neg\mathtt{aliveJ} \vee \neg\mathtt{aliveL} & \rightarrow 1.0: \mathtt{skip};
\end{array}
$$

endmodule

Example: Dueling Cowboys (slightly changed PRISM syntax)

module Duel || Joe || Luke

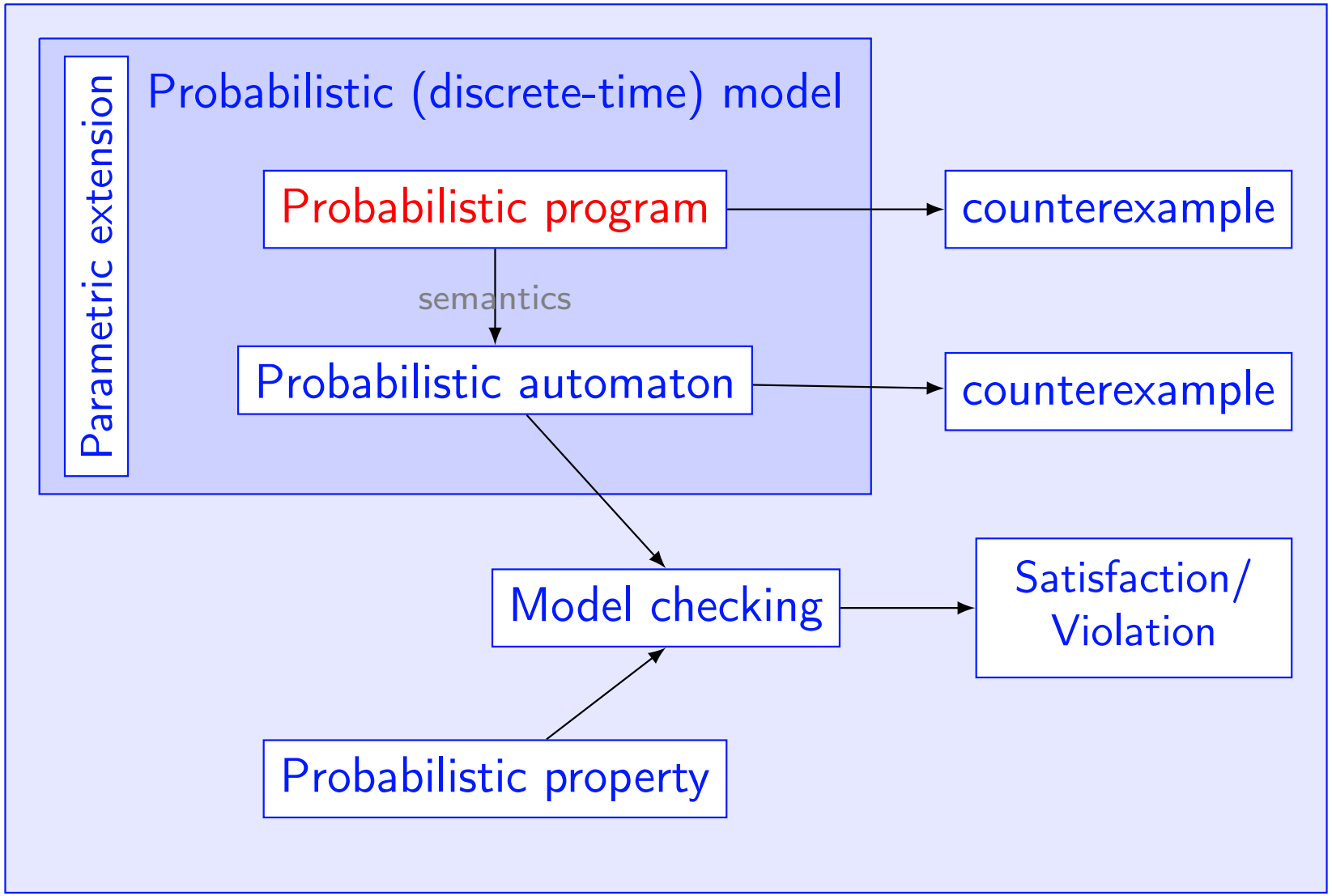
turn: $\{\bot, \mathtt{J}, \mathtt{L}\}$ **init** $\bot$;

aliveJ, aliveL: **bool init** 1;

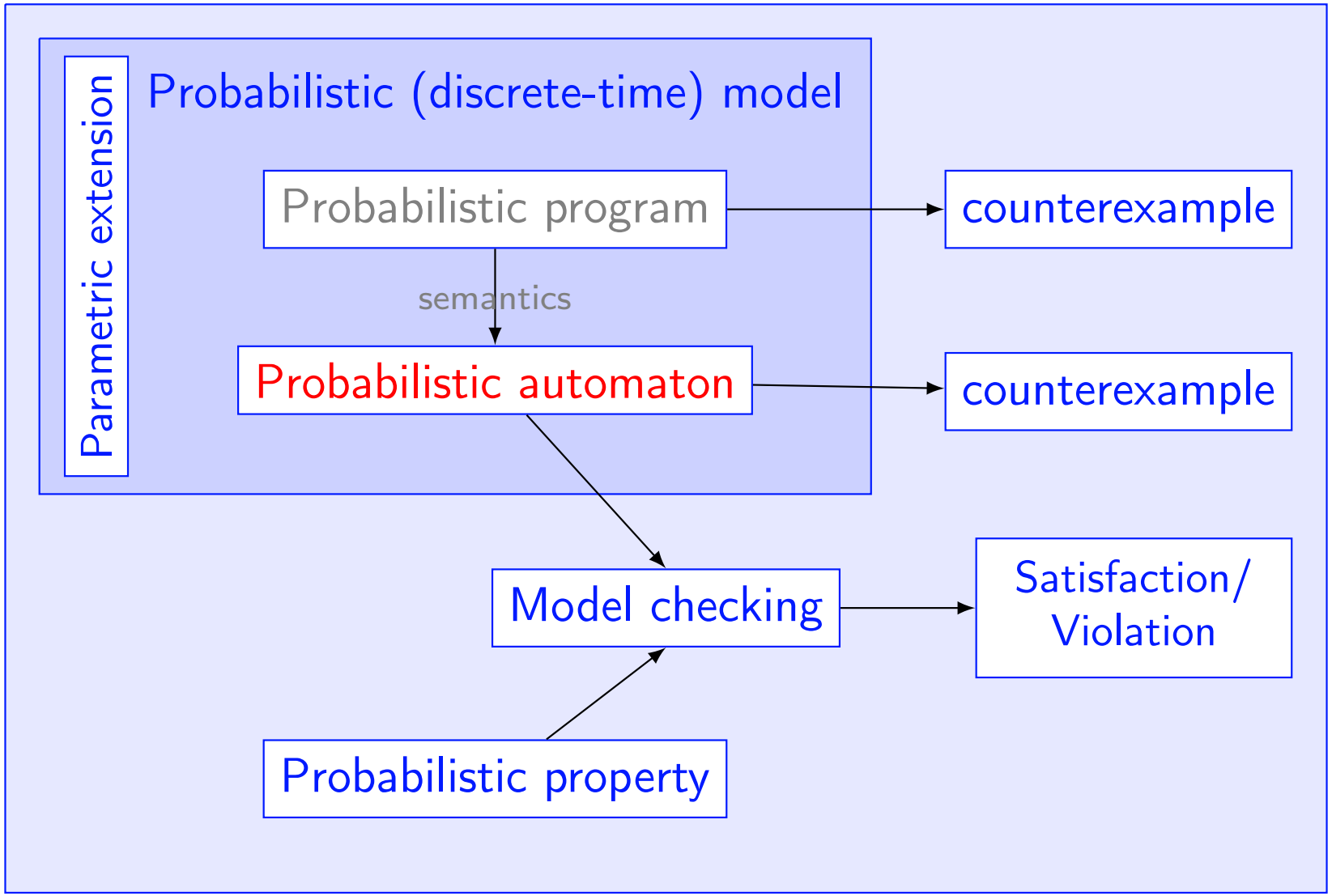
$$
\begin{array}{lll}
[\mathtt{start}] & \mathtt{turn} = \bot & \rightarrow 0.75: \mathtt{turn}' = \mathtt{L} + 0.25: \mathtt{turn}' = \mathtt{J}; \\
[\mathtt{luke}] & \mathtt{turn} = \mathtt{L} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.1: \mathtt{turn}' = \mathtt{J} + \\
 & & \quad\ 0.9: \mathtt{turn}' = \mathtt{J} \wedge \mathtt{aliveJ}' = 0; \\
[\mathtt{joe}] & \mathtt{turn} = \mathtt{J} \wedge \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 0.4: \mathtt{turn}' = \mathtt{L} + \\
 & & \quad\ 0.6: \mathtt{turn}' = \mathtt{L} \wedge \mathtt{aliveL}' = 0; \\
[\mathtt{hug}] & \mathtt{aliveJ} \wedge \mathtt{aliveL} & \rightarrow 1.0: \mathtt{turn}' = \bot; \\
[\tau] & \neg\mathtt{aliveJ} \vee \neg\mathtt{aliveL} & \rightarrow 1.0: \mathtt{skip};
\end{array}
$$

endmodule

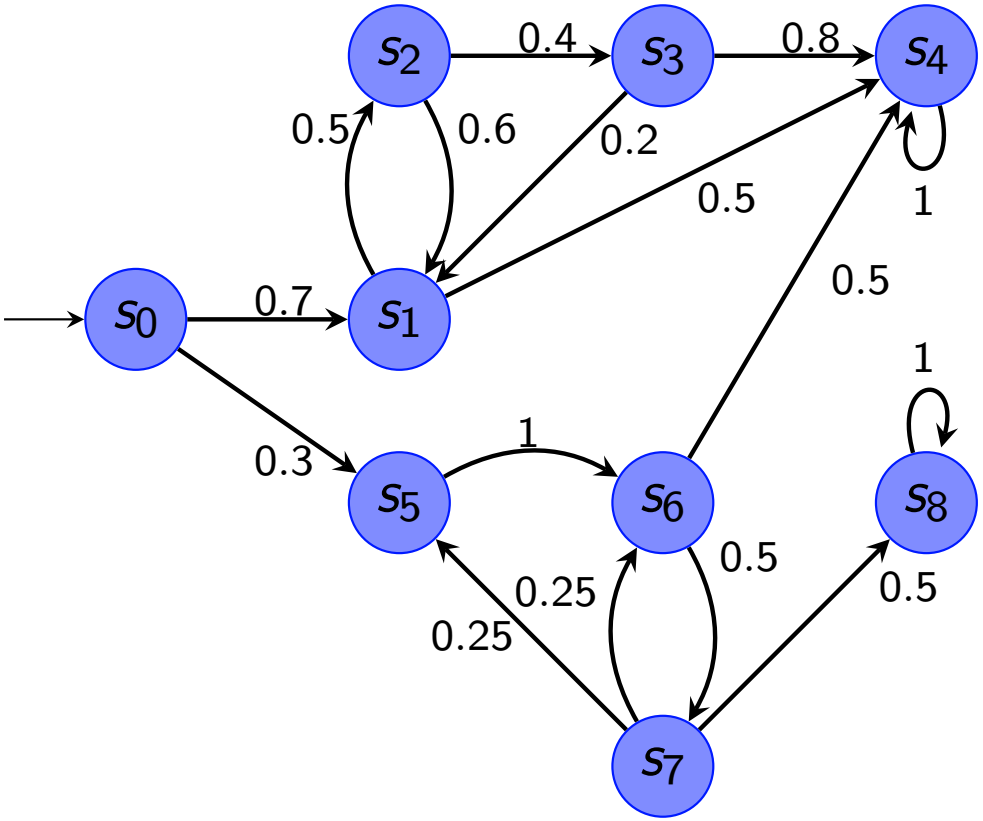
Contents


Probabilistic (discrete-time) model
Parametric extension
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

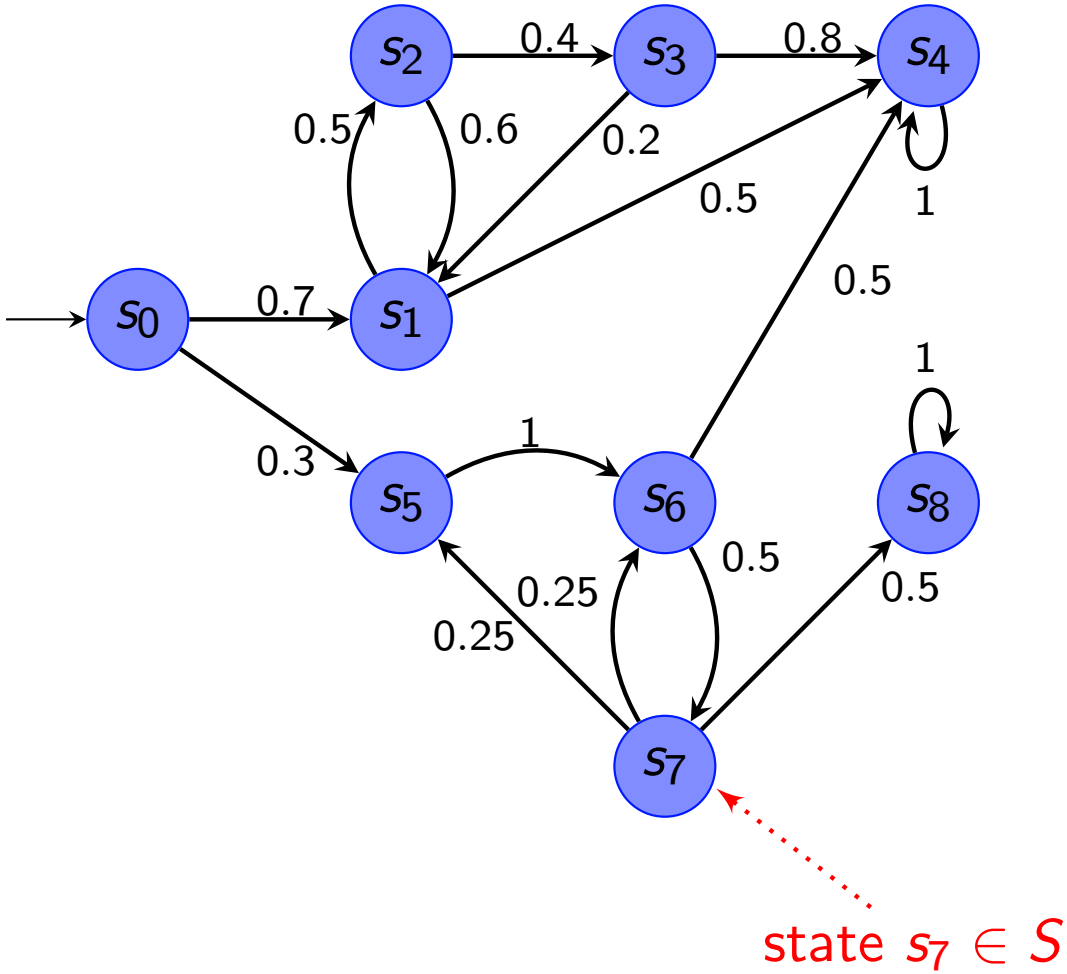
Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

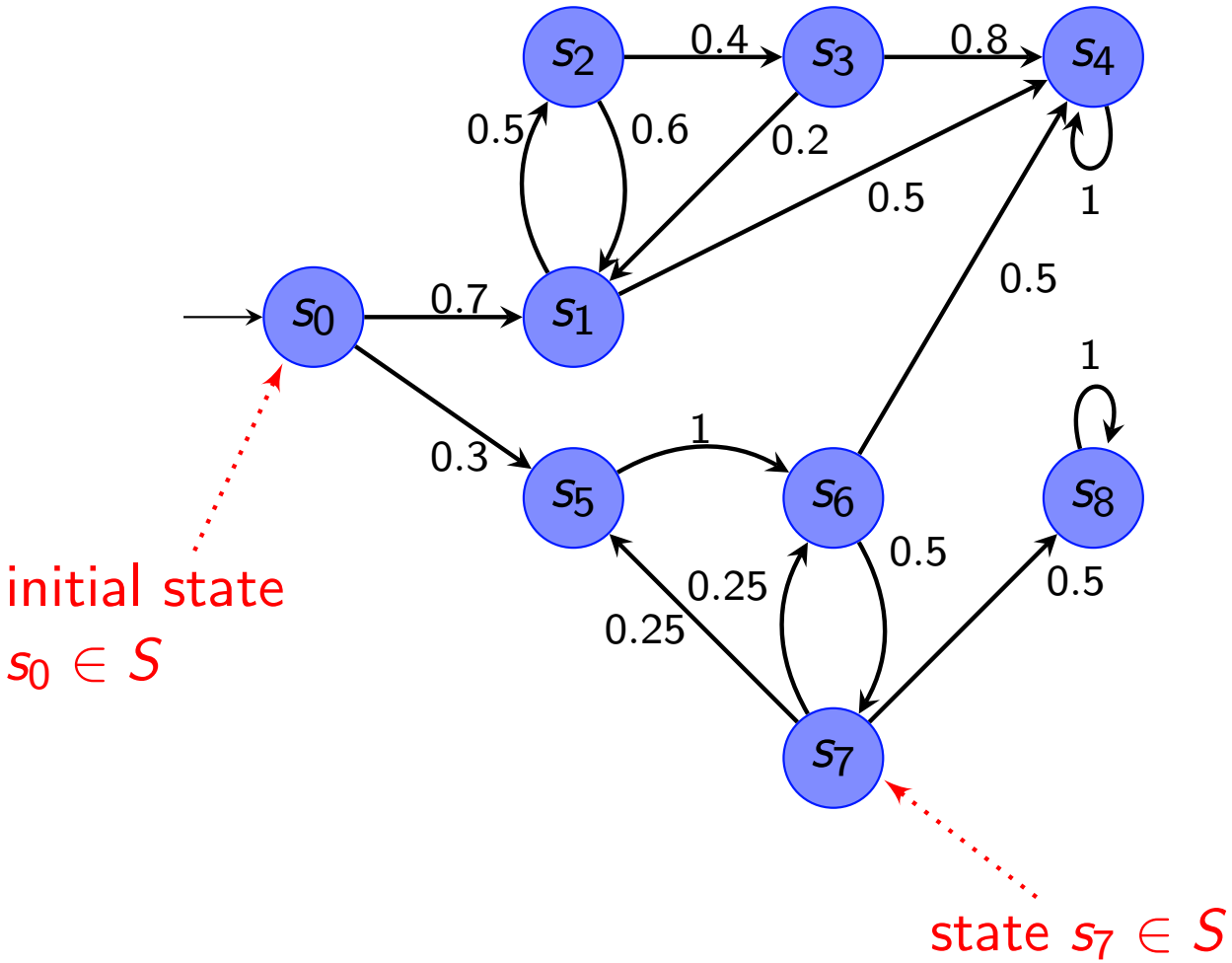
Discrete-time Markov chains (DTMCs)



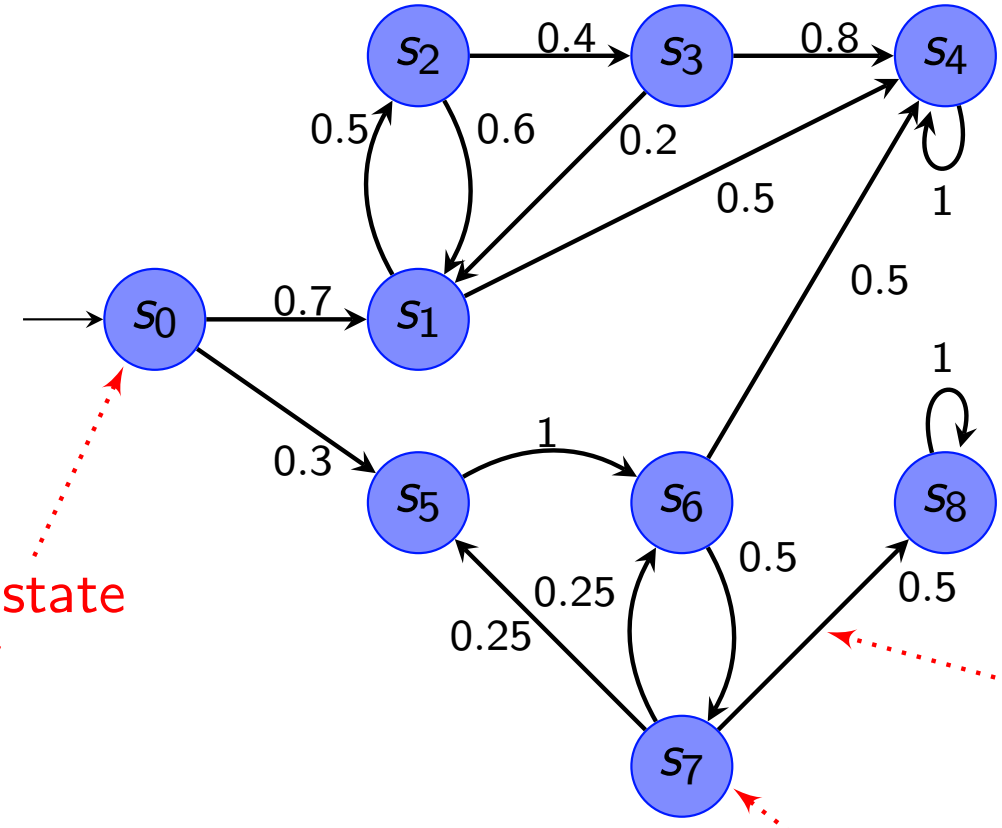
Discrete-time Markov chains (DTMCs)



Discrete-time Markov chains (DTMCs)



Discrete-time Markov chains (DTMCs)

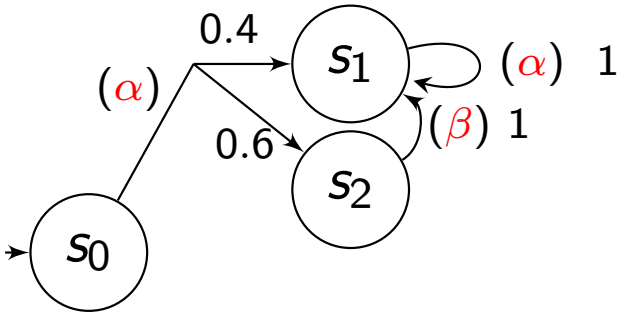


initial state
$s_0 \in S$

transition with probability
$P(s_7, s_8) = 0.5$

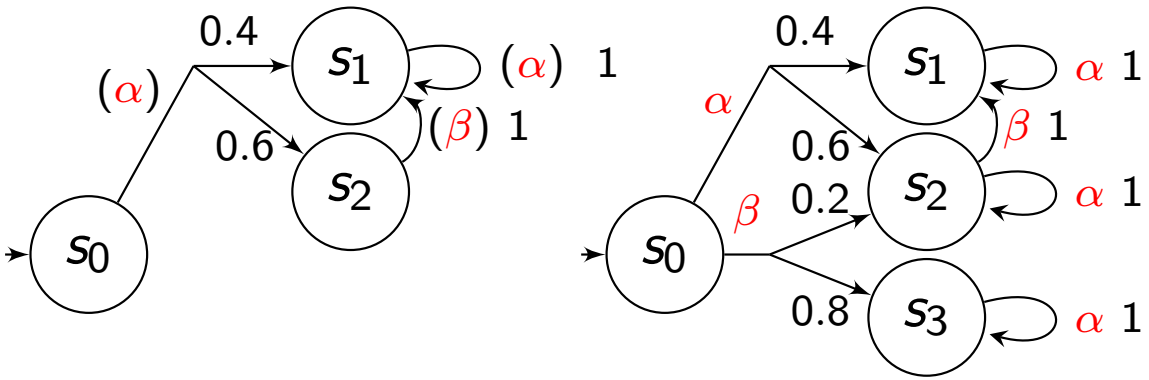
state $s_7 \in S$

Markov decision processes and probabilistic automata



Discrete-time Markov chain (DTMC)

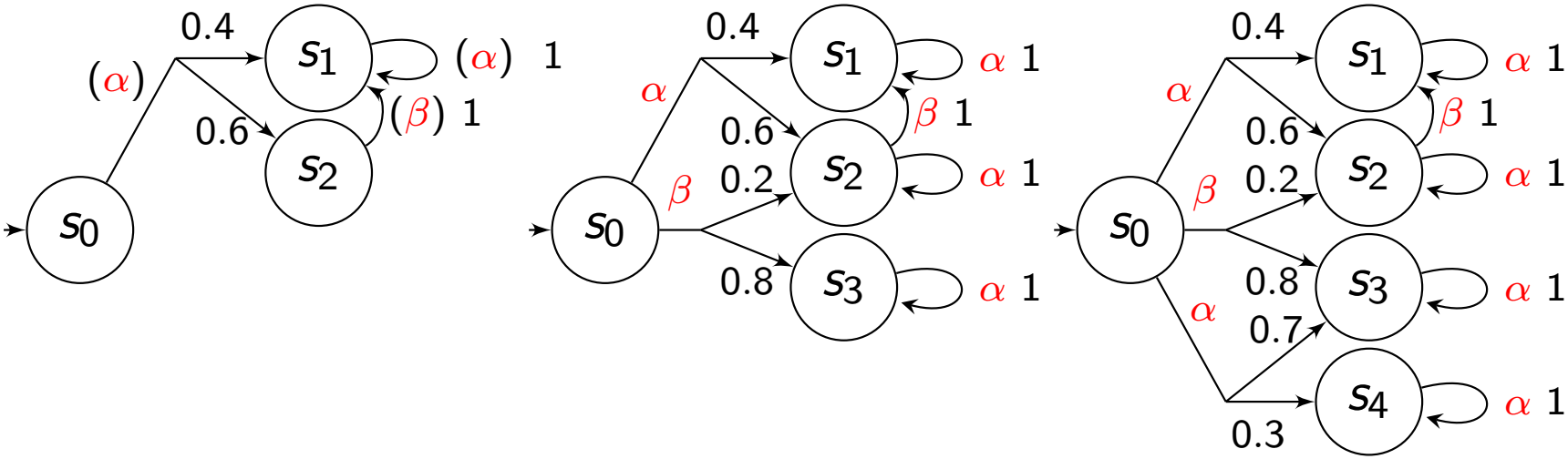
Markov decision processes and probabilistic automata



Discrete-time Markov chain (DTMC)

Markov decision process (MDP)

Markov decision processes and probabilistic automata



Discrete-time Markov chain (DTMC)

Probabilistic automaton (PA)

Markov decision process (MDP)

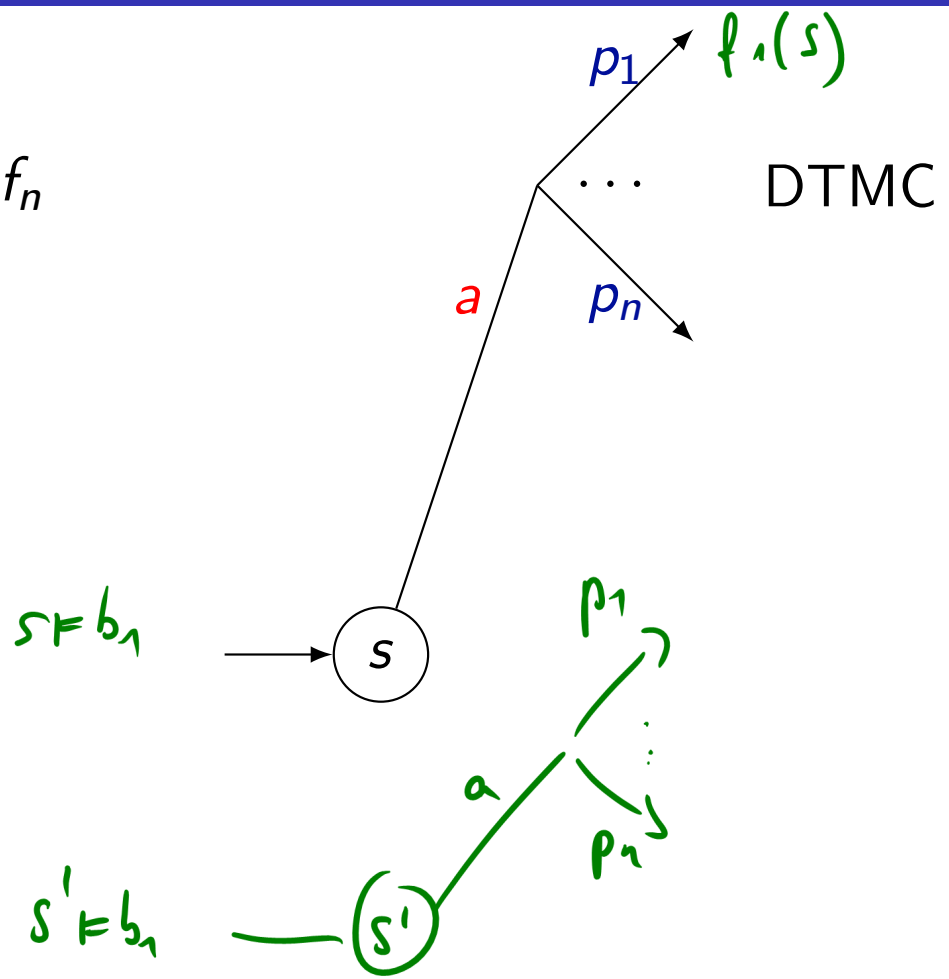
Interpreting PRISM modules

Interpreting PRISM modules

$$c_1 = [a]\ b_1 \rightarrow p_1 : f_1 + \ldots + p_n : f_n$$

Interpreting PRISM modules

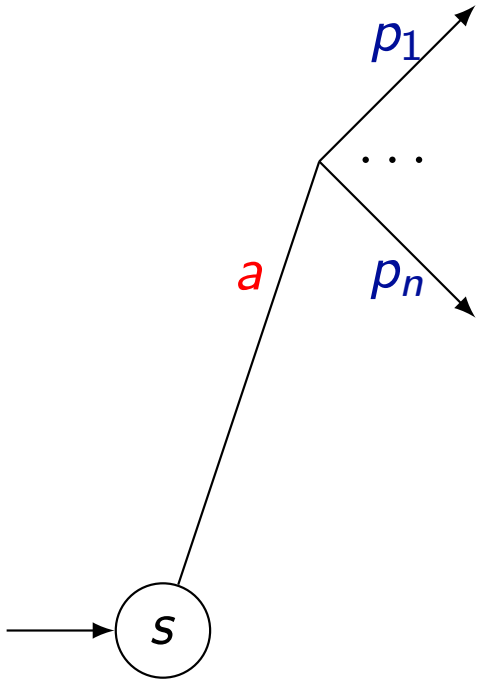
$$c_1 = [a]\ b_1 \rightarrow p_1 : f_1 + \ldots + p_n : f_n$$



Interpreting PRISM modules

$$c_1 = [a]\ b_1 \rightarrow p_1 : f_1 + \ldots + p_n : f_n$$

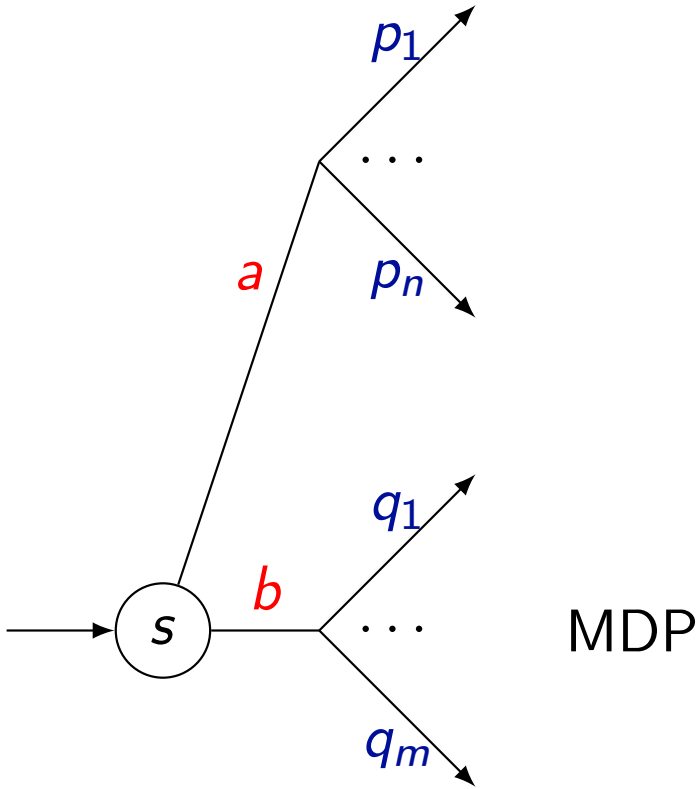
$$c_2 = [b]\ b_2 \rightarrow q_1 : g_1 + \ldots + q_m : g_m$$



Interpreting PRISM modules

$$c_1 = [a]\ b_1 \to p_1 : f_1 + \ldots + p_n : f_n$$

$$c_2 = [b]\ b_2 \to q_1 : g_1 + \ldots + q_m : g_m$$

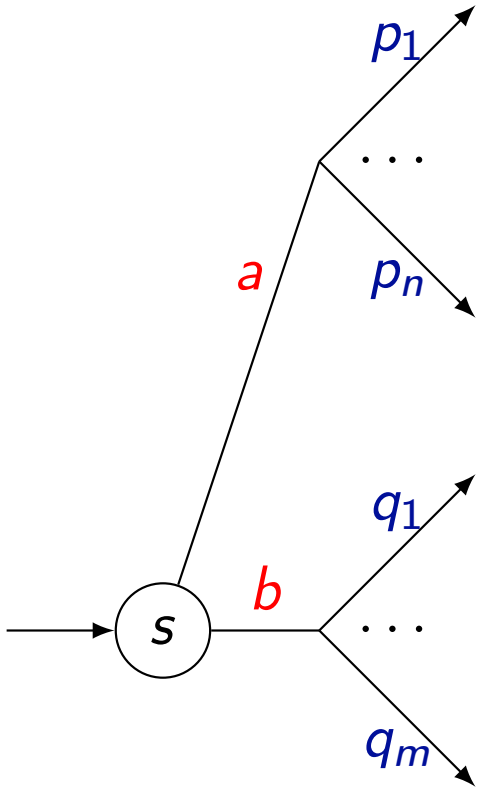


MDP

Interpreting PRISM modules

$c_1 = [a]\ b_1 \rightarrow p_1 : f_1 + \ldots + p_n : f_n$

$c_2 = [b]\ b_2 \rightarrow q_1 : g_1 + \ldots + q_m : g_m$



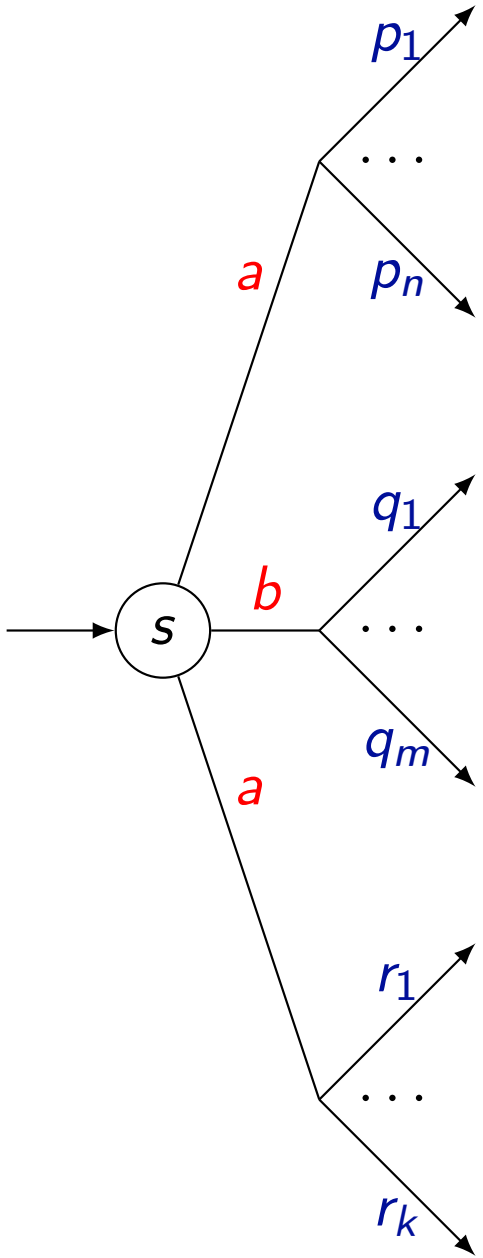
$c_3 = [a]\ b_3 \rightarrow r_1 : h_1 + \ldots + r_k : h_k$

Interpreting PRISM modules

$c_1 = [a]\ b_1 \to p_1 : f_1 + \ldots + p_n : f_n$

$c_2 = [b]\ b_2 \to q_1 : g_1 + \ldots + q_m : g_m$

$c_3 = [a]\ b_3 \to r_1 : h_1 + \ldots + r_k : h_k$



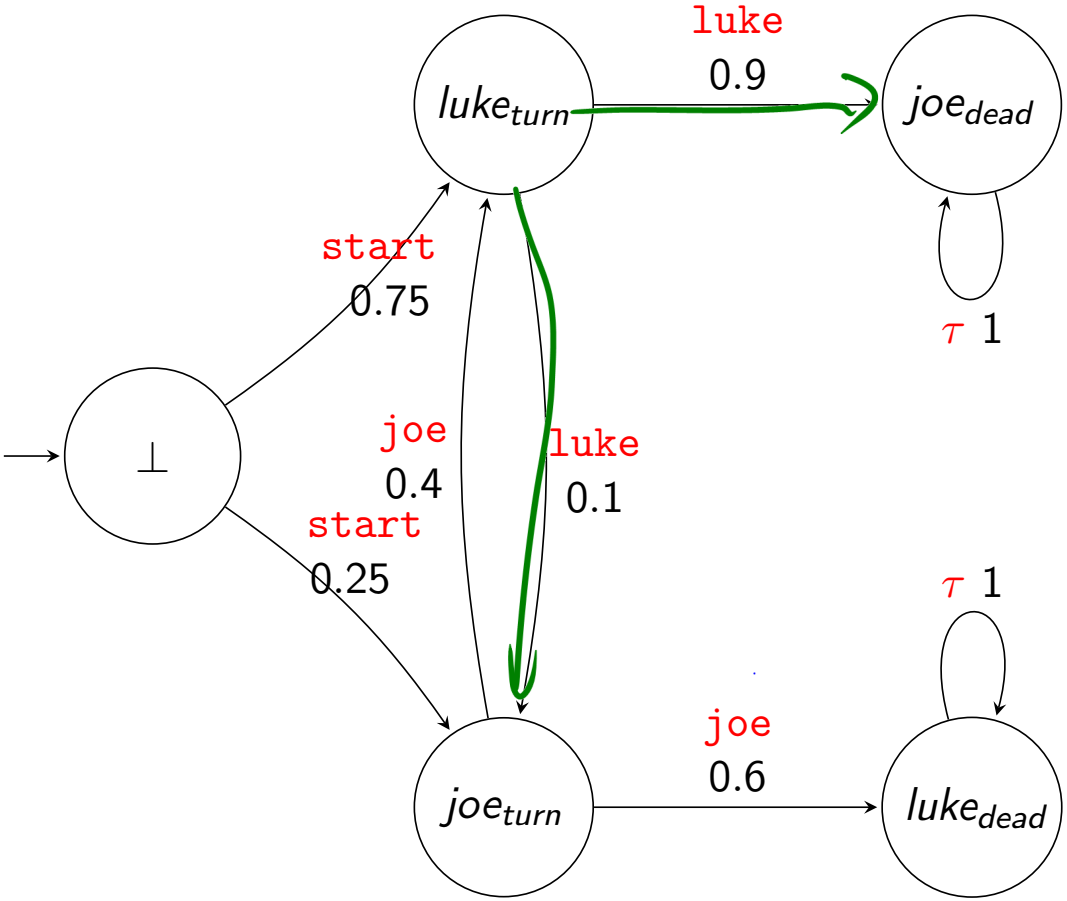
PA

Example: Dueling cowboys

module Duel || Joe || Luke

$\ldots$[luke] turn = L $\wedge$ aliveJ $\wedge$ aliveL $\rightarrow$ 0.1: turn$'$ = J +
0.9: turn$'$ = J $\wedge$ aliveJ$'$ = 0;

endmodule

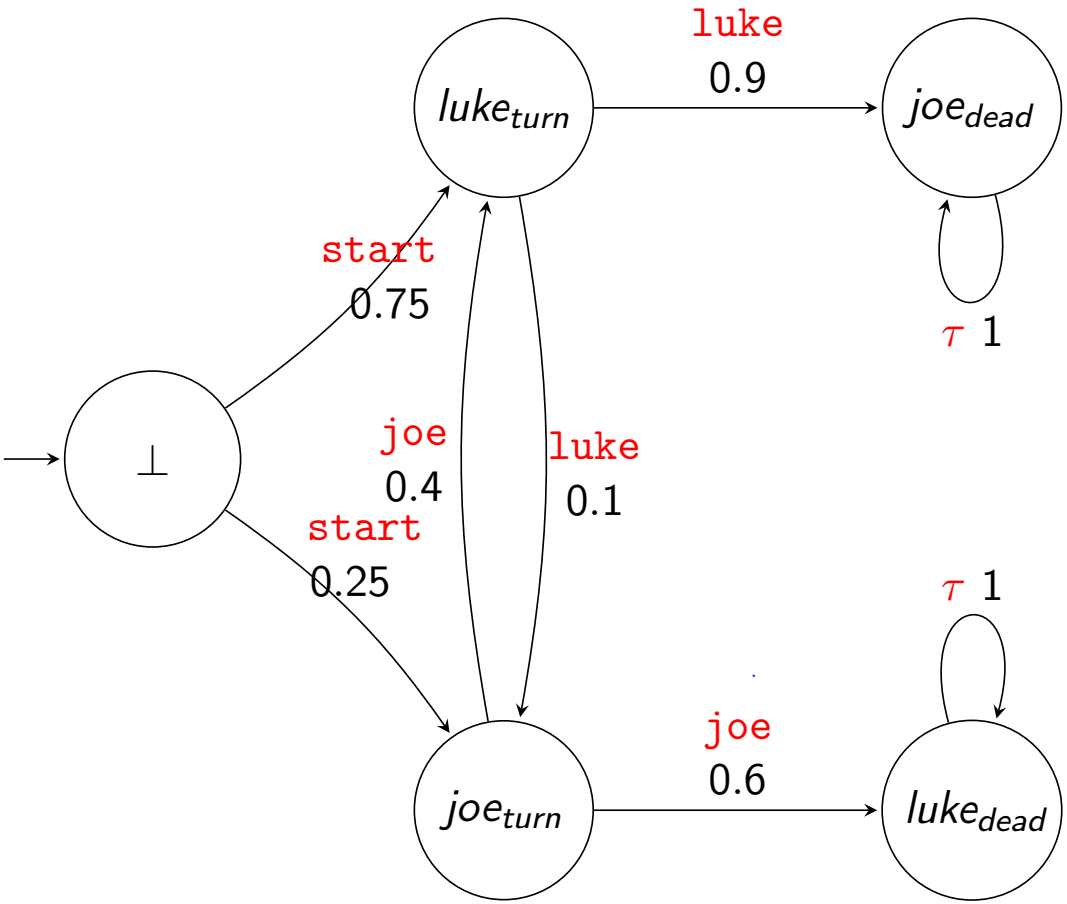


Example: Dueling cowboys

module Duel || Joe || Luke

$\ldots$[luke] turn = L $\wedge$ aliveJ $\wedge$ aliveL $\rightarrow$ 0.1: turn$'$ = J +
0.9: turn$'$ = J $\wedge$ aliveJ$'$ = 0;

endmodule



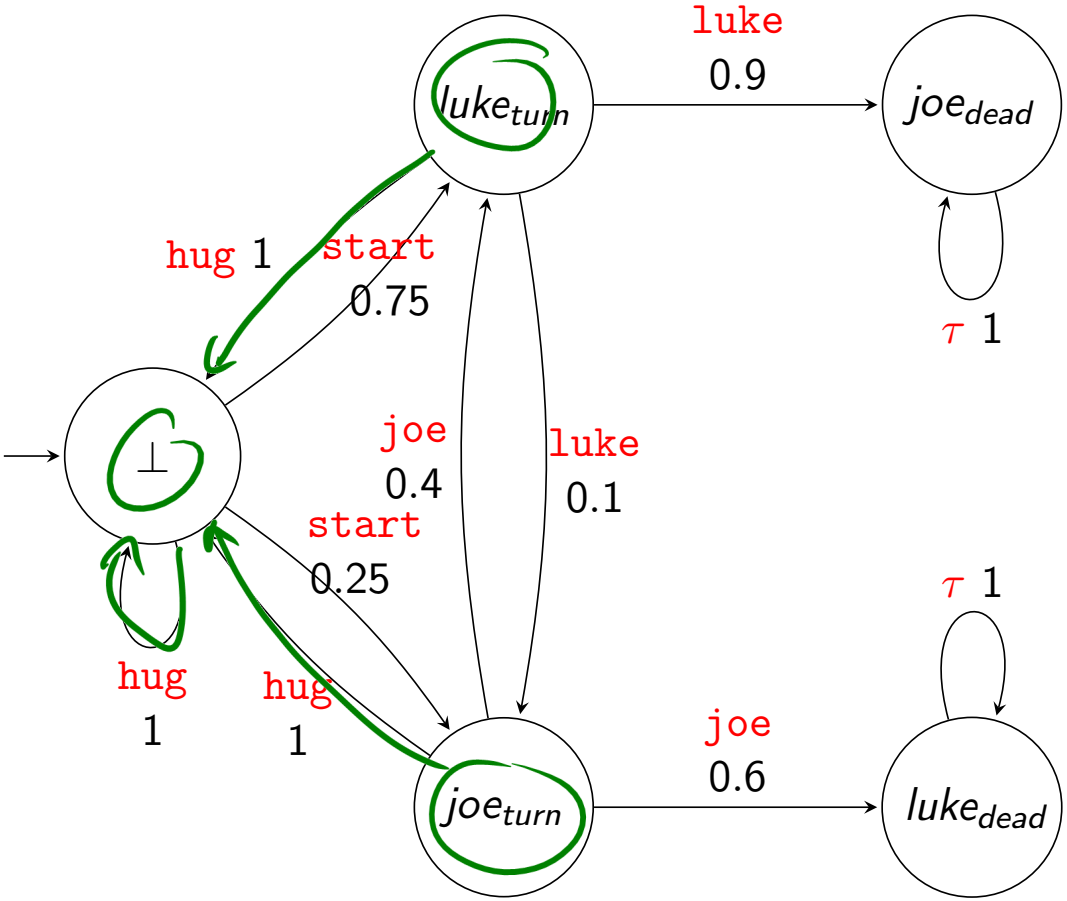
Example: Dueling cowboys

module Duel || Joe || Luke

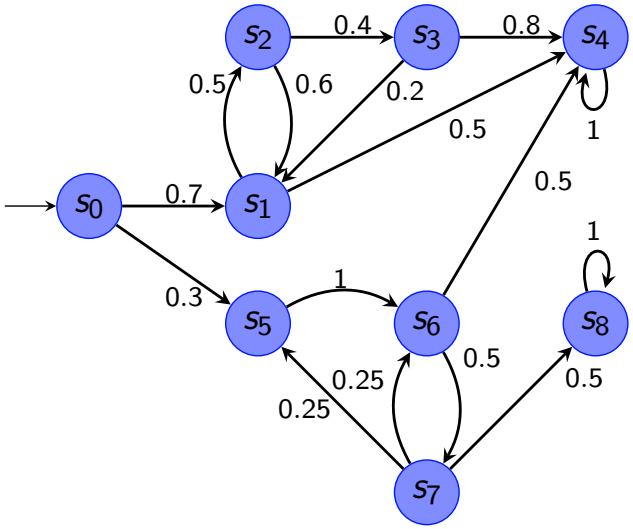
$\ldots$[luke] turn = L $\wedge$ aliveJ $\wedge$ aliveL $\rightarrow$ 0.1: turn$'$ = J +
0.9: turn$'$ = J $\wedge$ aliveJ$'$ = 0;

$\ldots$[hug] aliveJ $\wedge$ aliveL $\rightarrow$ 1.0: turn$'$ = $\bot$;

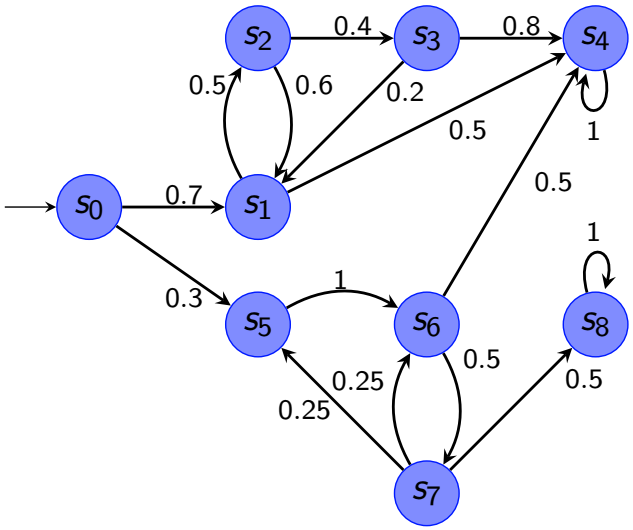
endmodule



Probability measure for DTMCs

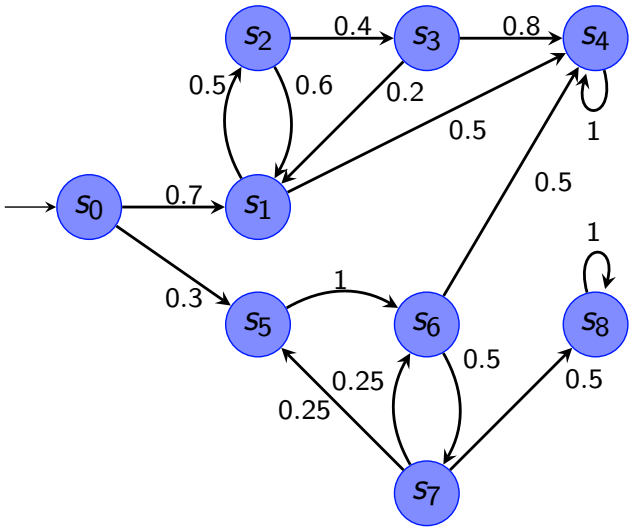


Probability measure for DTMCs



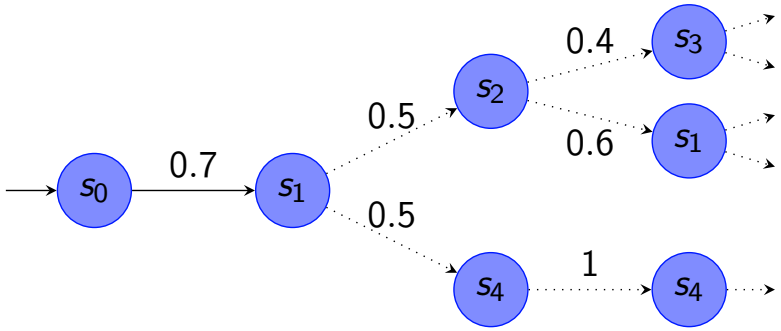
Paths are state sequences $s'_0 s'_1 \ldots$ with $P(s'_i, s'_{i+1}) > 0$

Probability measure for DTMCs

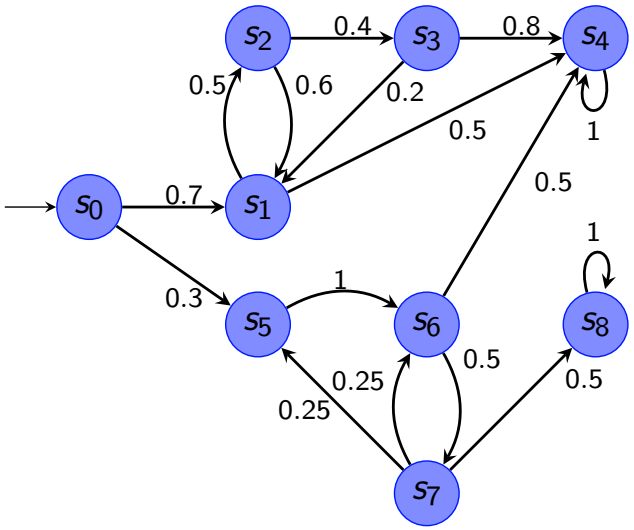


Paths are state sequences $s'_0 s'_1 \ldots$ with $P(s'_i, s'_{i+1}) > 0$

Cylinder set of finite path $s_0 s_1$:

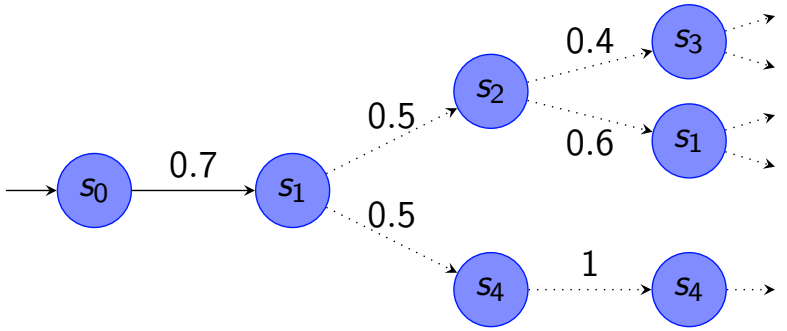


Probability measure for DTMCs

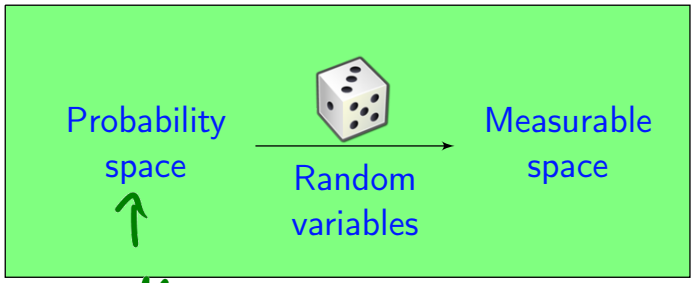


Paths are state sequences $s'_0 s'_1 \ldots$ with $P(s'_i, s'_{i+1}) > 0$

Cylinder set of finite path $s_0 s_1$:



Semantical basis: smallest σ-algebra over all cylinder sets with

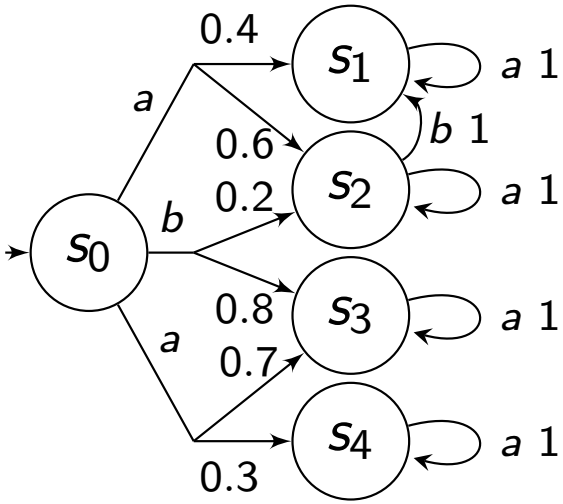


$$Pr_{s_0}\big(\mathrm{Cyl}(s_0 \ldots s_n)\big) = \prod_{i=0}^{n-1} P(s_i, s_{i+1})\,.$$

Probability measure under non-determinism

Atomic execution step:

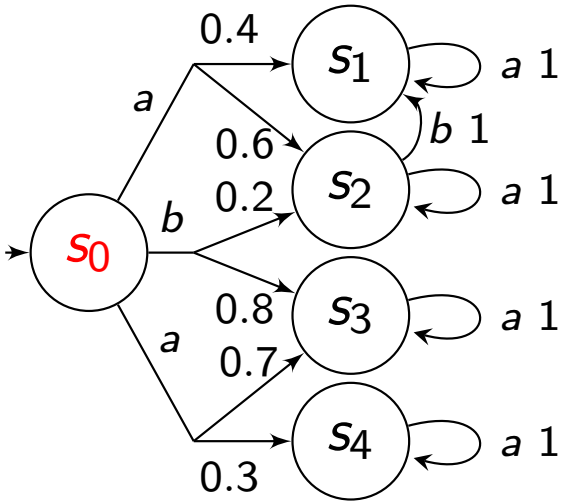
1. non-deterministic choice of an action-distribution pair
2. probabilistic choice of a transition



Probability measure under non-determinism

Atomic execution step:

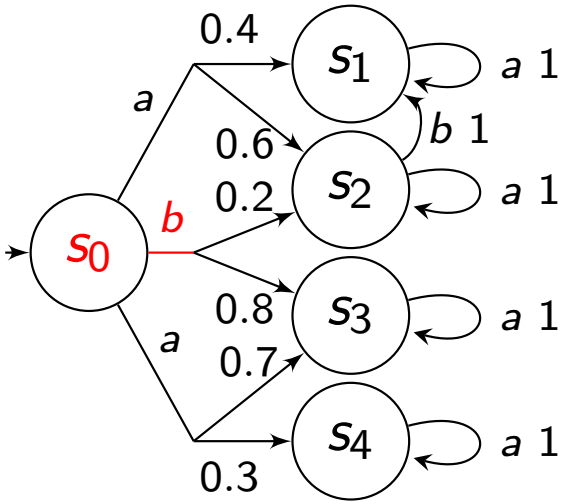
1. non-deterministic choice of an action-distribution pair
2. probabilistic choice of a transition



Probability measure under non-determinism

Atomic execution step:

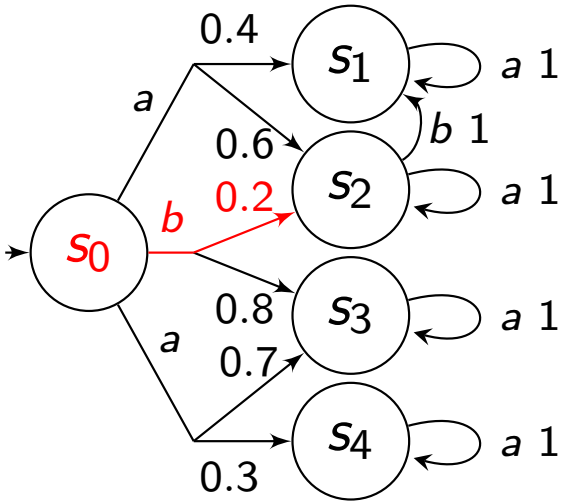
1. non-deterministic choice of an action-distribution pair
2. probabilistic choice of a transition



Probability measure under non-determinism

Atomic execution step:

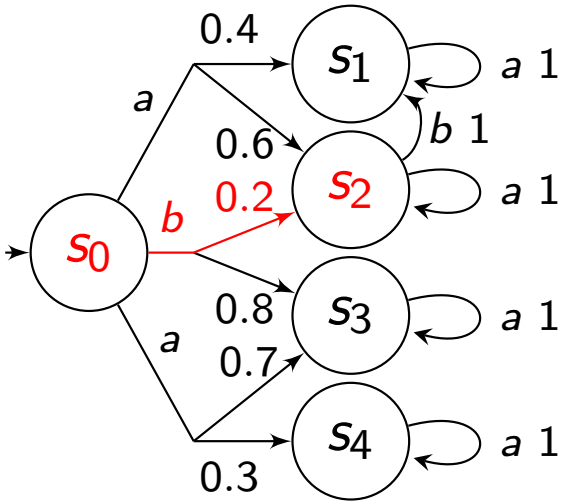
1. non-deterministic choice of an action-distribution pair
2. probabilistic choice of a transition



Probability measure under non-determinism

Atomic execution step:

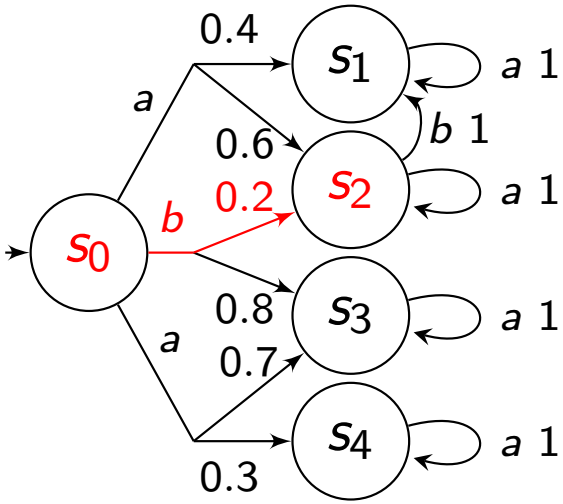
1. non-deterministic choice of an action-distribution pair
2. probabilistic choice of a transition



Probability measure under non-determinism

Atomic execution step:

1. non-deterministic choice of an action-distribution pair
2. probabilistic choice of a transition



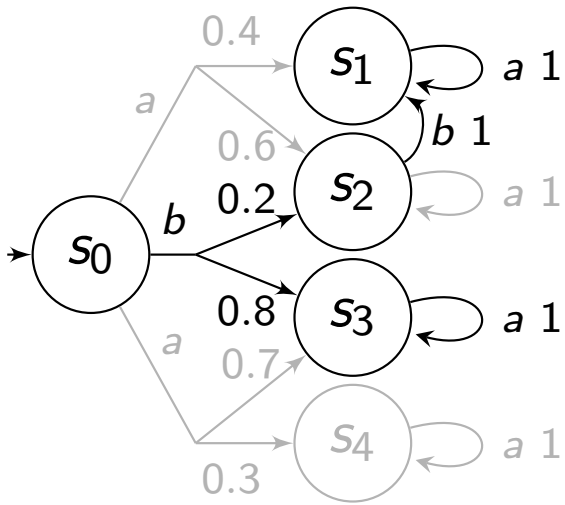
- A scheduler assigns to each finite path a distribution over the action-distribution pairs possible in the last state.
- Deterministic scheduler: single action-distribution pair
- Memoryless scheduler: history-independent

Each scheduler for an MDP/PA induces a DTMC.

Probability measure under non-determinism

Atomic execution step:

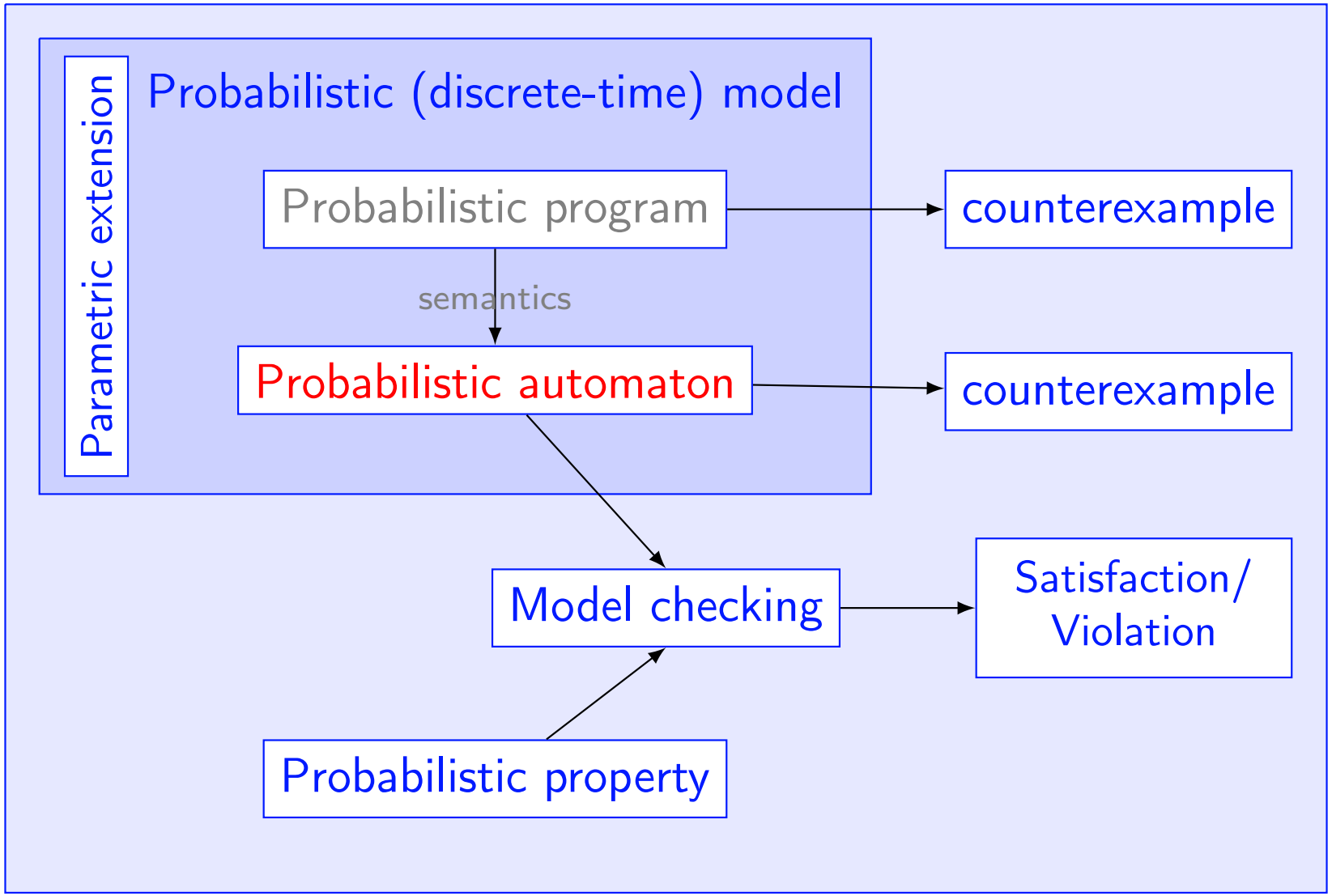
1. non-deterministic choice of an action-distribution pair
2. probabilistic choice of a transition



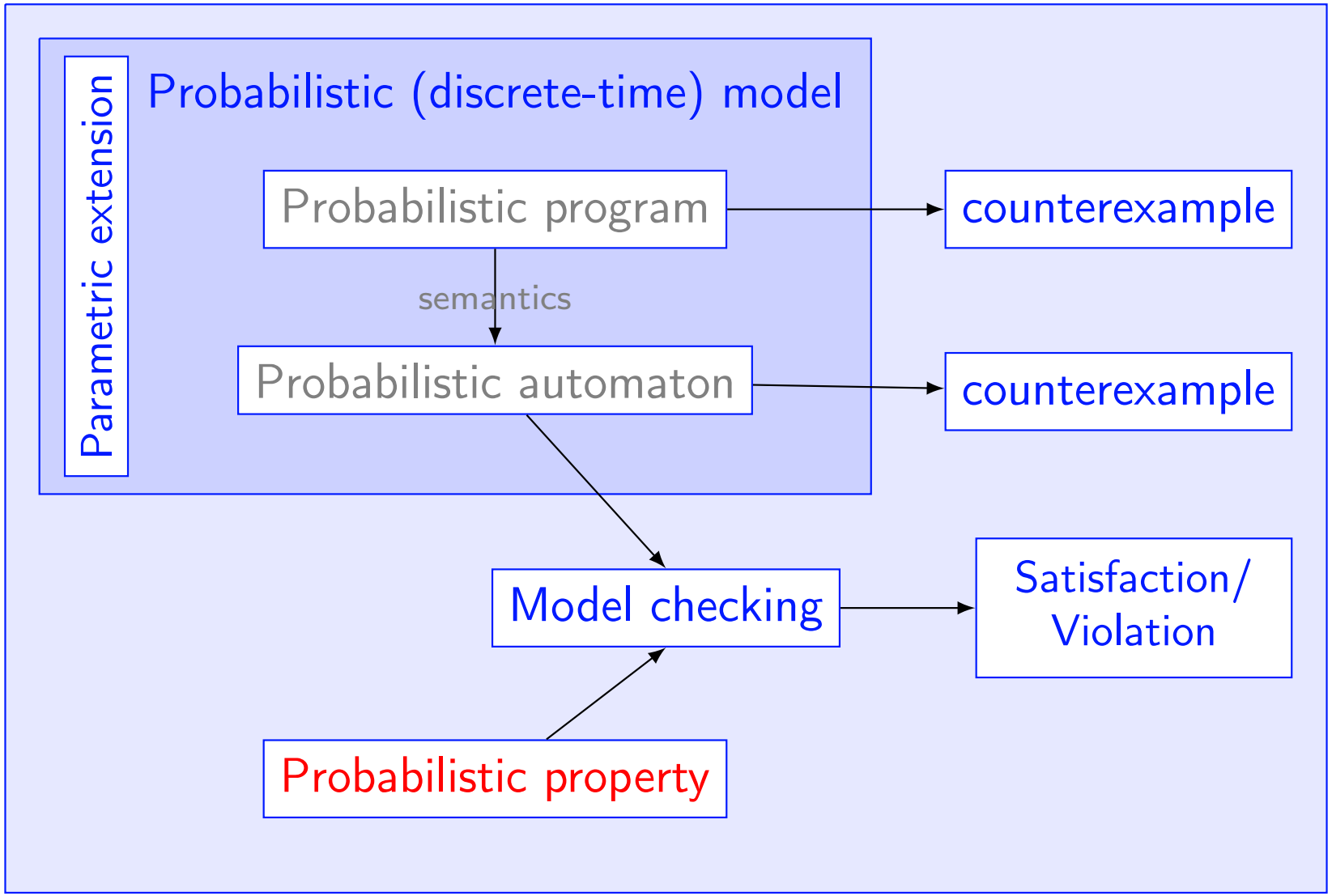
- A scheduler assigns to each finite path a distribution over the action-distribution pairs possible in the last state.
- Deterministic scheduler: single action-distribution pair
- Memoryless scheduler: history-independent

Each scheduler for an MDP/PA induces a DTMC.

Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Contents

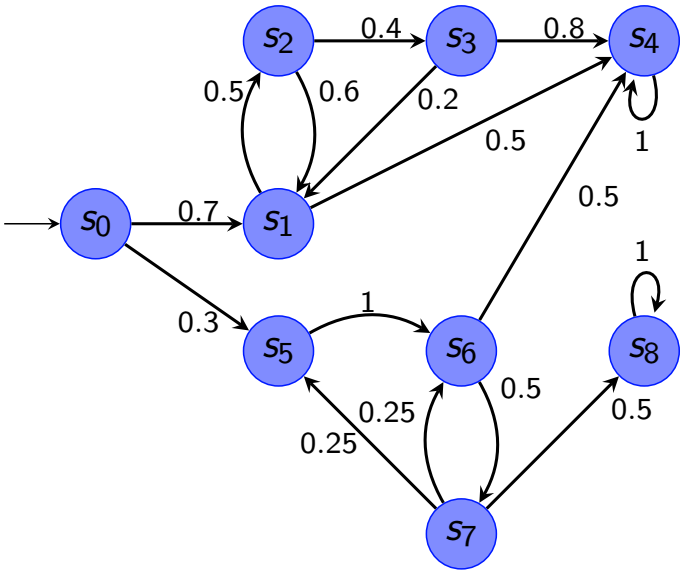

Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Probabilistic reachability properties of DTMCs

- Many interesting properties can be reduced to probabilistic reachability properties.
- E.g., $\mathcal{D}, s_{\text{init}} \models \mathbb{P}_{<0.01}(\Diamond t)$

  "The probability to reach t from s_{init} in $\mathcal{D}$ is less than 0.01."

$$Pr_{s_{\text{init}}}(\Diamond t) = \sum_{\text{finite paths } \underbrace{s_{\text{init}} \ldots t}_{\neg t}} Pr_{s_{\text{init}}}(s_{\text{init}} \ldots t)\,.$$



$$\begin{aligned} Pr_{s_0}(\Diamond s_4) = \; & Pr_{s_0}(s_0 s_1 s_4) + \\ & Pr_{s_0}(s_0 s_1 s_2 s_1 s_4) + \\ & Pr_{s_0}(s_0 s_5 s_6 s_4) + \ldots \end{aligned}$$

Reachability properties of MDPs/PAs/probabilistic programs

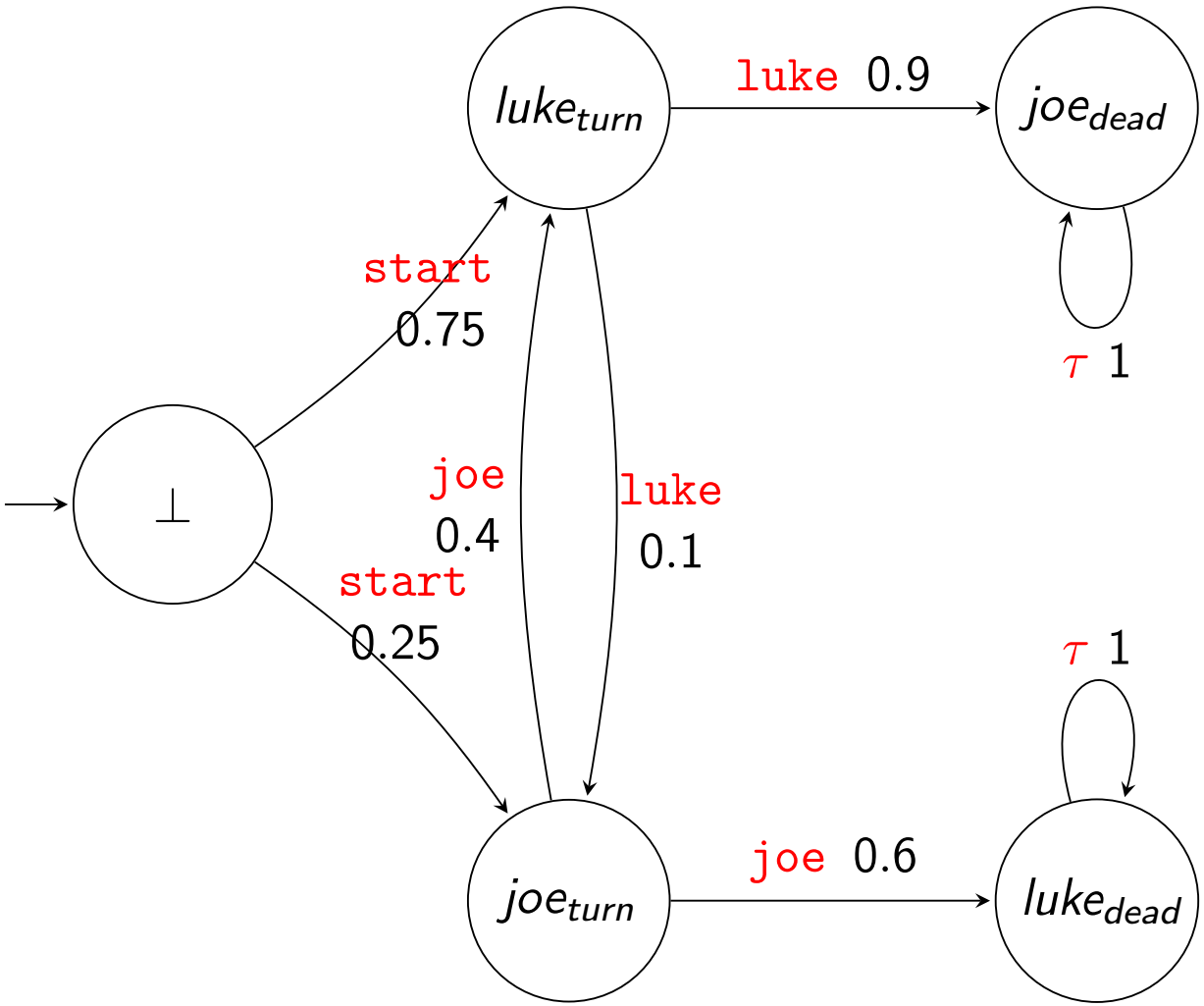
Semantics of reachability properties for MDPs/PAs:

- The property must be satisfied by the induced DTMCs of all schedulers.
- The lowest/highest probabilities can be always reached by a memoryless deterministic scheduler.
- Thus the property must be satisfied by the DTMC induced by a minimal/maximal memoryless deterministic scheduler.

Semantics of reachability properties for probabilistic programs:

- Defined by the induced DTMC/MDP/PA.

Example: Dueling cowboys

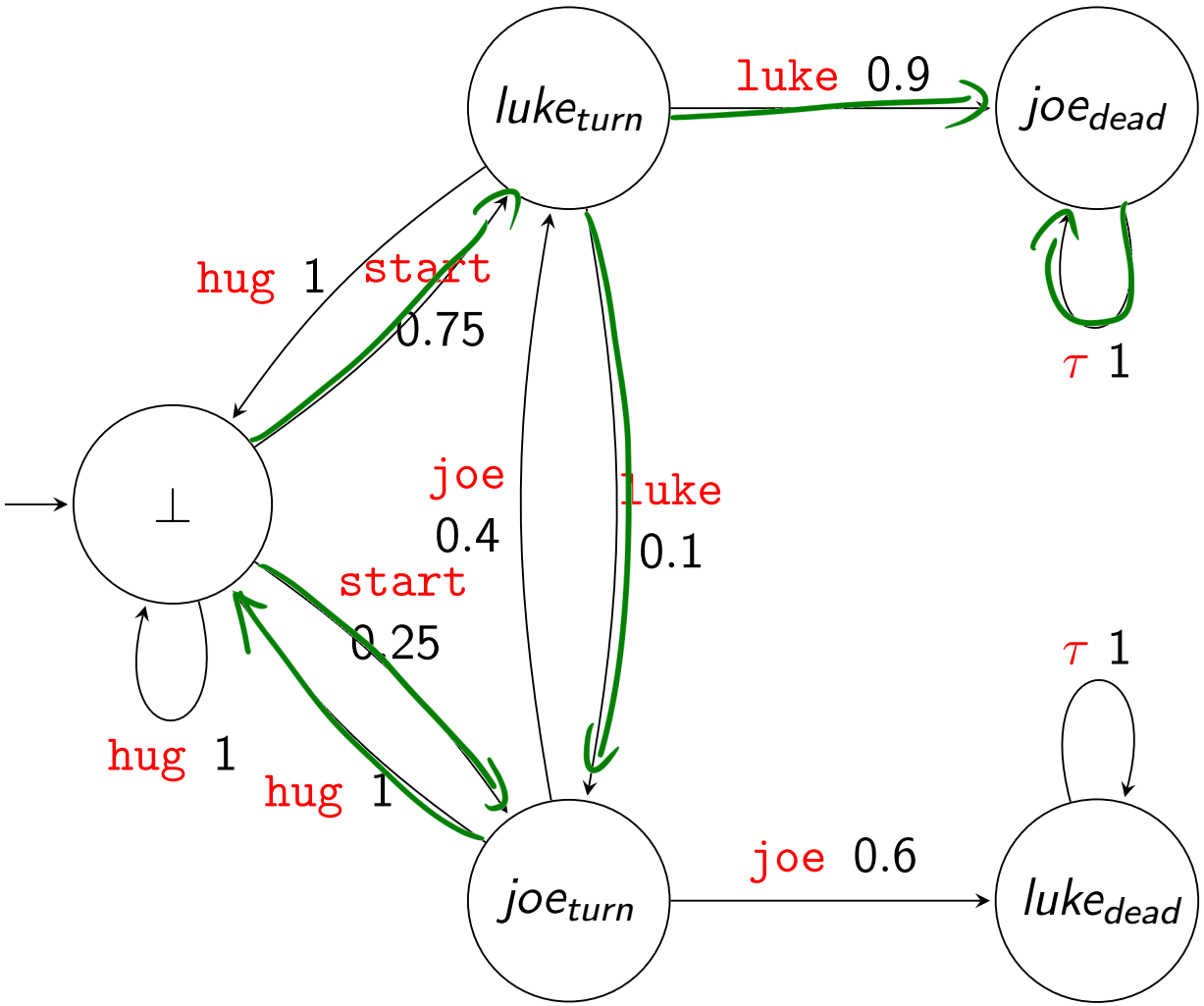


$Pr_{start}(\Diamond joe_{dead}) = 0.796875$

$\mathbb{P}_{\leq 0.8}(\Diamond joe_{dead})$ is satisfied

$Pr_{start}(\Diamond luke_{dead}) = 0.203125$

Example: Dueling cowboys

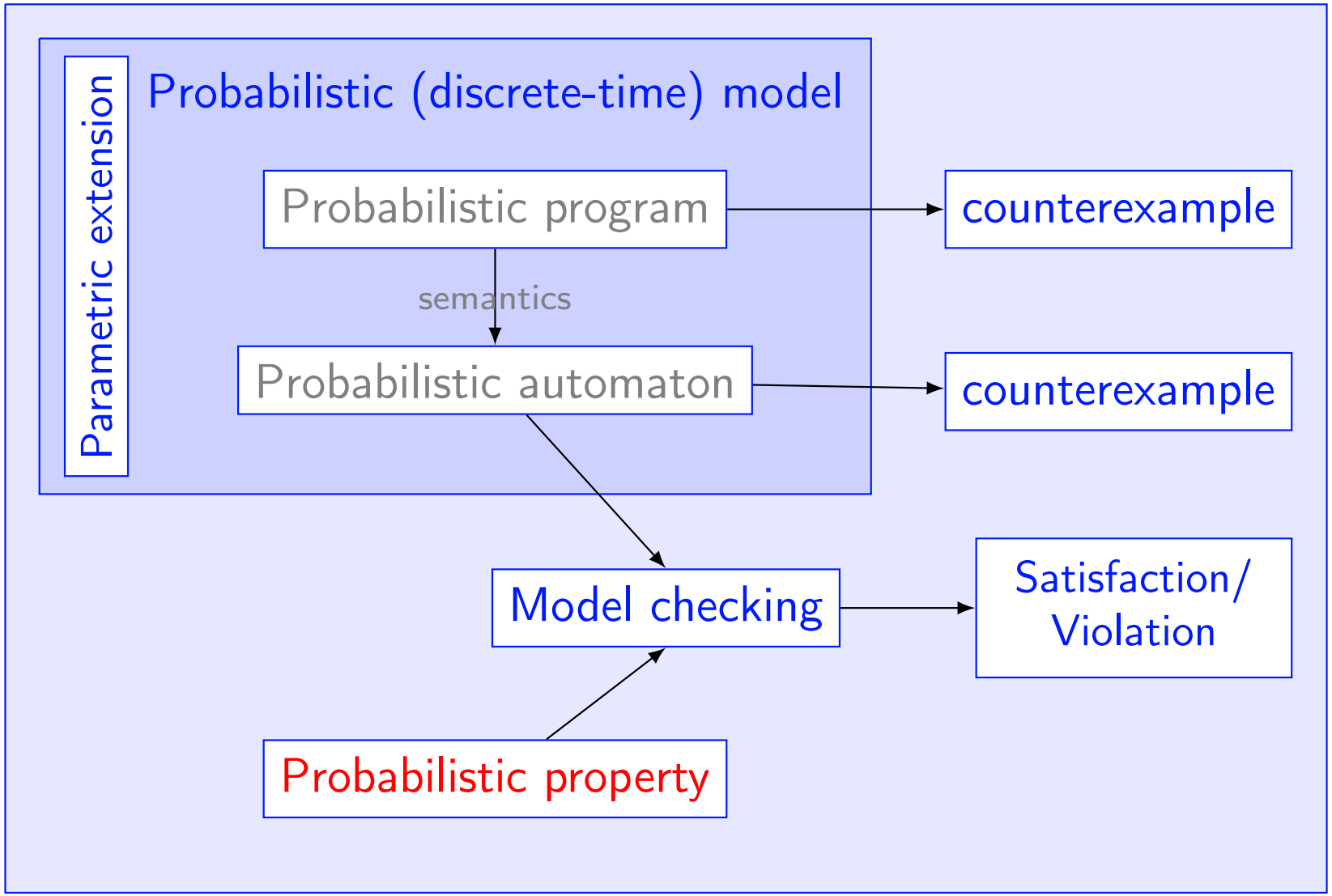


$0 \leq Pr_{start}(\Diamond joe_{dead}) \leq 1$

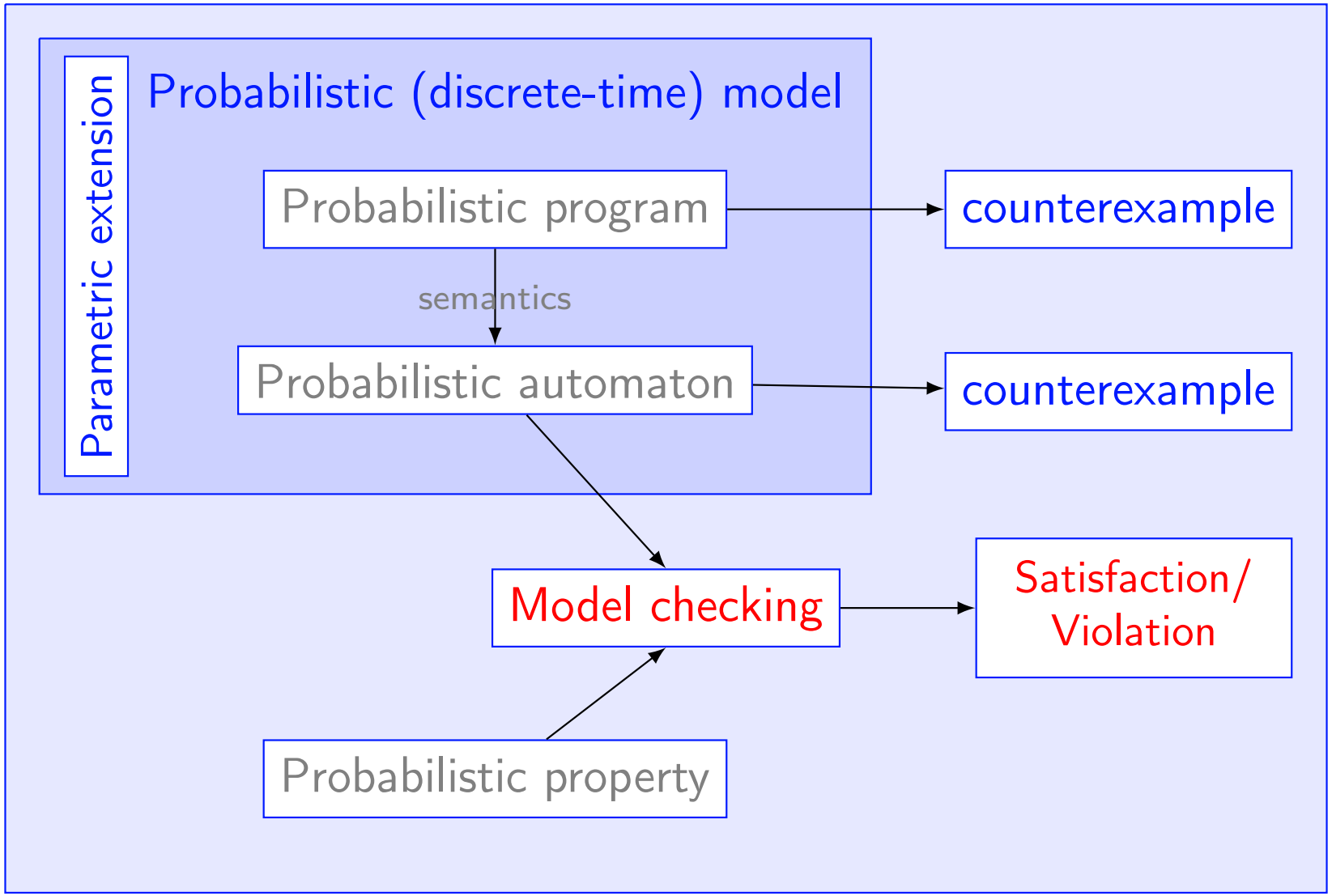
$\mathbb{P}_{\leq 0.8}(\Diamond joe_{dead})$ is violated

$0 \leq Pr_{start}(\Diamond luke_{dead}) \leq 1$

Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Other Tools

Below is a brief summary of some other available tools with support for probabilistic model checking, including thos

---

Probabilistic Model Checkers

Model checkers for discrete-/continuous-time Markov chains (DTMCs/CTMCs) and extensions of:

- MRMC (formerly ETMCC) - explicit-state (and approximate) model checking for DTMCs, CTMCs and CTMDPs (w
- The PEPA Eclipse Plug-in project - CSL model checking (plus steady-state/ODE analysis and abstraction techniq
- PARAM - *parametric* probabilistic model checking of DTMCs. See [HHWZ10].
- INFAMY - model checking of *infinite-state* CTMCs. See [HHWZ09b].
- CASPA - symbolic (MTBDD-based) model checking of (extended) stochastic labelled transition systems against

Model checkers for Markov decision processes (MDPs):

- LiQuor - explicit-state model checking of MDPs, expressed in Probmela (a probabilistic extension of SPIN's Pro
- ProbDiVinE - parallel and/or distributed LTL model checking of MDPs. See [BBC+08].
- PASS - *abstraction refinement* of probabilistic models. See [HHWZ10b].

Tools Connecting to or Integrating PRISM

- Infobiotics Workbench - tool for computational systems and synthetic biology. Integrates PRISM.
- kamiframework - run-time modelling and anlaysis of software designs. Integrates PRISM.
- Snoopy - Petri net-based tool for analysis of biomolecular networks. Exports to PRISM. See [RMH10].
- Henshin Statespace Explorer - part of the Eclipse Modeling Framework Project.
- PSPWizard - generates temporal logic properties from patterns for several tools, including PRISM. See [LMG11]

See also the connections listed here.

Other related tools

- APNN-Toolbox - a Petri Net tool-set, that includes support for CTMCs and CSL model checking. See [BK99].
- CADP - "Construction and Analysis of Distributed Processes" toolset, with some support for probabilistic model
- GreatSPN analysis of Generalised Stochastic Petri Nets (GSPNs) and extensions.
- GRIP - language-level symmetry reduction for PRISM models.
- ipc/DNAmaca - tool extending PEPA's modelling capability with additional performability measures.
- Marcie - tool for analyis of Generalised Stochastic Petri Nets (GSPNs) with extended arcs.
- Möbius - multi-formalism modelling and analysis toolset for stochastic systems.
- MOTOR - tool environment for the MODEST language. See [BHK07].
- SMART - explicit-state/symbolic numerical solution (and CTL model checking) of Markov chains. See [CJMS06].
- COMICS - model checking and generation of counterexamples for DTMCs. See [JAV+12].
- DiPro - a directed probabilistic counterexample generation tool. See [ALLS11].
- PAT - a model checking framework which includes support for probabilistic models. See [SLDP09]

Model checking reachability properties of DTMCs

Model checking reachability properties of DTMCs

- S set of states, T set of absorbing target states
- States from which T is not reachable are irrelevant and get deleted.

Model checking reachability properties of DTMCs

- S set of states, T set of absorbing target states
- States from which T is not reachable are irrelevant and get deleted.

Probabilities of reaching T

$$p_s = \begin{cases} 1 & \text{if } s \in T, \\ \sum_{s' \in S} P(s, s') \cdot p_{s'} & \text{otherwise.} \end{cases} \qquad \textit{for all } s \in S$$

Model checking reachability properties of DTMCs

- S set of states, T set of absorbing target states
- States from which T is not reachable are irrelevant and get deleted.

Probabilities of reaching T

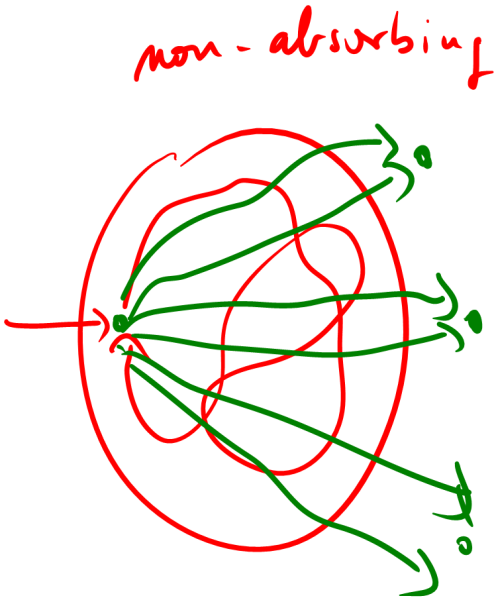
$$p_s = \begin{cases} 1 & \text{if } s \in T, \\ \sum_{s' \in S} P(s,s') \cdot p_{s'} & \text{otherwise.} \end{cases} \qquad \textit{for all } s \in S$$

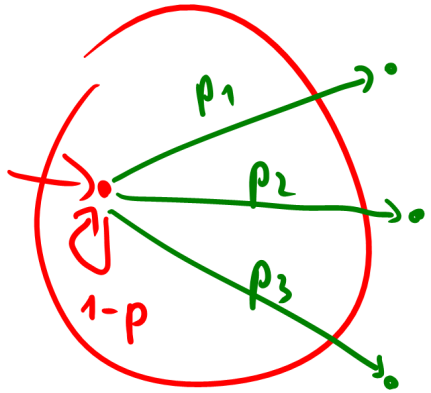
- Option 1: Solve the linear equation system
- Option 2: Iterative approach to approximate (explicitly or symbolically)

$$\lim_{i \to \infty} P^i \cdot \underbrace{\underline{T}}_{\textit{characteristic vector for } T}$$

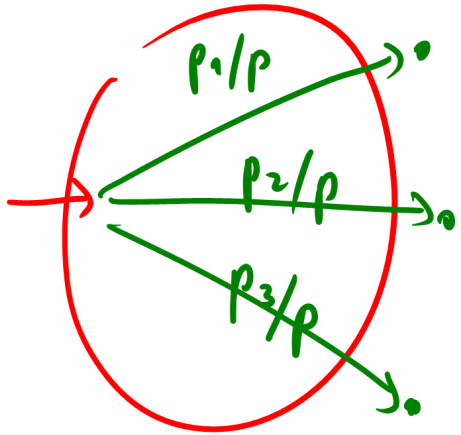
- Option 3: SCC-based model checking [QEST'10]

Option 3: SCC-based model checking for DTMCs



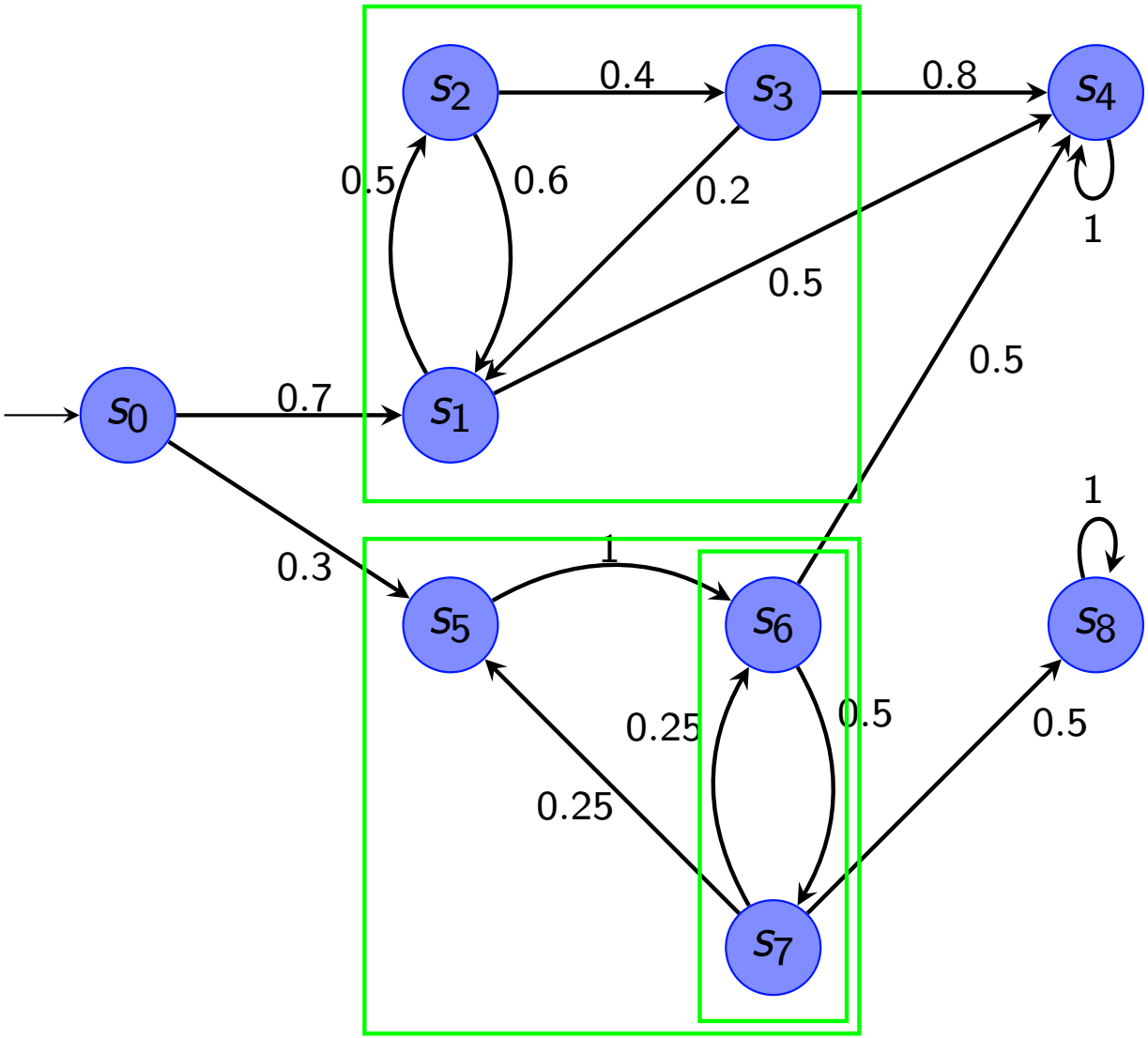


$$p = p_1 + p_2 + p_3$$

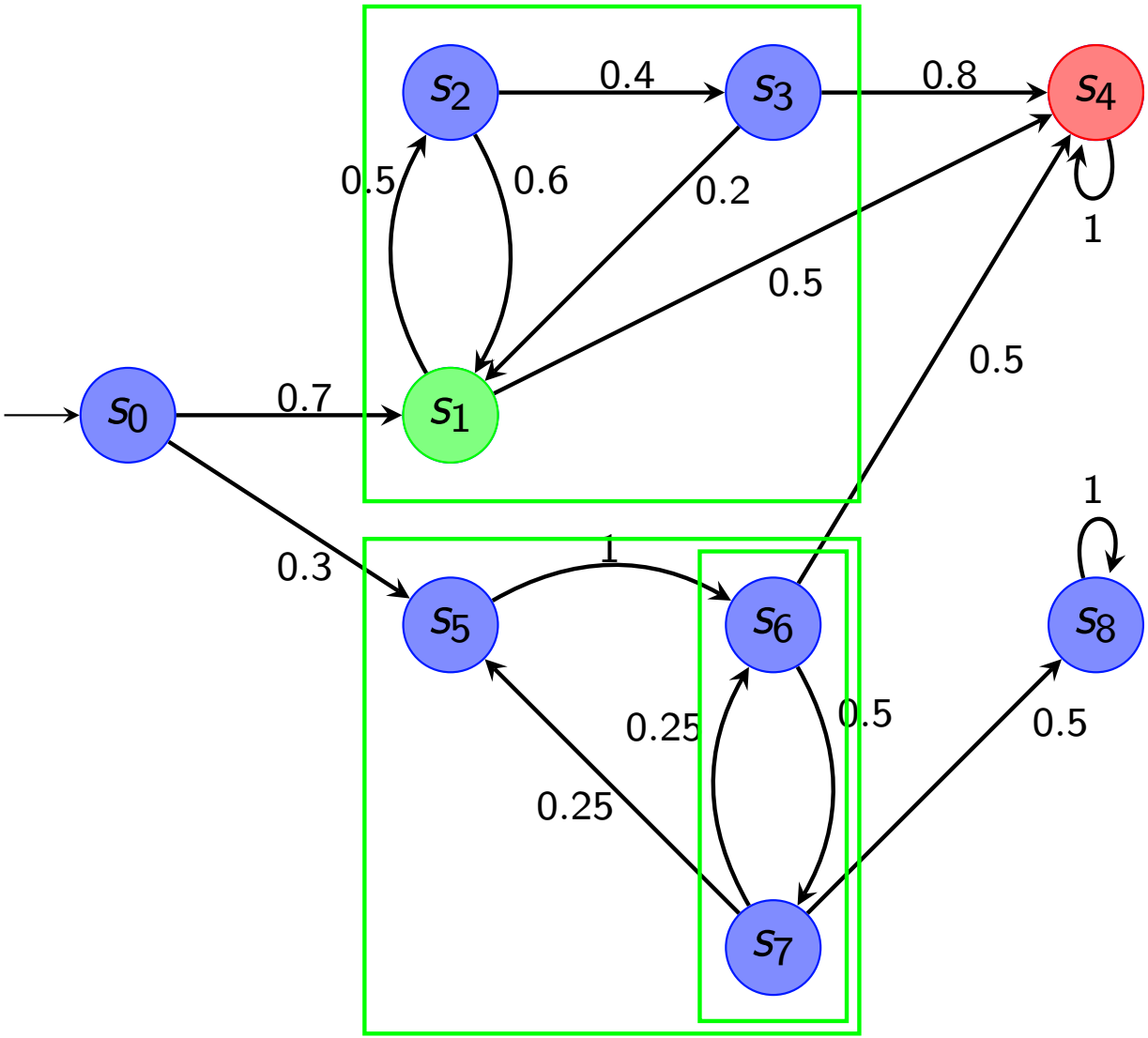


$$\sum_i (1-p)^i = \frac{1}{p}$$

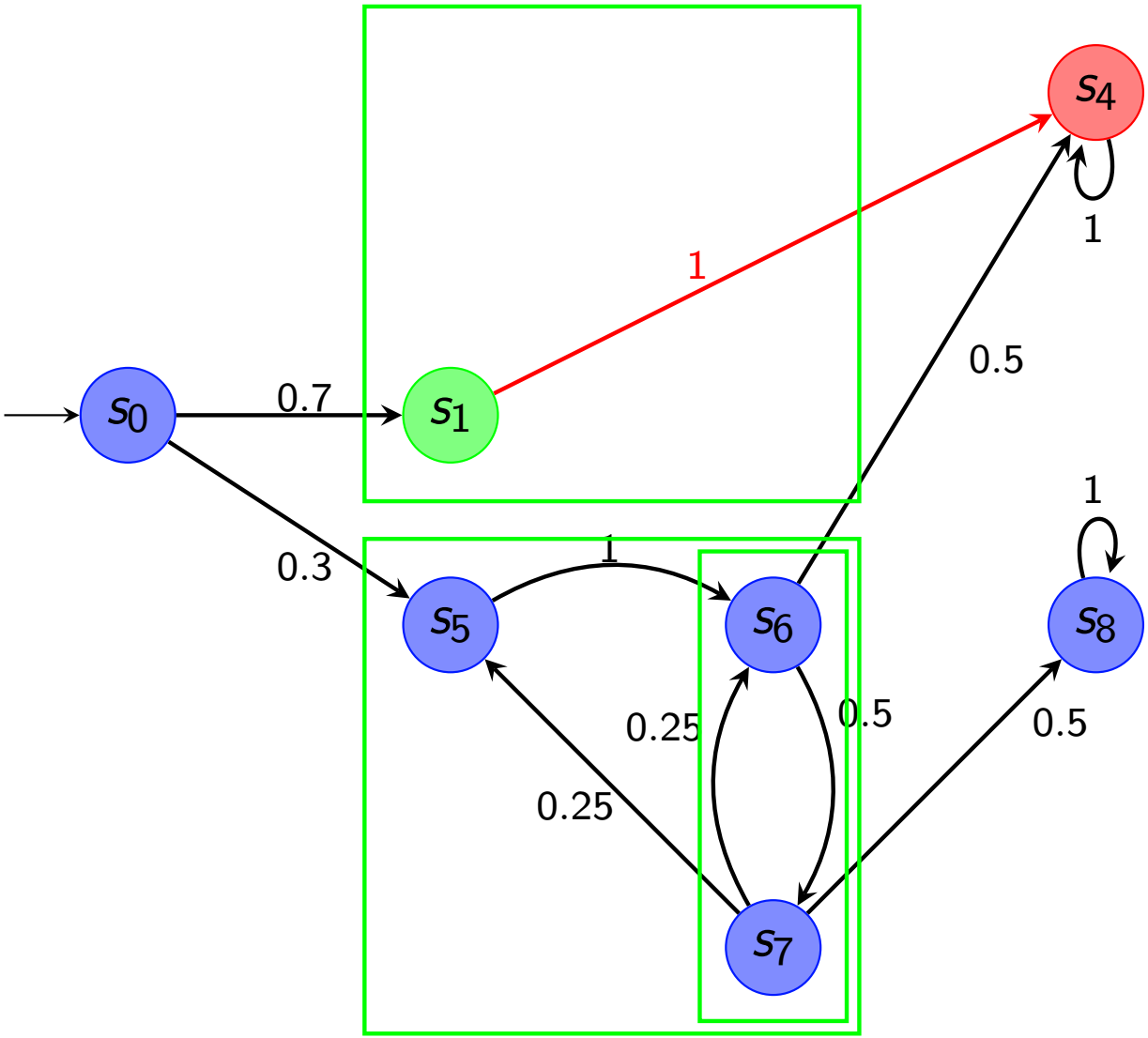
Option 3: SCC-based model checking for DTMCs



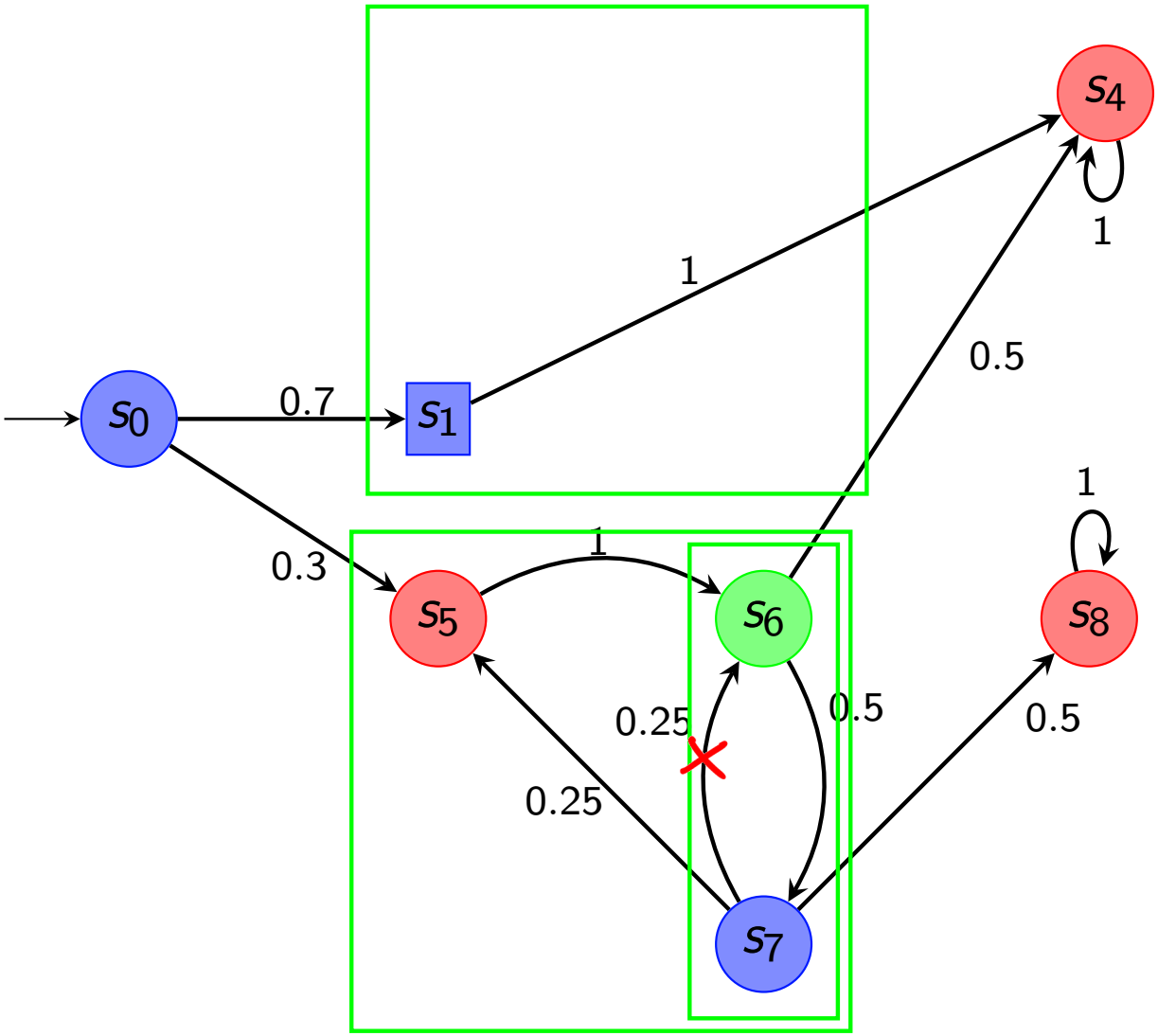
Option 3: SCC-based model checking for DTMCs



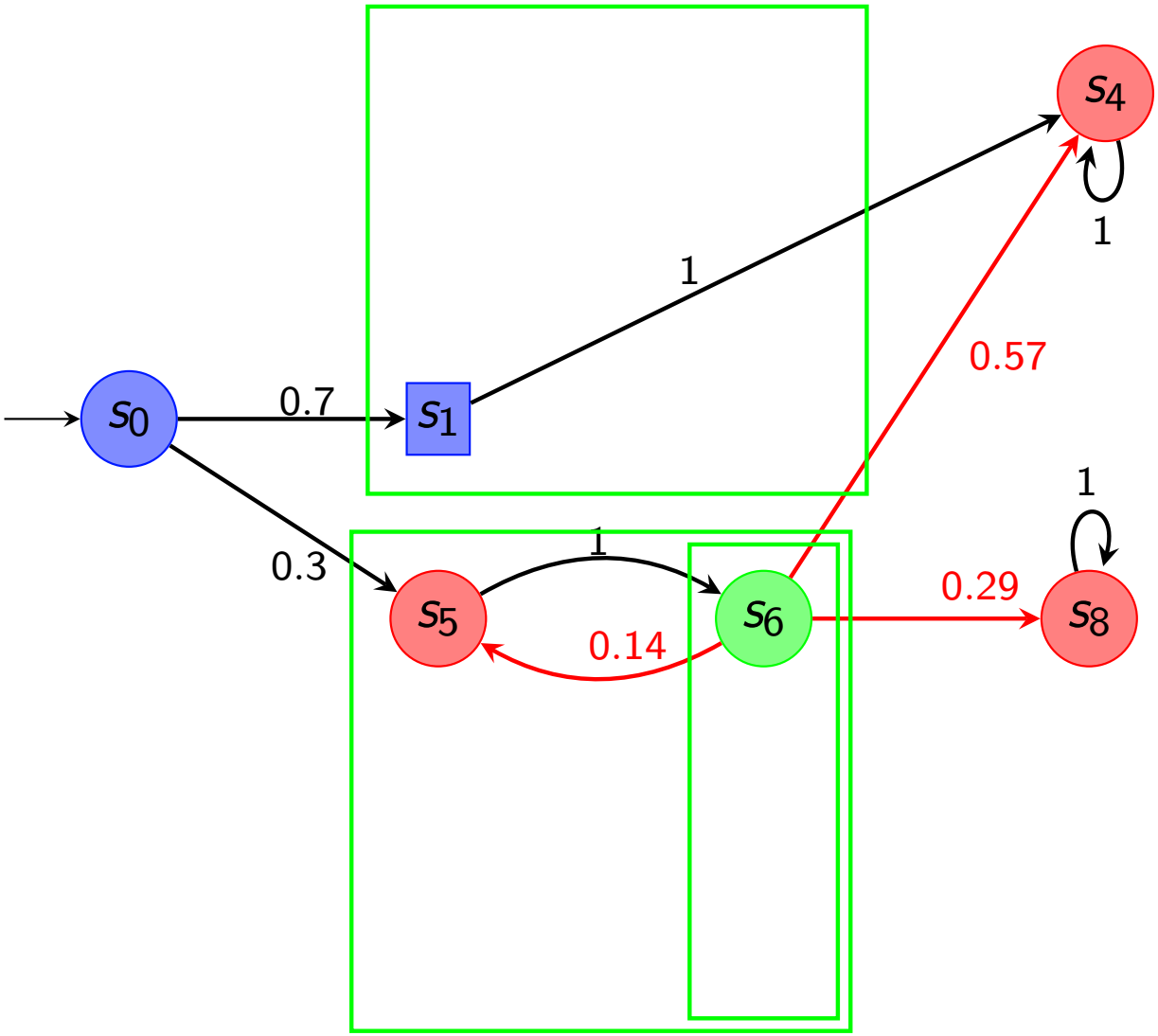
Option 3: SCC-based model checking for DTMCs



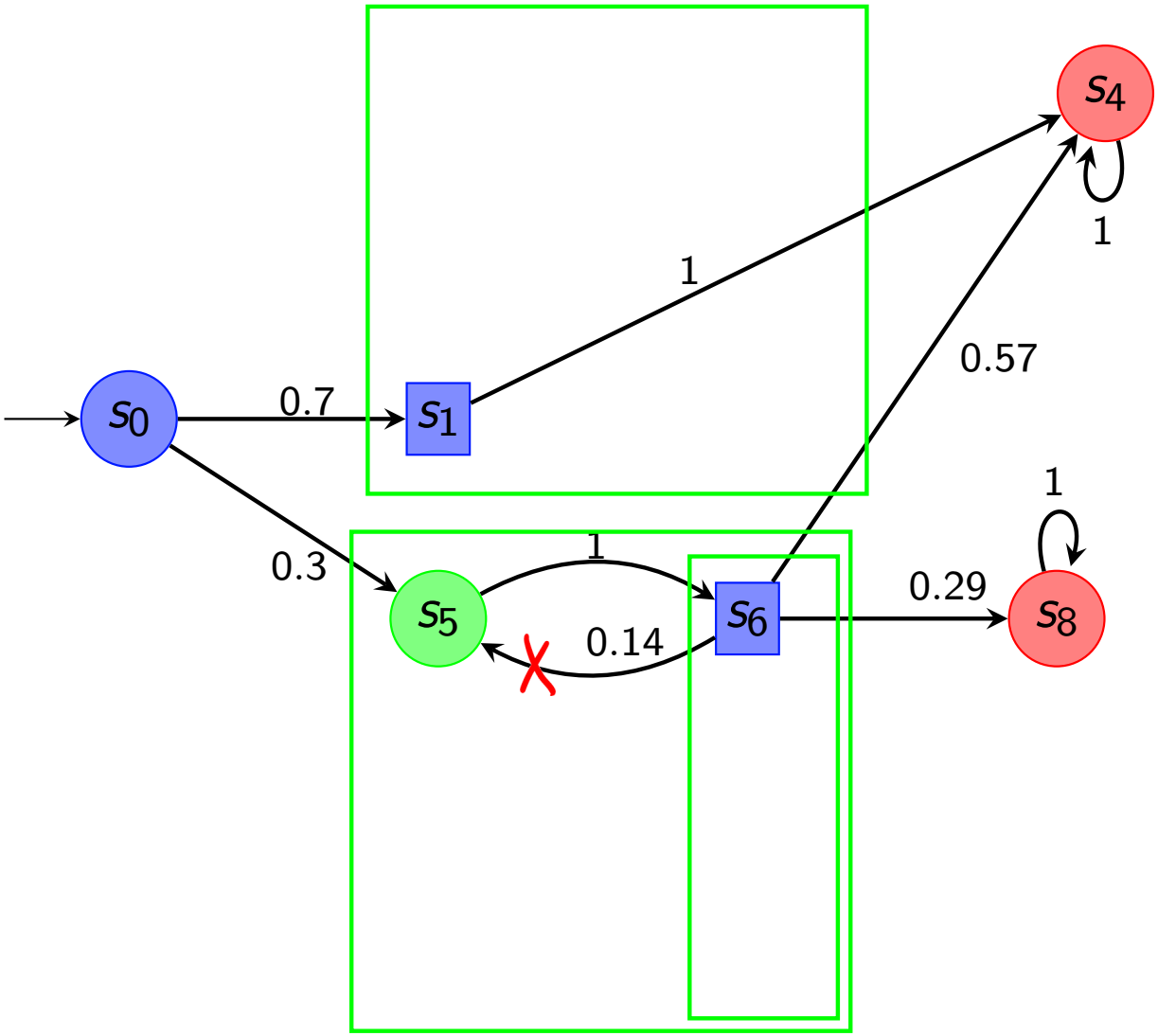
Option 3: SCC-based model checking for DTMCs



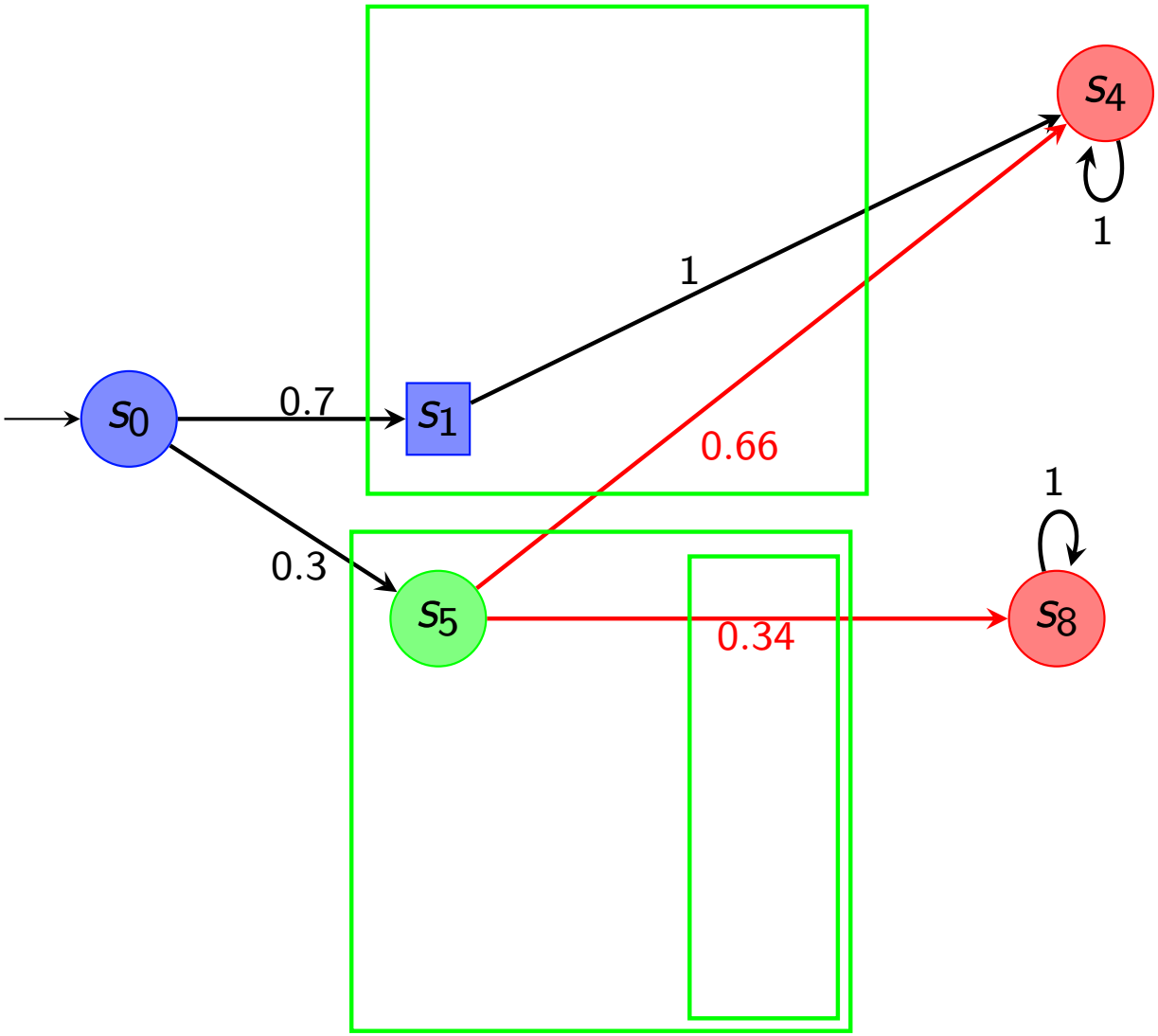
Option 3: SCC-based model checking for DTMCs



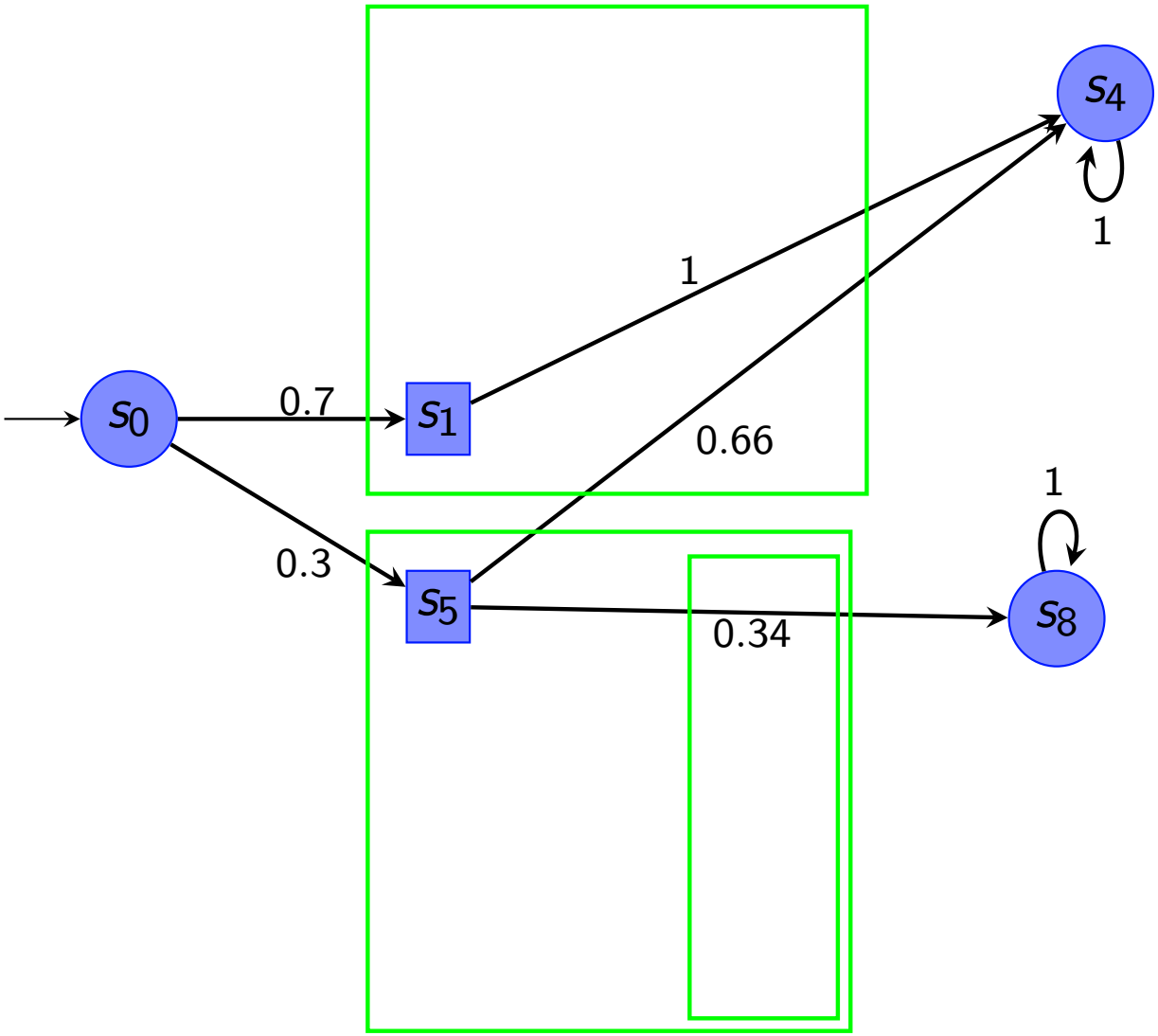
Option 3: SCC-based model checking for DTMCs



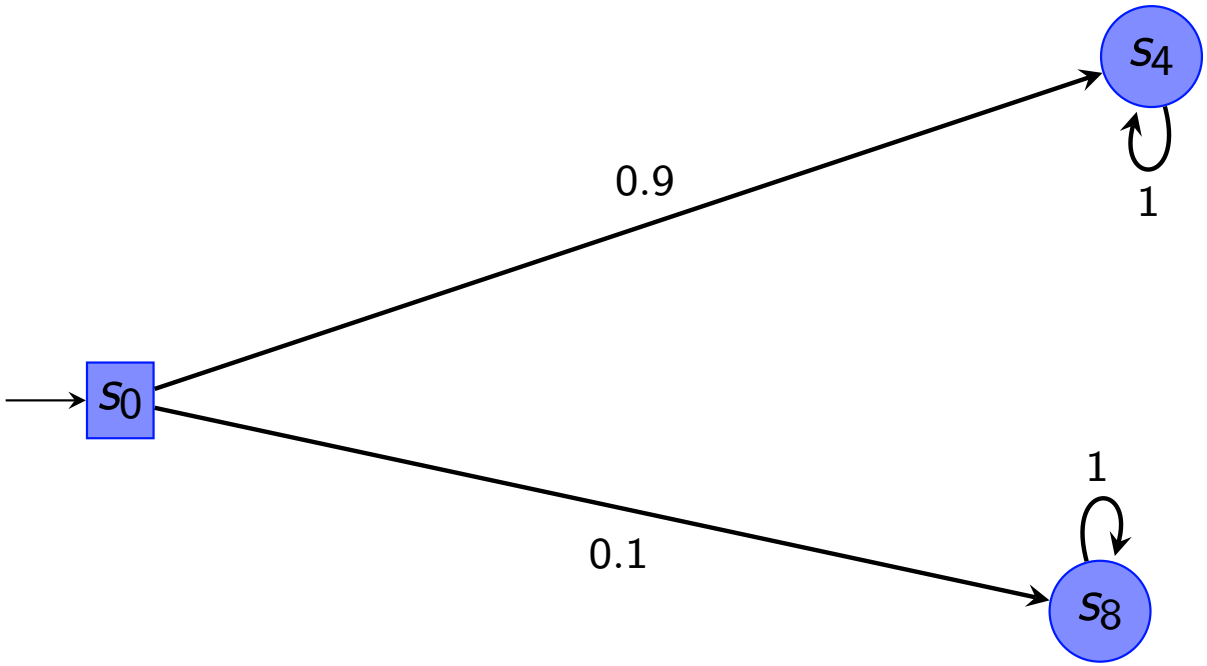
Option 3: SCC-based model checking for DTMCs



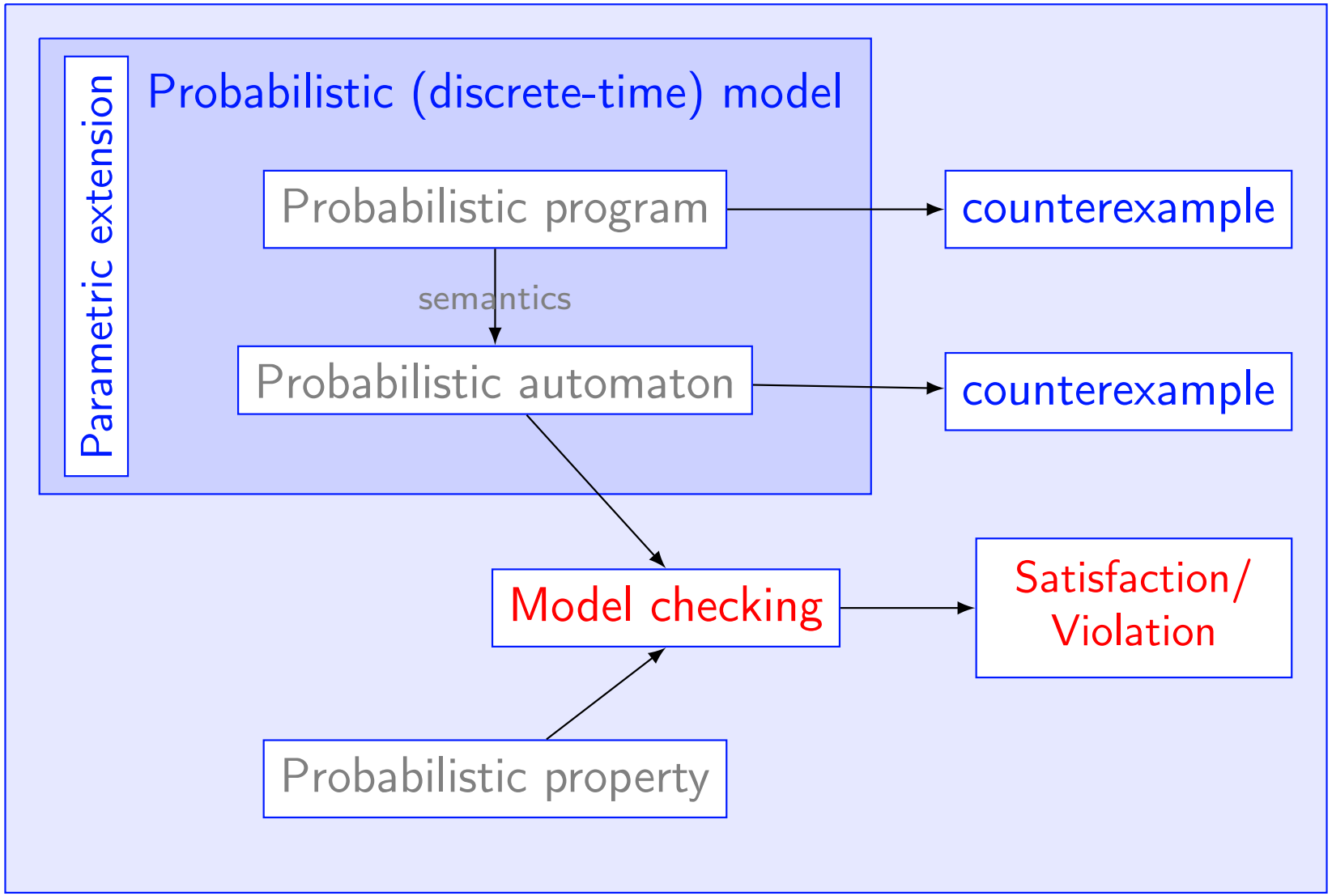
Option 3: SCC-based model checking for DTMCs



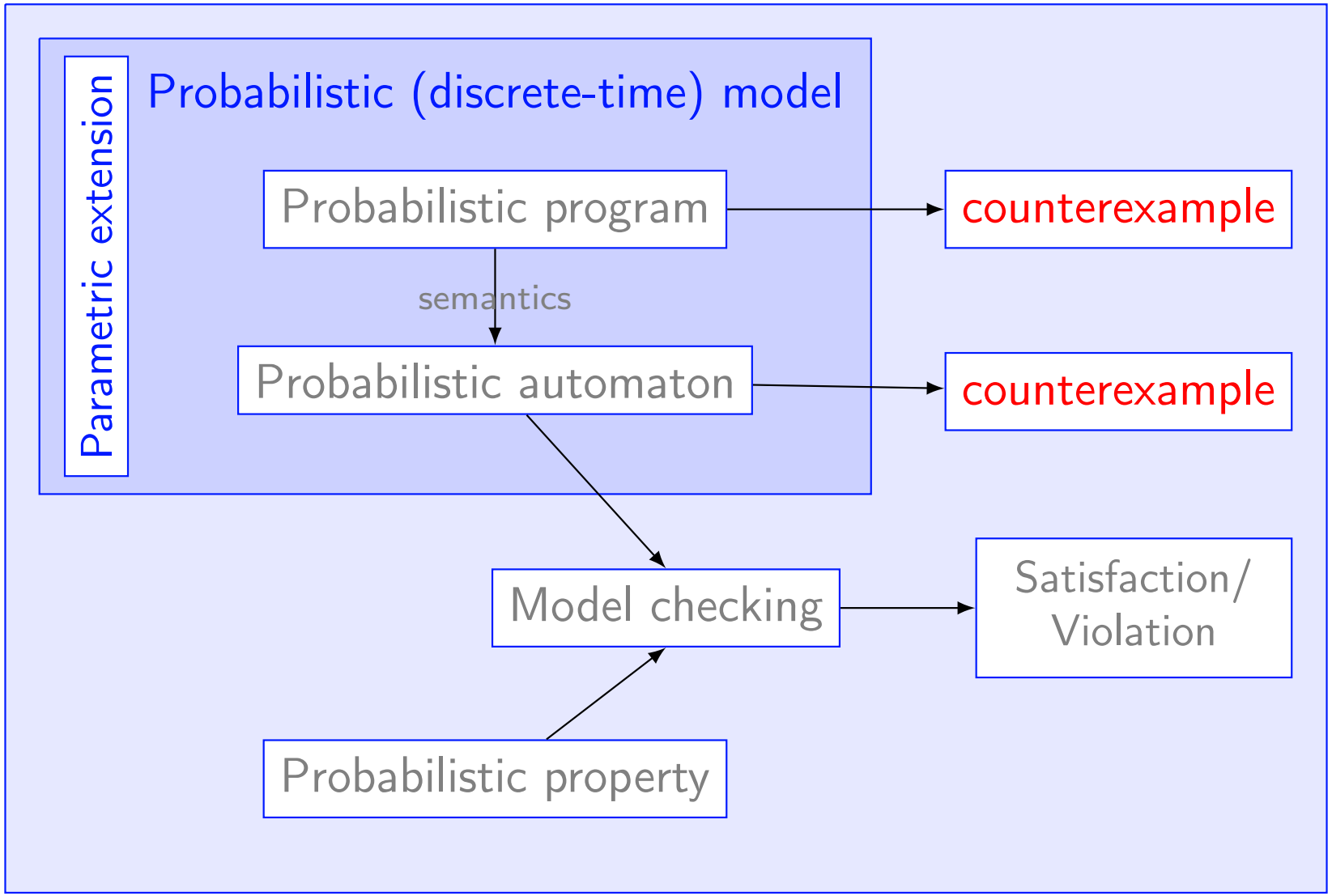
Option 3: SCC-based model checking for DTMCs



Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

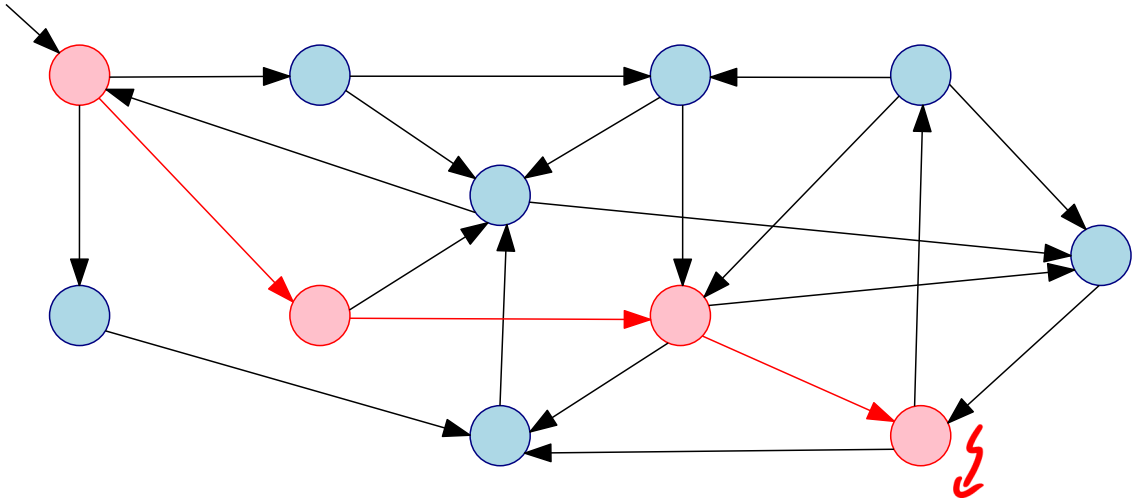
Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Counterexamples for DTMCs

Digital systems:

- Safety property: $\mathcal{AG}\, safe$
- Violation: $\mathcal{EF}\, \neg safe$
- Counterexample: Path from the initial state to a $\neg safe$ state



Counterexamples for DTMCs

Digital systems:

- Safety property: $\mathcal{AG}\, safe$
- Violation: $\mathcal{EF}\, \neg safe$
- Counterexample: Path from the initial state to a $\neg safe$ state

Probabilistic systems:

- Safety property: $\mathbb{P}_{\geq \lambda}(\mathcal{G}\, safe)$
- Violation: $\mathbb{P}_{>1-\lambda}(\mathcal{F}\, \neg safe)$
- Counterexample: Set C of unsafe paths with $Pr(C) > 1 - \lambda$
- Not computed as a by-product of model checking

Counterexamples for DTMCs

Digital systems:

- Safety property: $\mathcal{AG}\, safe$
- Violation: $\mathcal{EF}\, \neg safe$
- Counterexample: Path from the initial state to a $\neg safe$ state

Probabilistic systems:

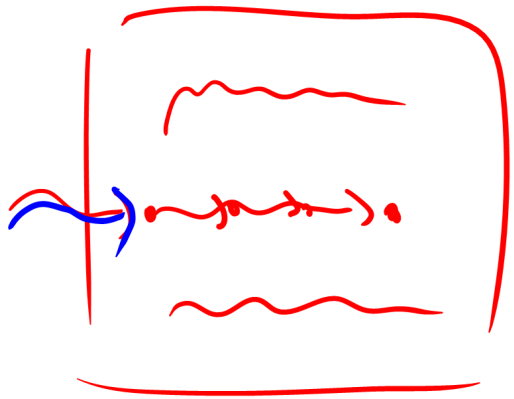
- Safety property: $\mathbb{P}_{\geq\lambda}(\mathcal{G}\, safe)$
- Violation: $\mathbb{P}_{>1-\lambda}(\mathcal{F}\, \neg safe)$
- Counterexample: Set C of unsafe paths with $Pr(C) > 1 - \lambda$
- Not computed as a by-product of model checking

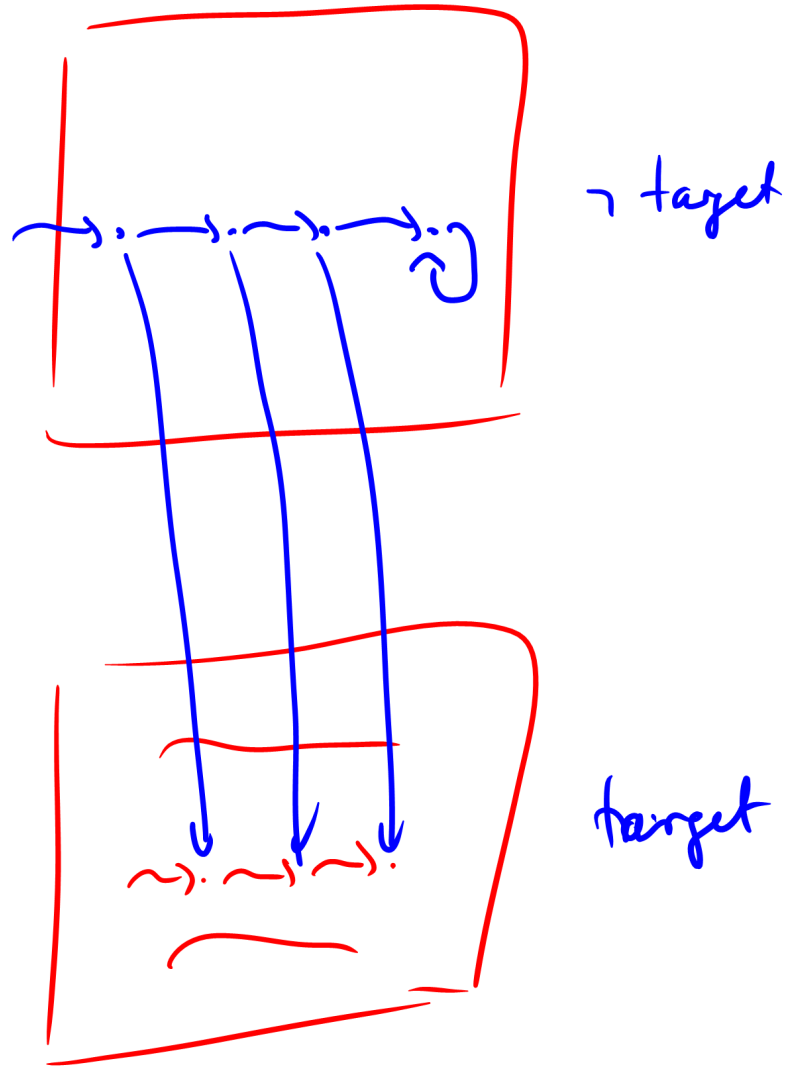
Existing Tools:

- DiPro *(Aljazzar et al.)*
- COMICS [ATVA'12]

Finding evidence paths iteratively

- *k* shortest paths search (*Han et al.*):
  - ☺ most probable paths
  - ☹ exponential growth of the state space
- Bounded model checking via SAT solving [ATVA'11]:
  - ☺ very fast (embedded loop detection)
  - ☹ no probability infos
- Bounded model checking via directed SAT-solving [SCP'14]:
  - ☺ fairly fast
  - ☹ leightweight probability infos
- Bounded model checking via SMT solving [FMOODS/FORTE'11]:
  - ☹ slower
  - ☺ gives more (but still not most) probable paths

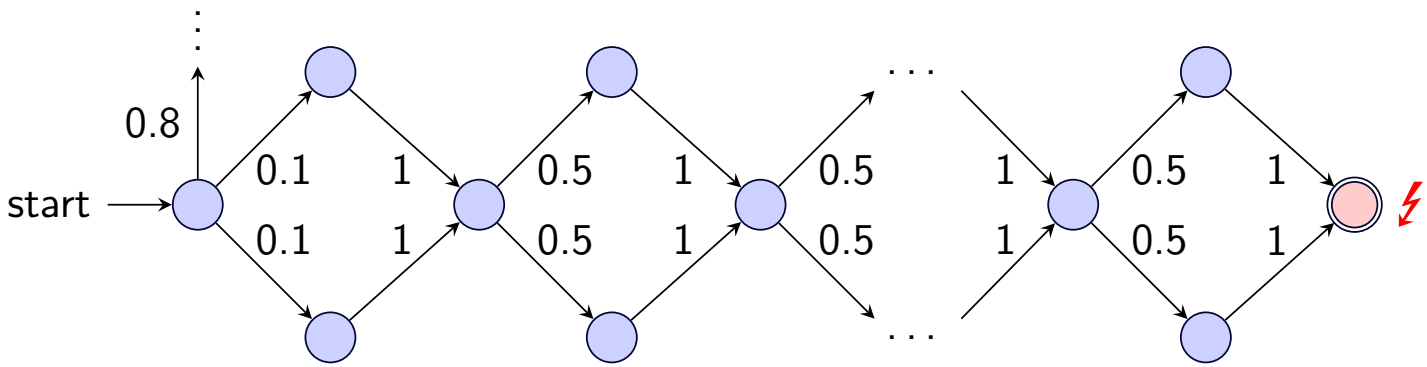



target
target

Counterexample size

Problem

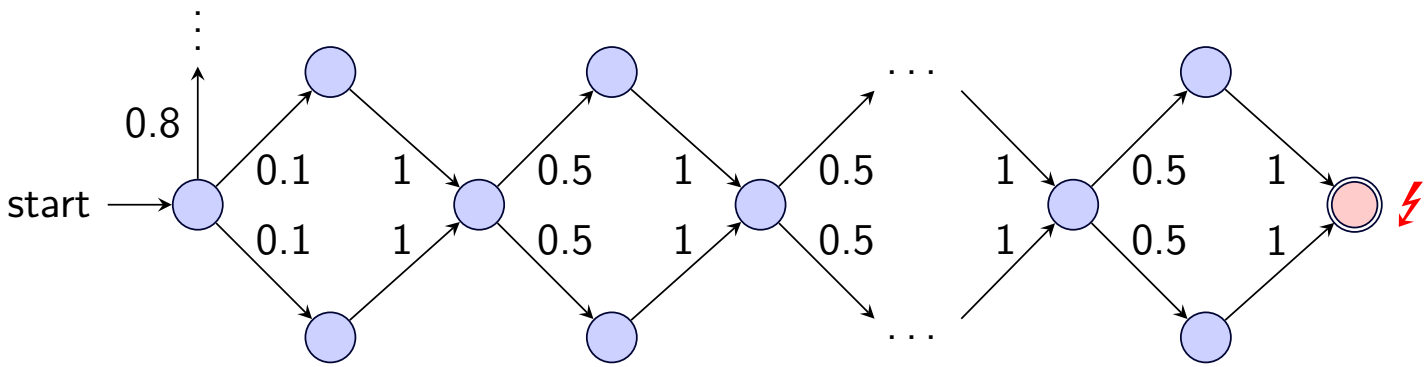
The number of paths in a counterexample C can be VERY large...



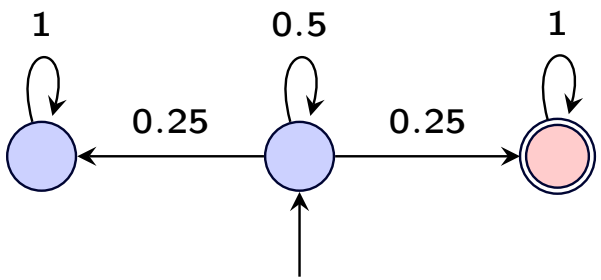
Counterexample size

Problem

The number of paths in a counterexample C can be VERY large...



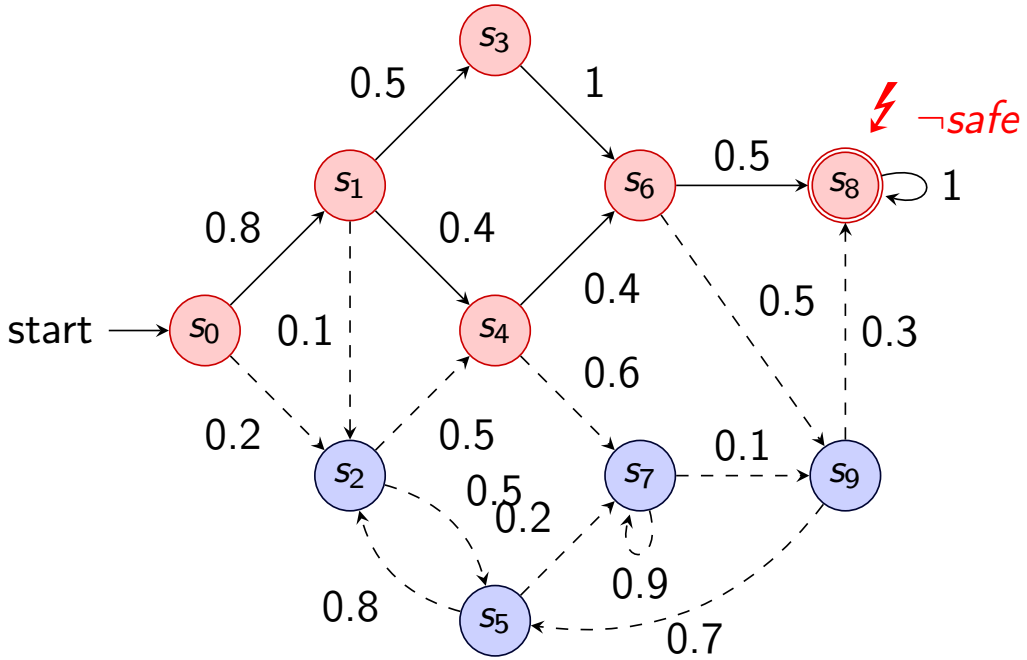
...or even infinite!



Critical subsystems for DTMCs

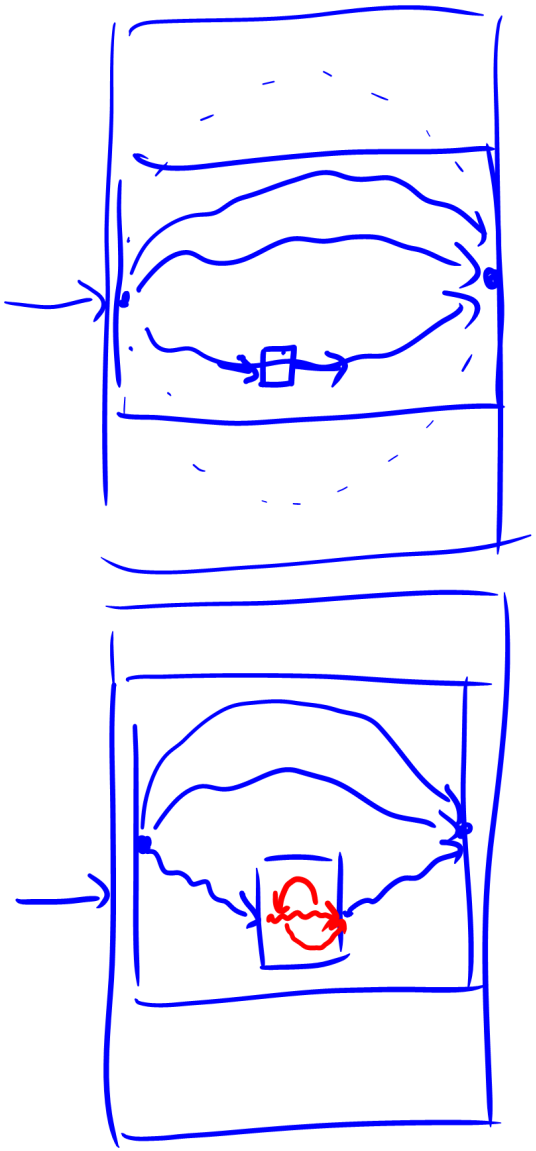
Critical subsystem [Aljazzar et al. 2009; Jansen et al. 2011]

Subset S' of the states such that the paths inside S' form a counterexample.

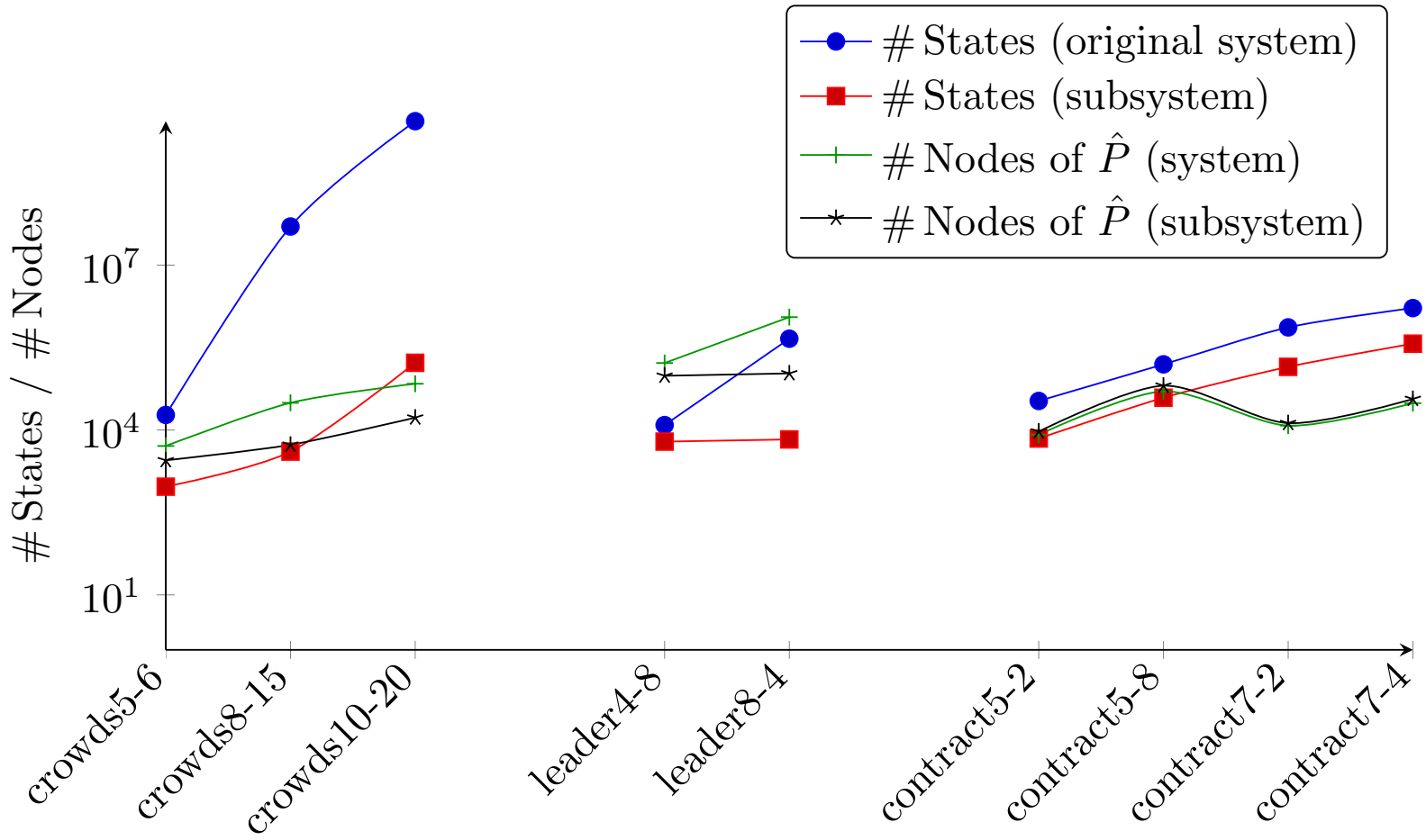


Critical subsystem for $\mathbb{P}_{\geq 0.75}(\mathcal{G}\ safe)$

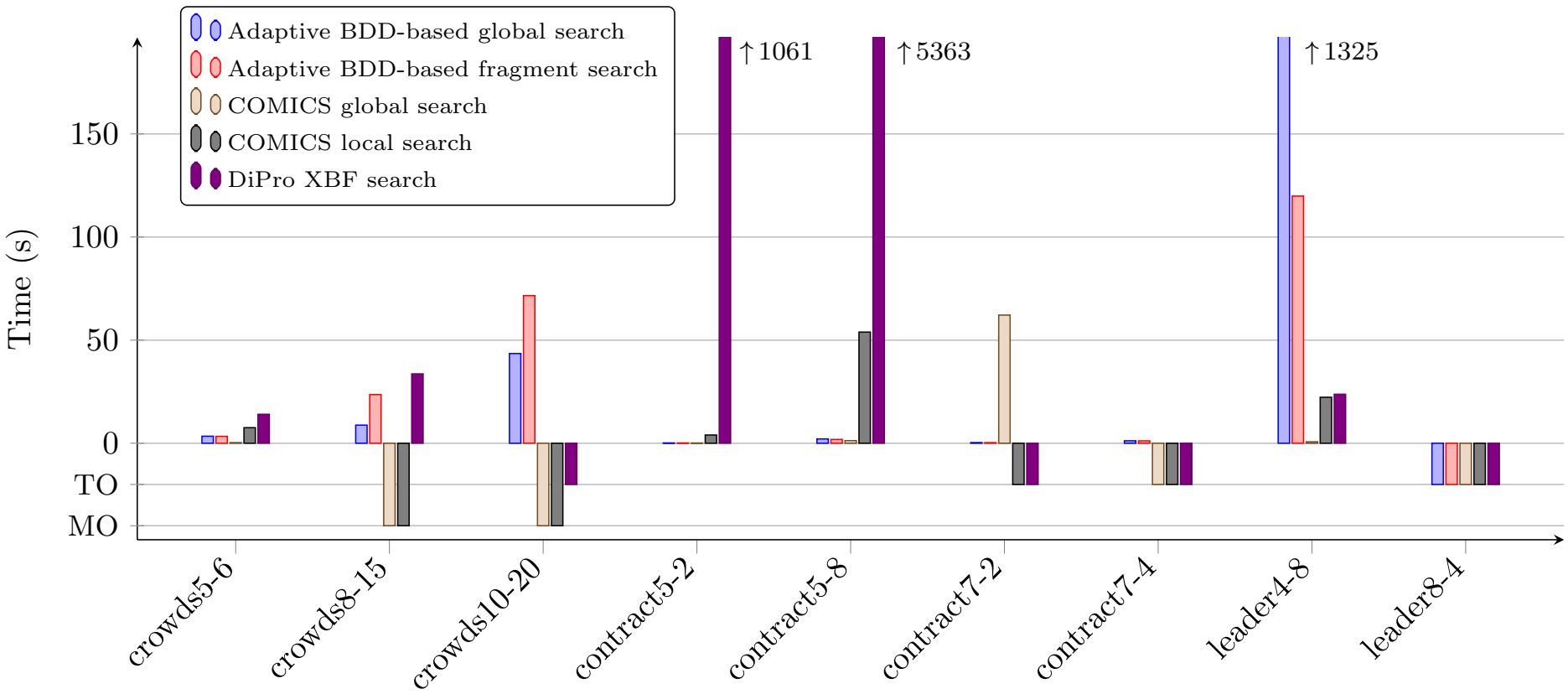
Hierarchical critical subsystems for DTMCs



Some experimental results



Some experimental results



Another approach

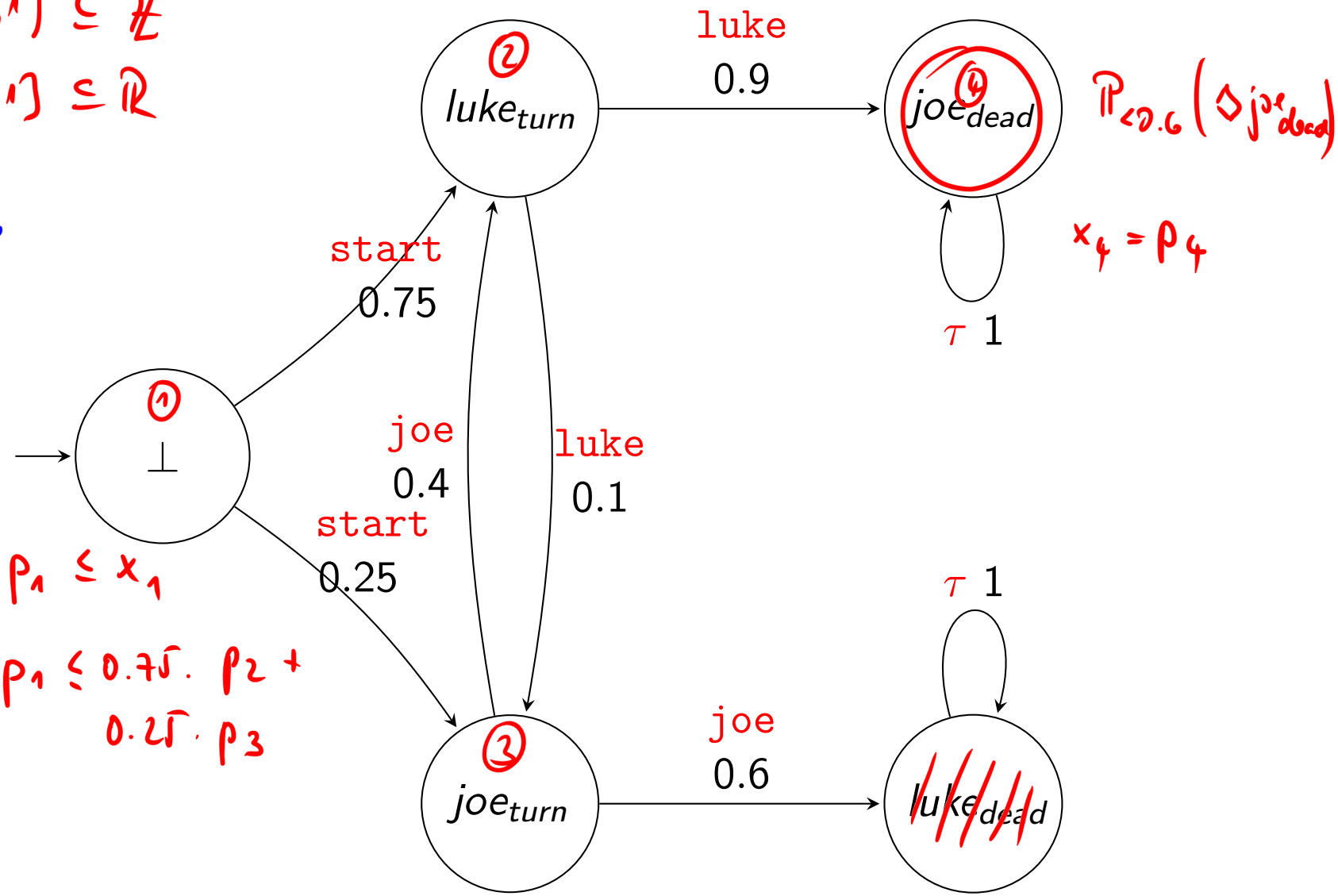
Goal

Compute a critical subsystem with a minimal number of states.

Possible approaches:

- SAT-modulo-theories solving
- Mixed integer linear programming [TCS'14]

MILP formulation for DTMCs



MILP formulation for DTMCs

- λ: probability bound
- $x_s \in \{0,1\} \subseteq \mathbb{Z}$ with $x_s = 1$ iff s belongs to the subsystem
- $p_s \in [0,1] \subseteq \mathbb{R}$: probability of state s within the subsystem

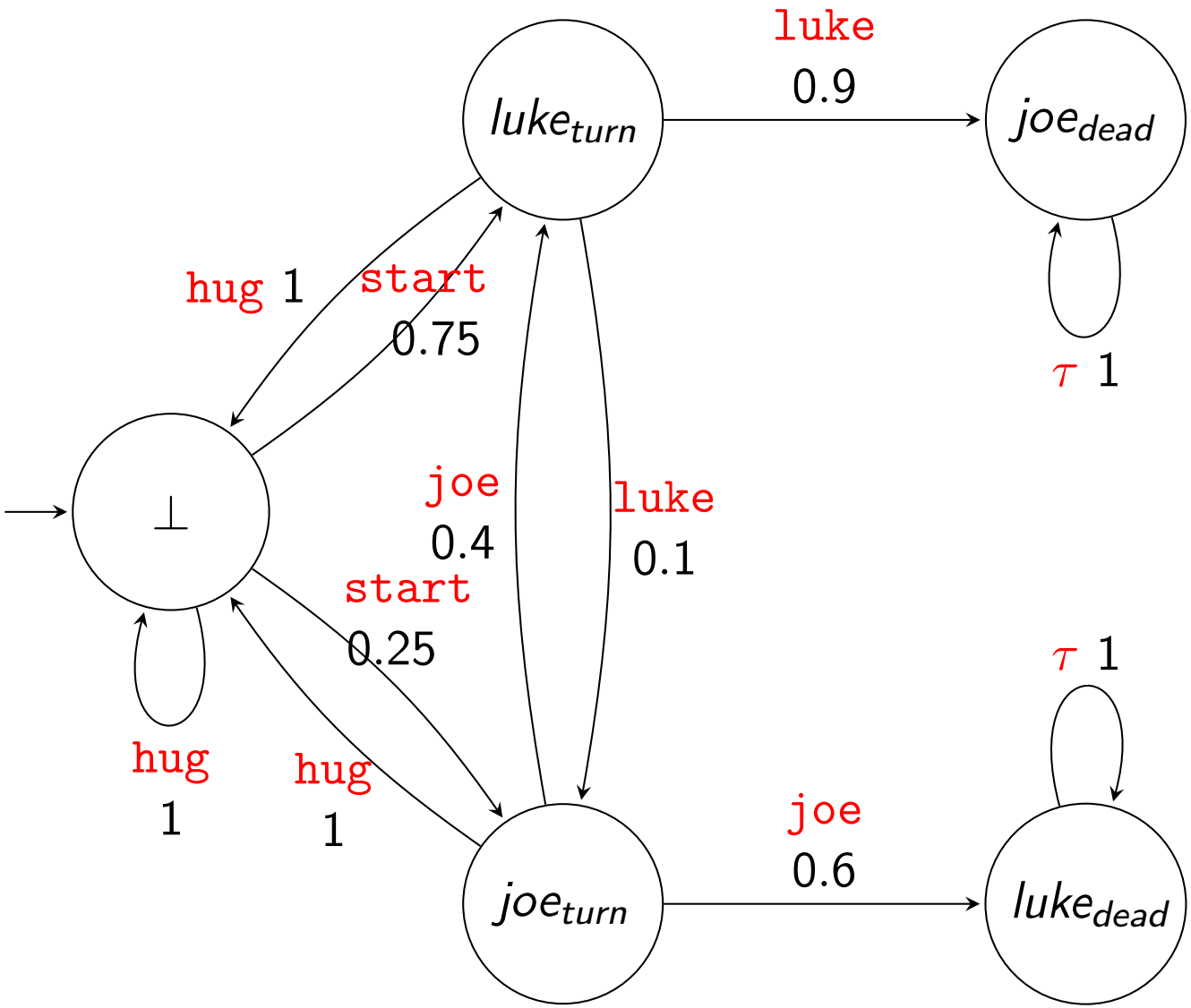
$$\min\left(-\tfrac{1}{2} p_{s_{\text{init}}} + \textstyle\sum_{s \in S} x_s\right) \text{ such that}$$

$$p_{s_{\text{init}}} > \lambda$$

$$\forall s \in T : x_s = p_s$$

$$\forall s \in S \setminus T : p_s \leq x_s \qquad p_s \leq \textstyle\sum_{s' \in S} P(s,s') \cdot p_{s'}$$

MILP formulation for MDPs



MILP formulation for MDPs

- $\sigma_{s,a} \in [0,1] \subseteq \mathbb{Z}$: encoding of the scheduler

$$\min\left(-\tfrac{1}{2} p_{s_{\text{init}}} + \textstyle\sum_{s \in S} x_s\right) \text{ such that}$$

$$p_{s_{\text{init}}} > \lambda$$

$$\textit{targets} : x_s = p_s$$

$$\textit{non-target } s : p_s \leq x_s \qquad x_s = \textstyle\sum_{a \in A} \sigma_{s,a}$$

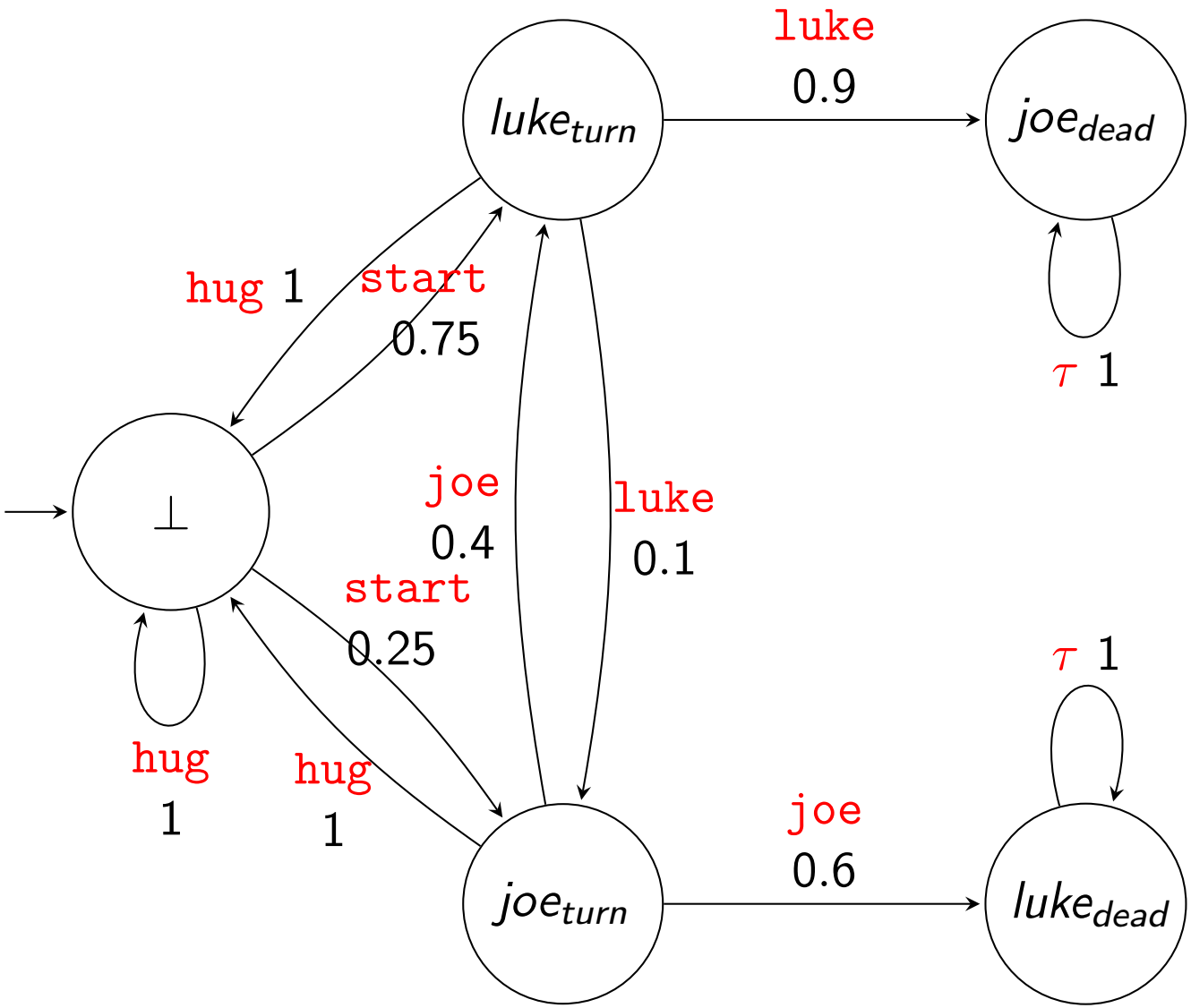
$$\textit{non-target } s,\ \textit{action } a : p_s \leq (1 - \sigma_{s,a}) + \textstyle\sum_{s' \in S} P(s,a,s') \cdot p_{s'}$$

$$\textit{probl. } s,\ s' \in \textit{succ}(s,a) : 2t_{s,s'} \leq x_s + x_{s'}$$

$$r_s < r_{s'} + (1 - t_{s,s'})$$

$$(1 - x_s) + (1 - \sigma_{s,a}) + \textstyle\sum_{s' \in \textit{succ}(s,a)} t_{s,s'} \geq 1$$

MILP approach for minimal critical command sets



MILP approach for minimal critical command sets

- $x_c \in \{0,1\} \subseteq \mathbb{Z}$: selecting commands

$$\min\left(-\tfrac{1}{2} p_{s_{\text{init}}} + \sum_{c \in C} x_c\right) \text{ such that}$$

$$p_{s_{\text{init}}} > \lambda$$

$$\textit{targets} : p_s = 1$$

$$\textit{non-target } s : p_s \leq \sum_{a \in A} \sigma_{s,a}$$

$$\textit{non-target } s,\ \textit{action } a : p_s \leq (1 - \sigma_{s,a}) + \sum_{s' \in S} P(s,a,s') \cdot p_{s'}$$

$$\textit{non-target } s,\ \textit{action } a : \sigma_{s,a} \leq x_c$$

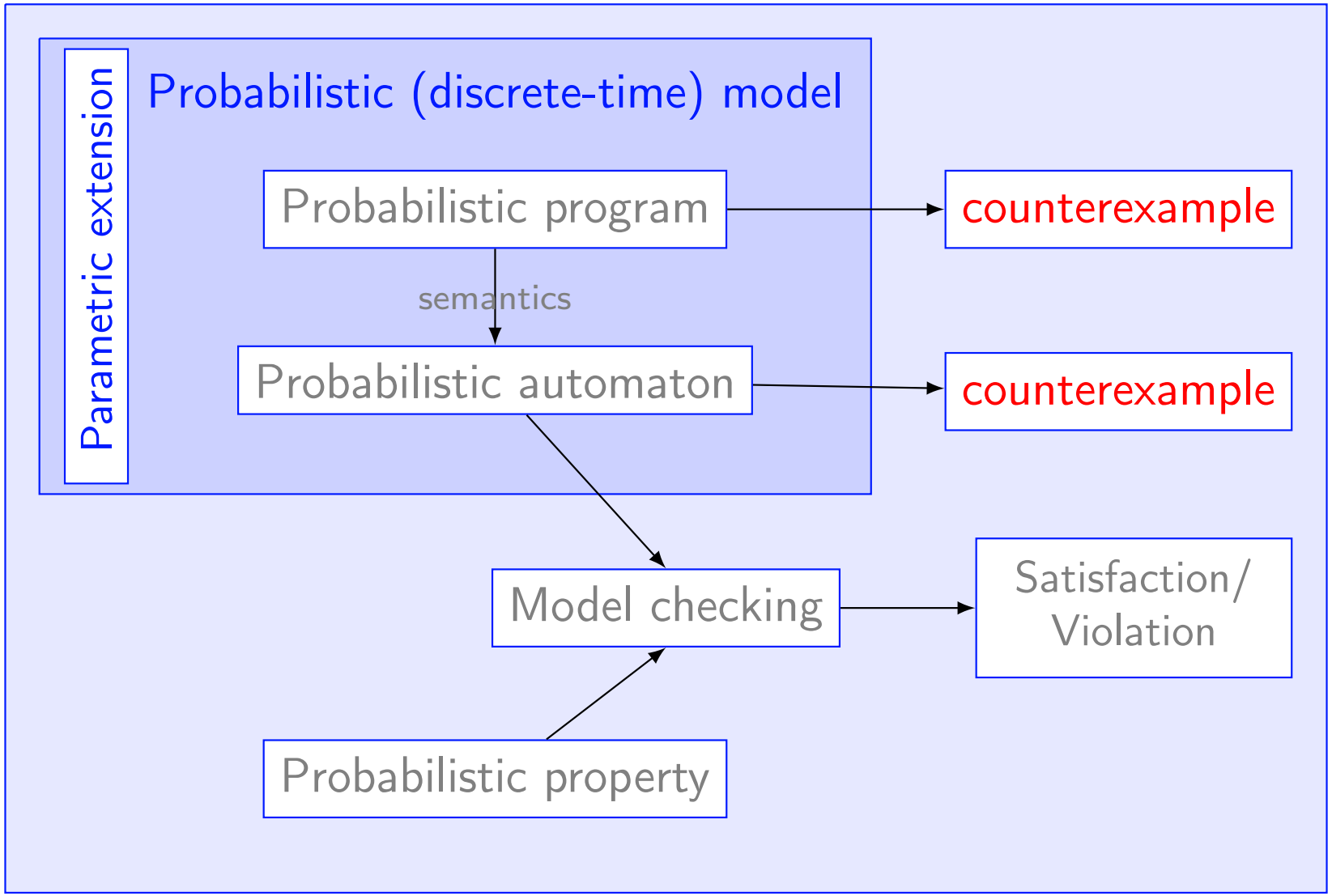
$$\textit{probl. } s,\ \textit{action } a : \sigma_{s,a} \leq \sum_{s' \in succ(s,a)} t_{s,s'}$$

$$\textit{probl. } s,\ s' \in succ(s,a) : r_s < r_{s'} + (1 - t_{s,s'})$$

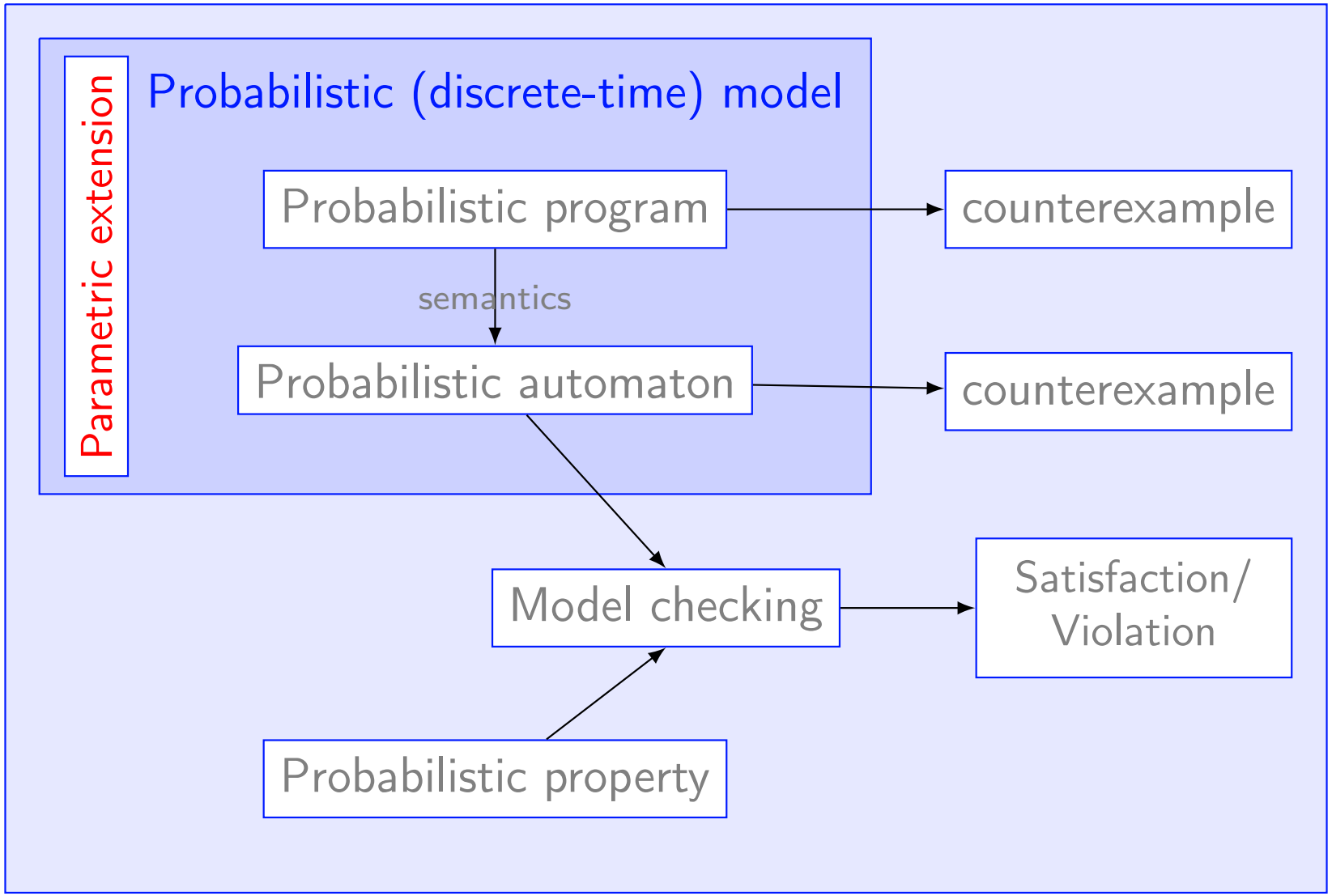
Some experimental results for DTMCs

Model	$\lvert S_{\min}\rvert$	$\lvert S\rvert$	Minimal subsystem $\lvert\mathrm{Vars}\rvert$	$\lvert\mathrm{Constr}\rvert$	Time	Mem.
brp32-2	212	1 349	1992	1988	0.09	8
brp512-2	3 263	21 509	31752	31748	18.85	70
crowds5-4	83	3 515	2161	2119	5.25	17
crowds5-6	83	18 817	14436	14184	190.50	129
crowds5-8	83	68 740	56156	55232	310.39	347
crowds12-6	*270**	829 669	395488	391848	TO *(223)*	3944
nand5-2	394	1 728	3457	3447	19.50	21
nand5-3	614	2 526	5053	5043	50.01	36
nand5-4	854	3 324	6649	6639	299.62	101
sleader4-4	392	782	1565	1563	0.33	10
sleader4-6	1 950	3 902	7805	7803	2.24	23
sleader4-8	6 149	12 302	24605	24603	13.00	156
sleader8-4	229 389	458 847	917695	917693	1 021,33	1018

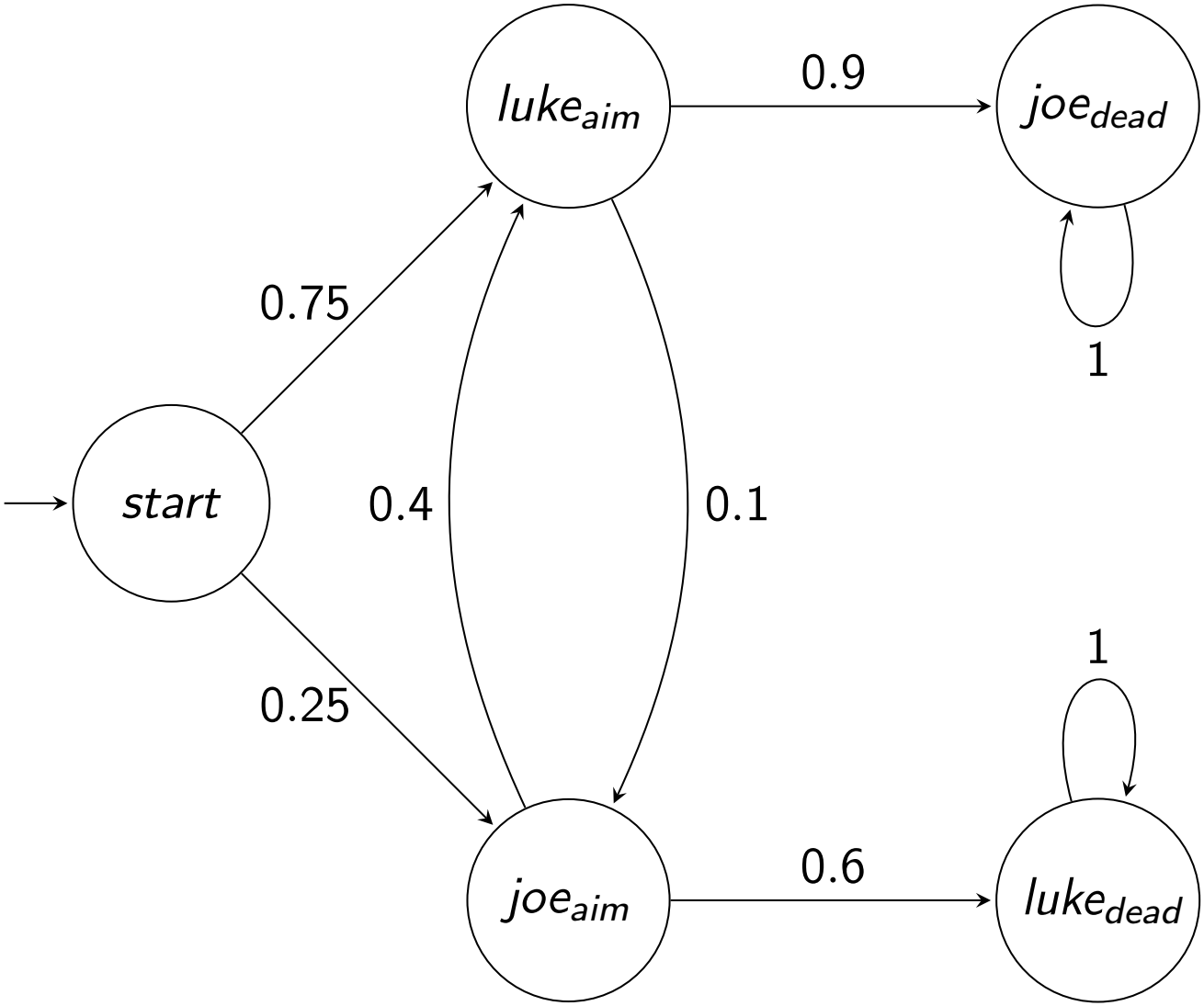
Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

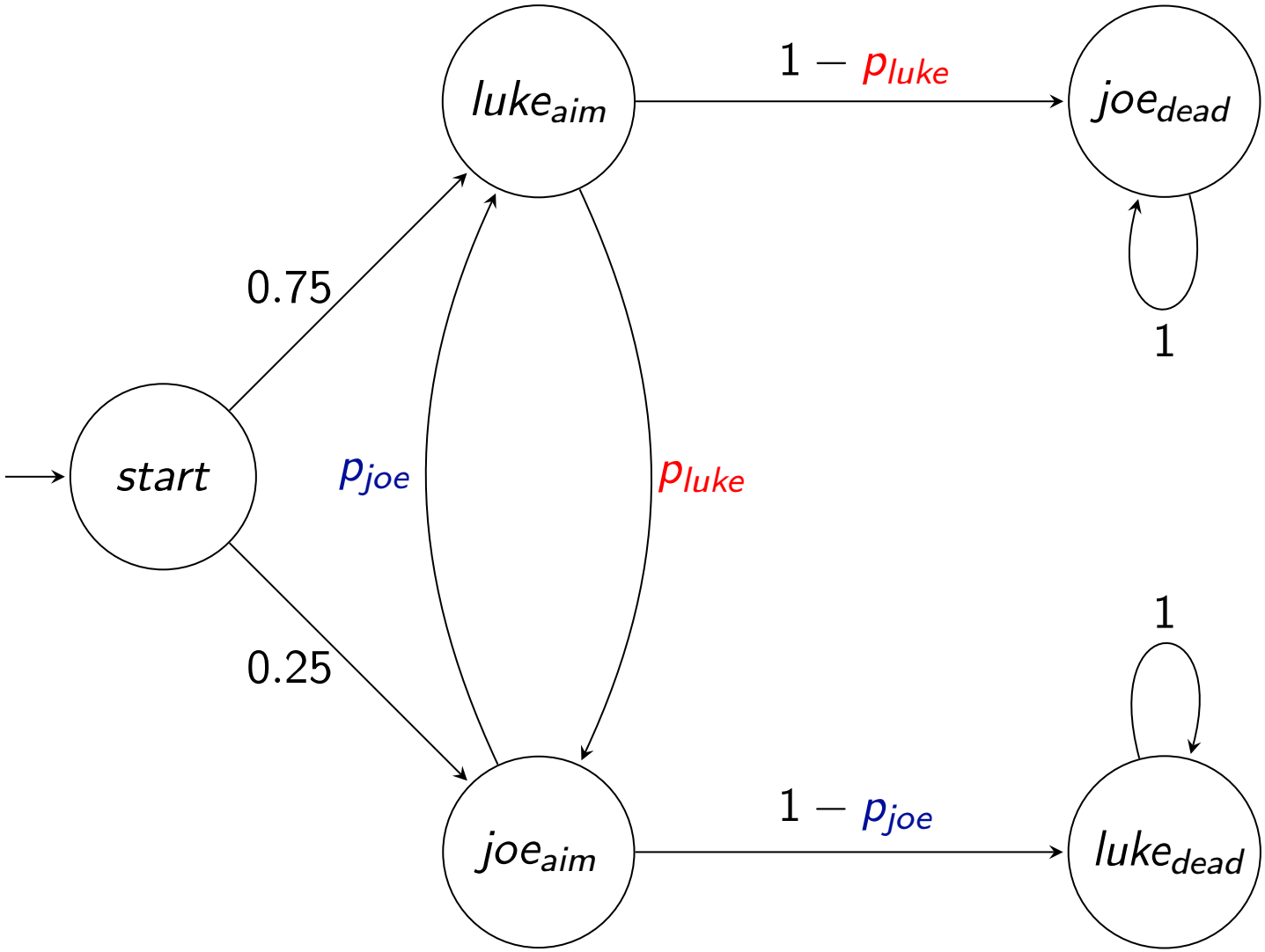
Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Parametric models



Parametric models



Parametric model checking

Daws (2004)

- State elimination
- Reachability probabilities as regular expressions

Parametric model checking

Daws (2004)

- State elimination
- Reachability probabilities as regular expressions

Hahn, Hermanns, Wachter, Zhang (2009)

- Based on Daws' idea
- Fractions of polynomials describe probabilities
- Tool PARAM

Parametric model checking

Daws (2004)

- State elimination
- Reachability probabilities as regular expressions

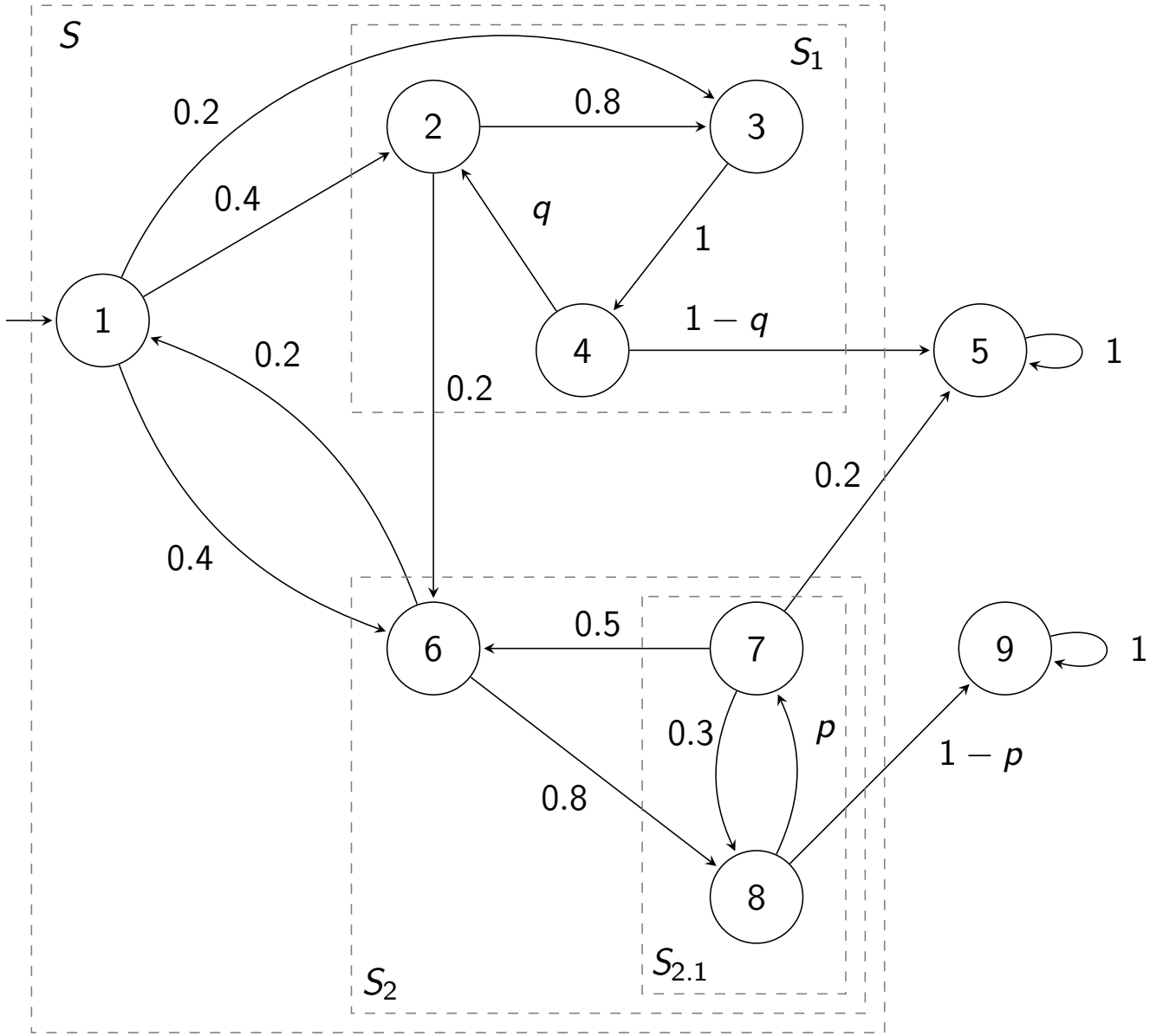
Hahn, Hermanns, Wachter, Zhang (2009)

- Based on Daws' idea
- Fractions of polynomials describe probabilities
- Tool PARAM

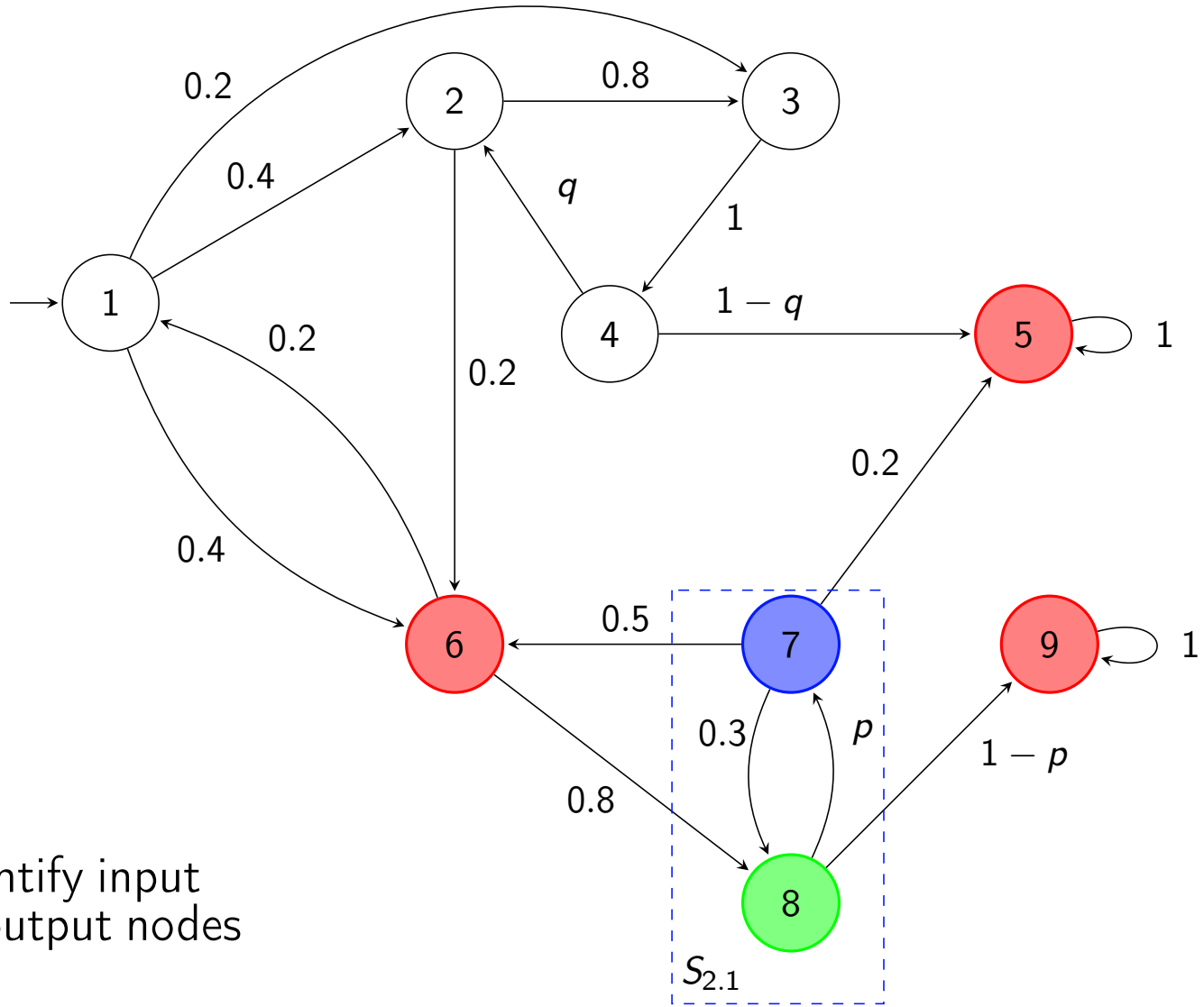
Our approach [QEST'14]:

- Parametric SCC-based model checking
- Special datatype for polynomials with partial factorization
- Tool COMICS

Example

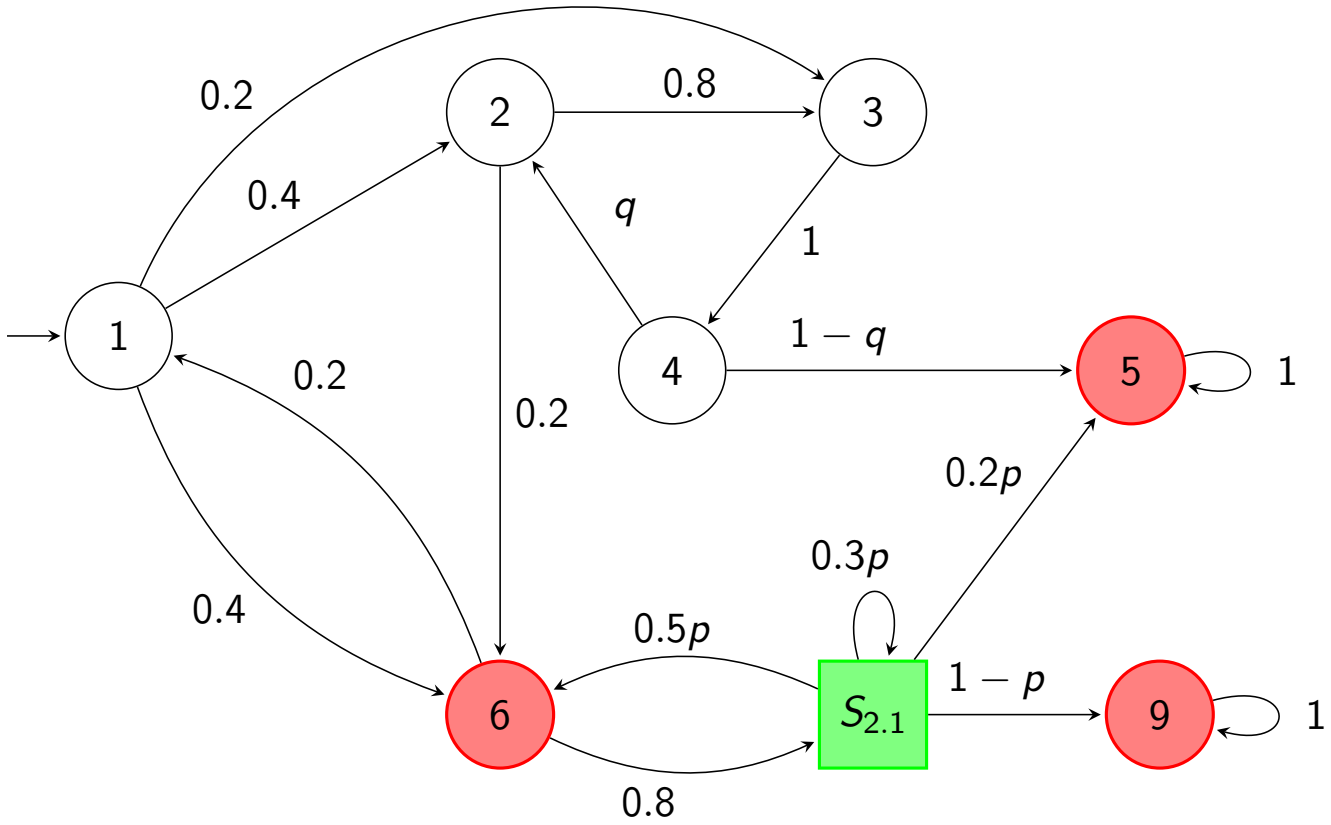


Example



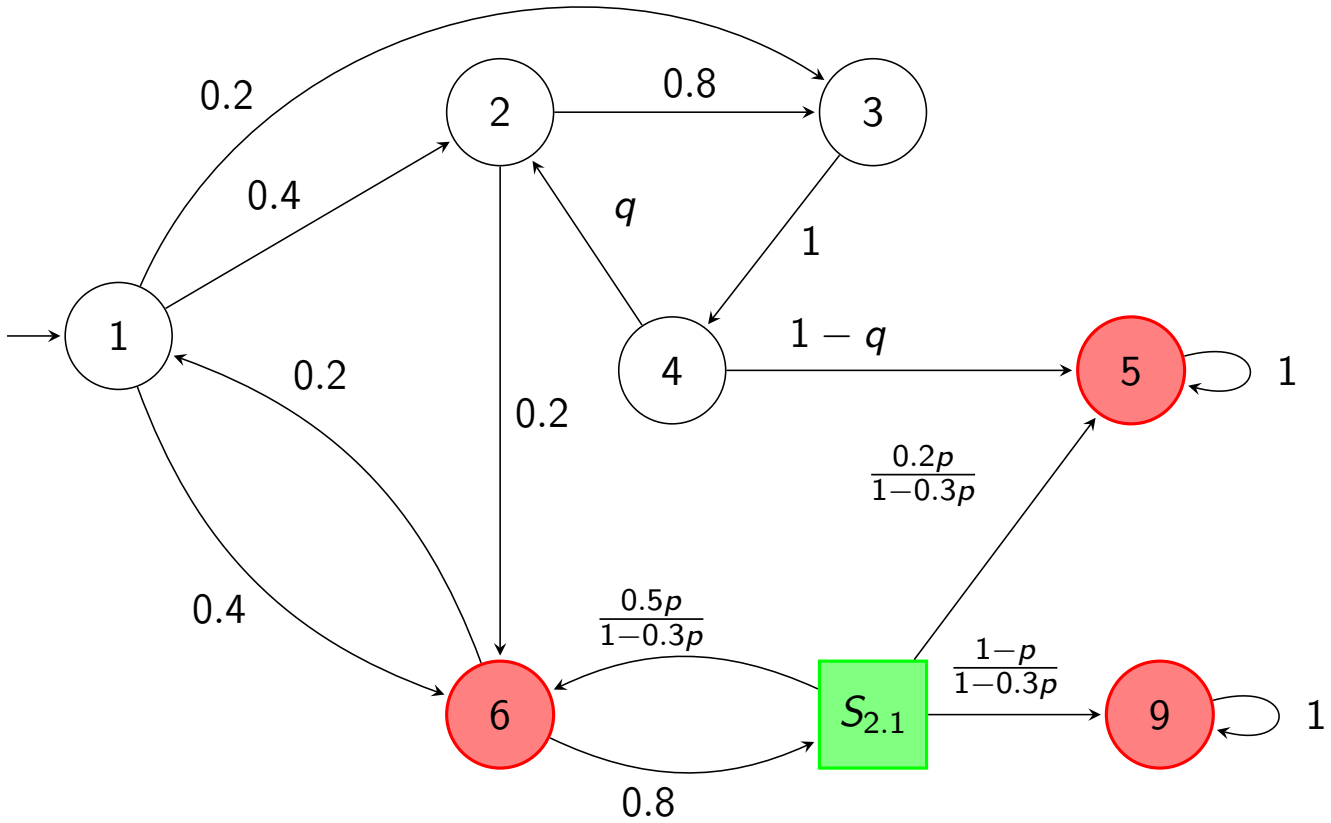
Identify input
and output nodes

Example



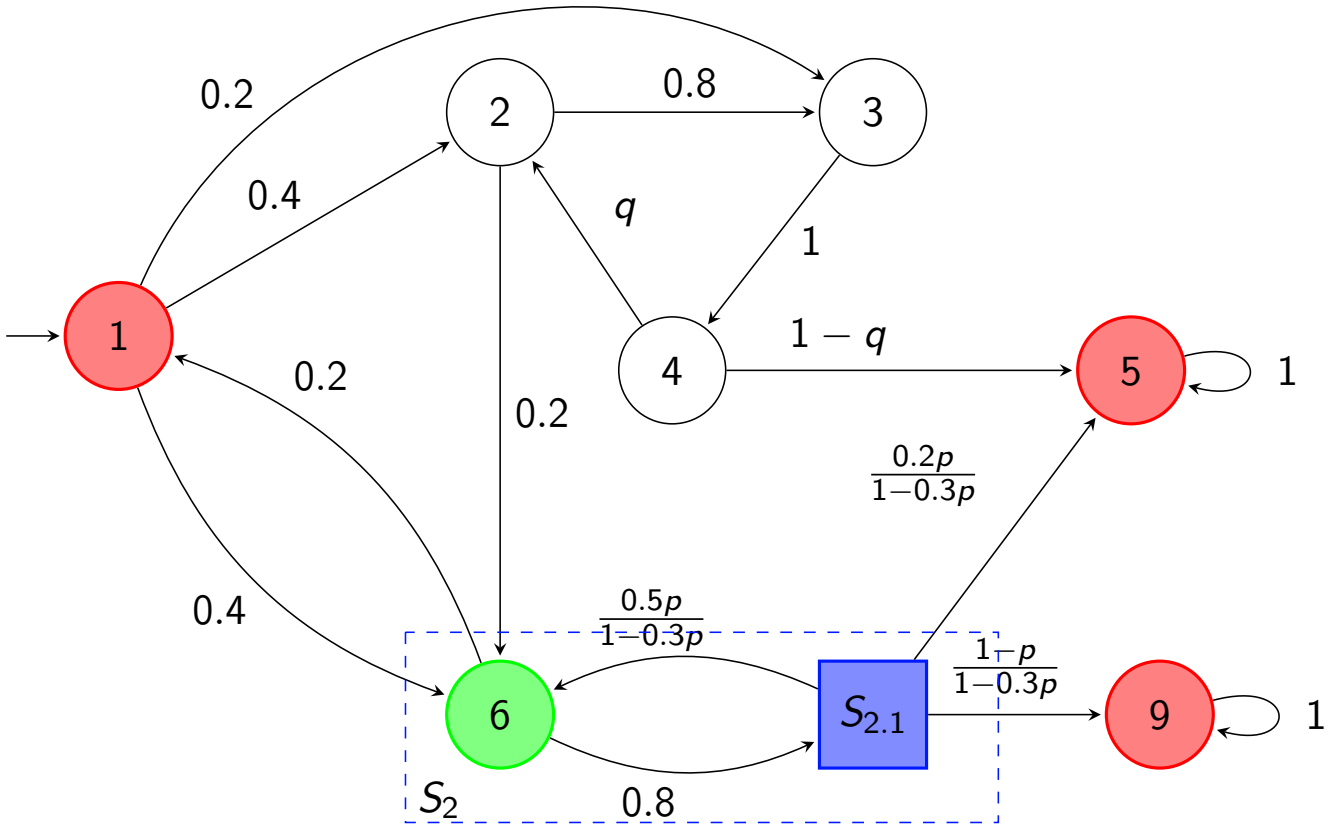
Compute outgoing probabilities

Example

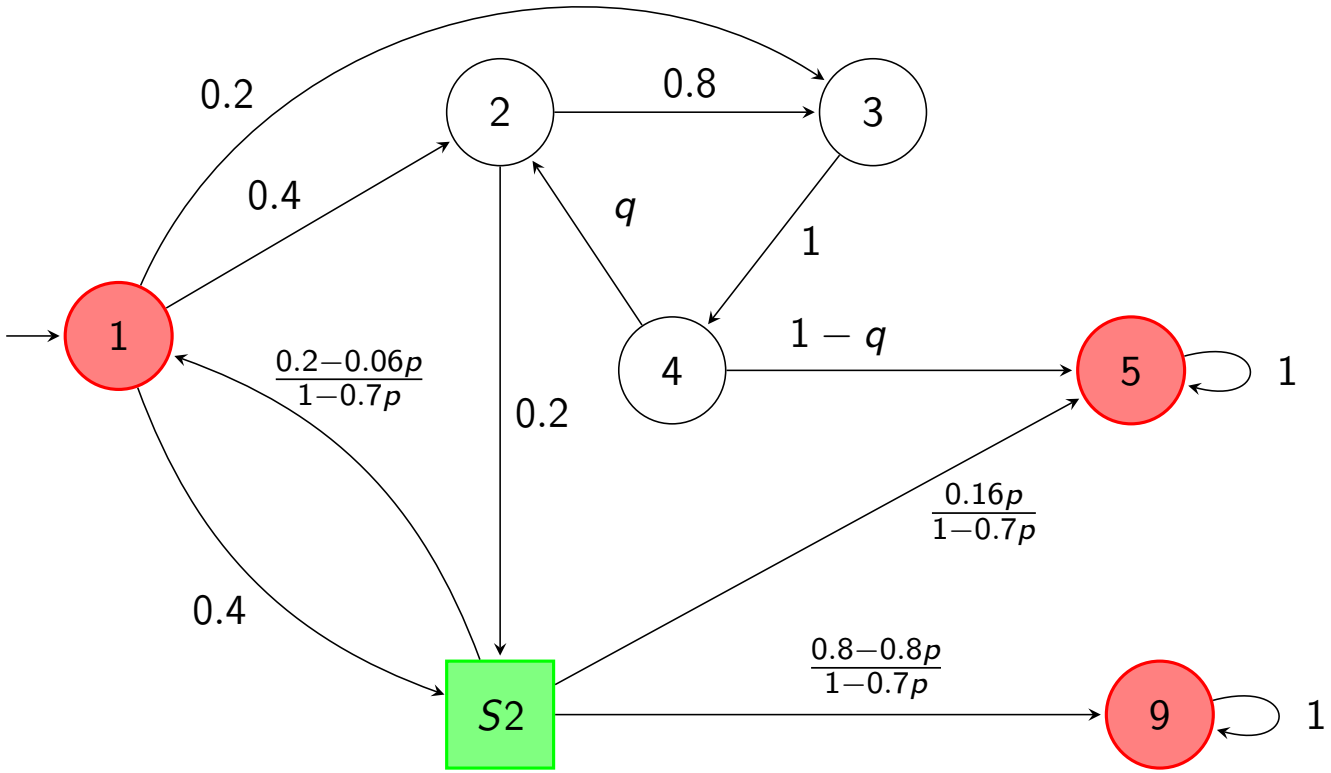


Scale with sum of all outgoing probabilities

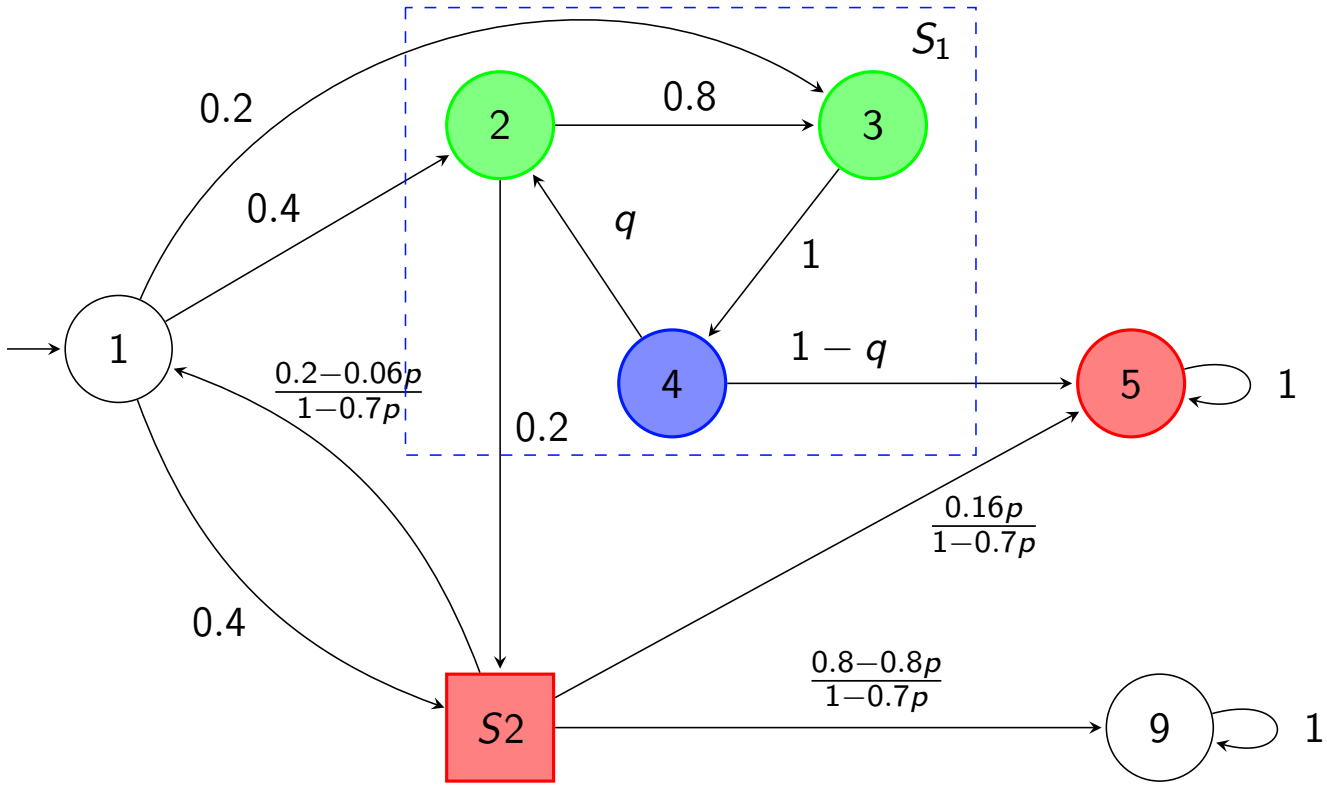
Example


0.2
2
0.8
3
0.4
q
1
1
4
1 − q
5
1
0.2
0.2
0.2p / 1−0.3p
0.4
0.5p / 1−0.3p
1−p / 1−0.3p
6
S2.1
9
1
S2
0.8

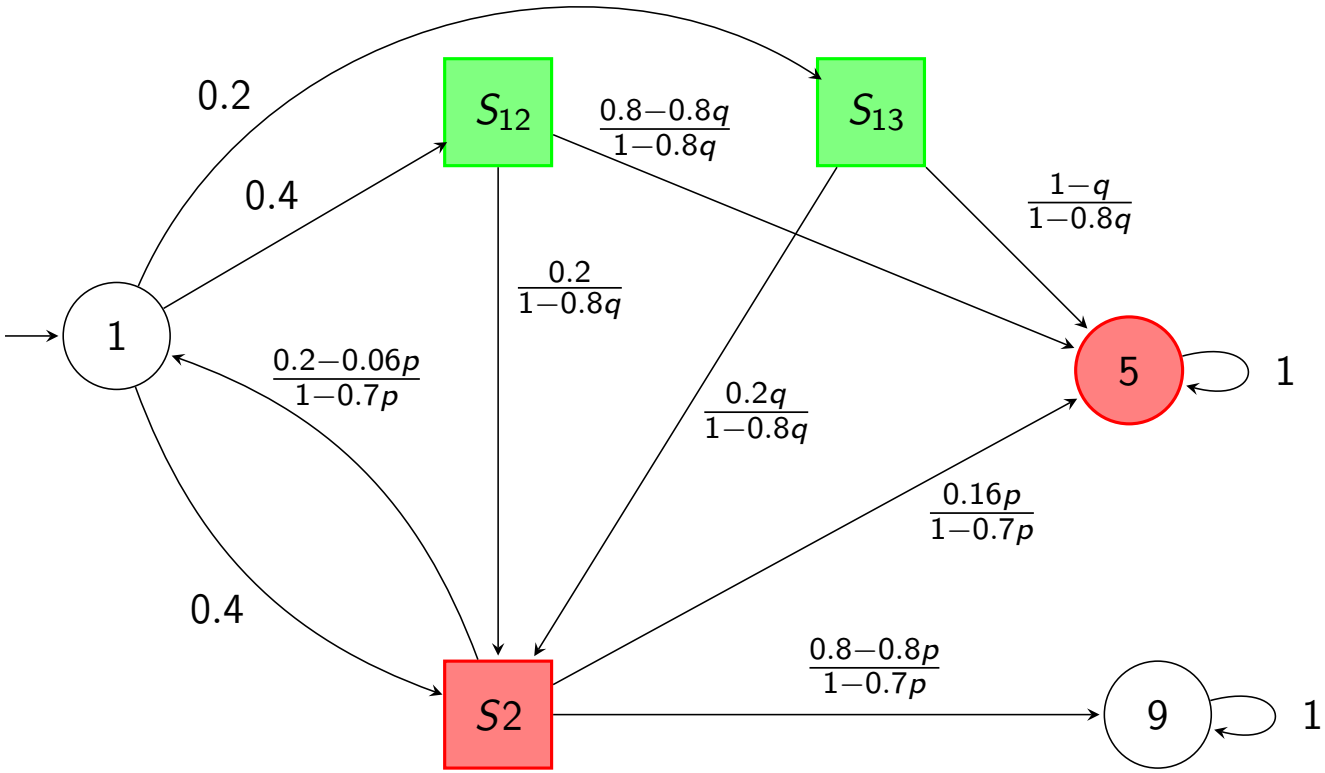
Example


0.2
2
0.8
3
0.4
q
1
1
4
1 − q
5
1
$\frac{0.2-0.06p}{1-0.7p}$
0.2
$\frac{0.16p}{1-0.7p}$
0.4
$\frac{0.8-0.8p}{1-0.7p}$
S2
9
1

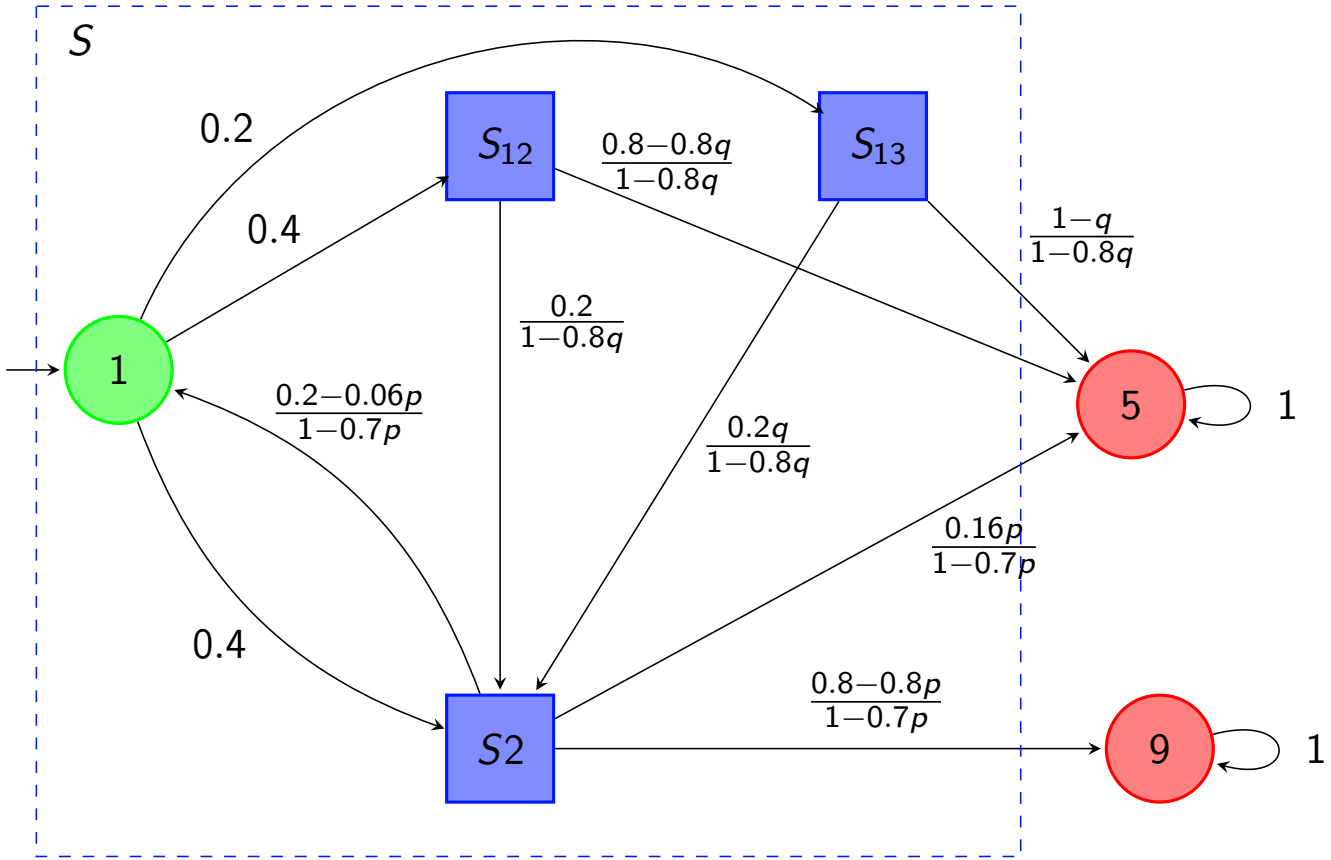
Example


S_1
0.2
0.8
2
3
0.4
q
1
1
$1-q$
4
5
1
$\frac{0.2-0.06p}{1-0.7p}$
0.2
$\frac{0.16p}{1-0.7p}$
0.4
$\frac{0.8-0.8p}{1-0.7p}$
$S2$
9
1

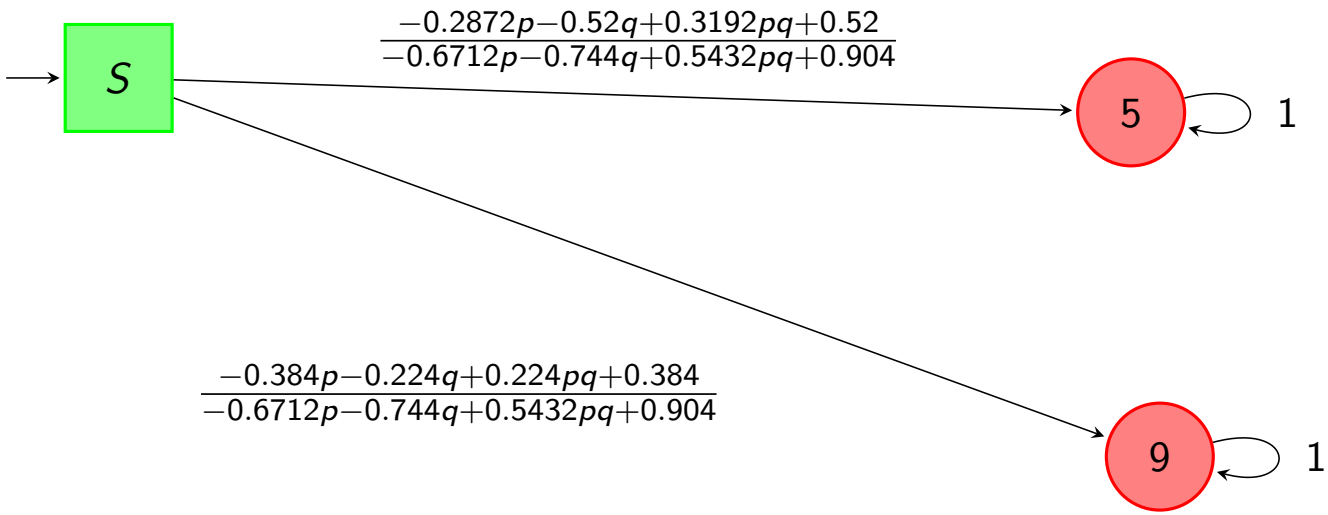
Example


0.2
0.4
S_{12}
$\frac{0.8-0.8q}{1-0.8q}$
S_{13}
$\frac{1-q}{1-0.8q}$
1
$\frac{0.2}{1-0.8q}$
5
1
$\frac{0.2-0.06p}{1-0.7p}$
$\frac{0.2q}{1-0.8q}$
$\frac{0.16p}{1-0.7p}$
0.4
$\frac{0.8-0.8p}{1-0.7p}$
S2
9
1

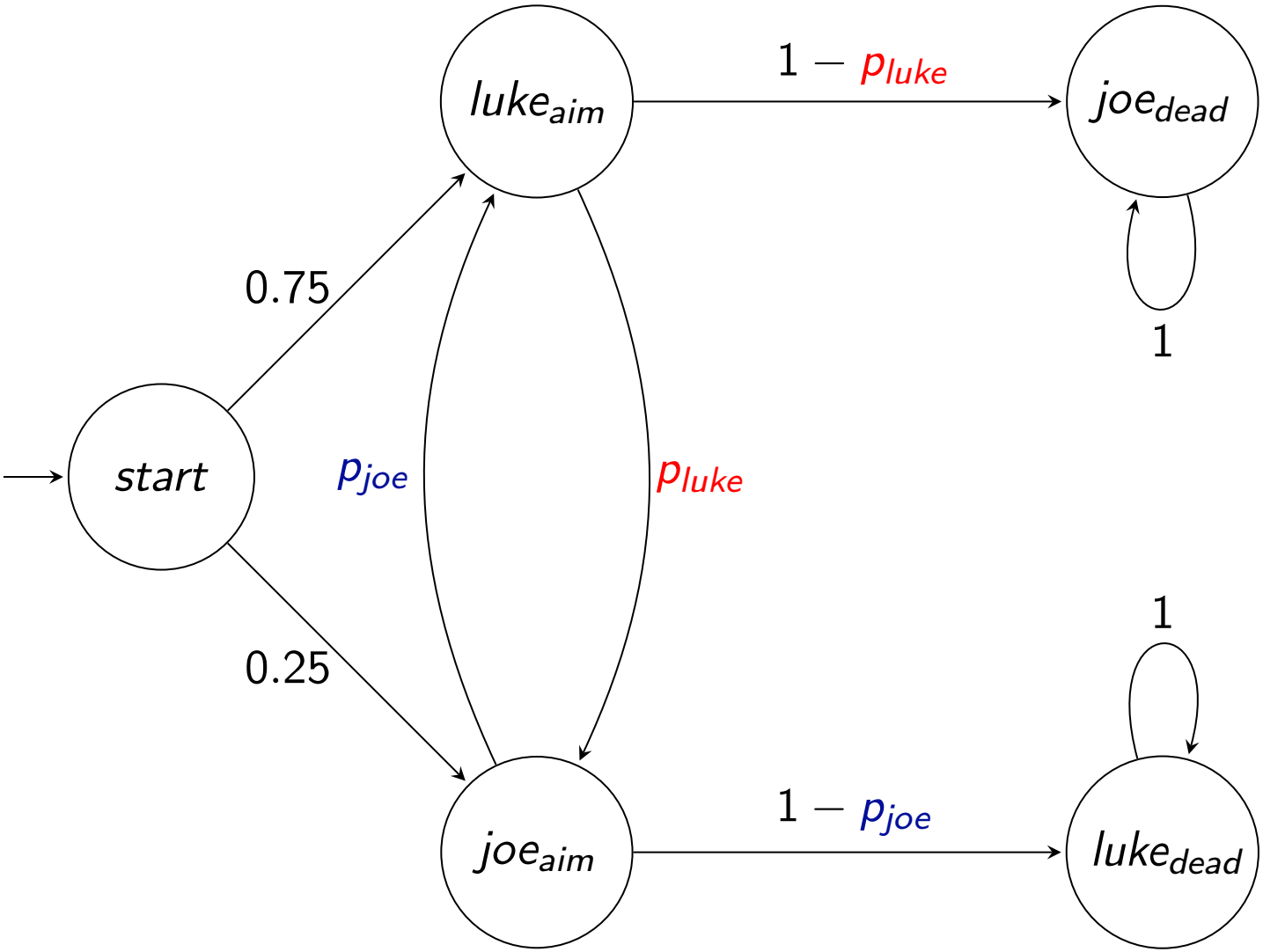
Example


S
0.2
0.4
S_{12}
$\frac{0.8-0.8q}{1-0.8q}$
S_{13}
$\frac{1-q}{1-0.8q}$
$\frac{0.2}{1-0.8q}$
1
$\frac{0.2-0.06p}{1-0.7p}$
$\frac{0.2q}{1-0.8q}$
5
1
$\frac{0.16p}{1-0.7p}$
0.4
$\frac{0.8-0.8p}{1-0.7p}$
S2
9
1

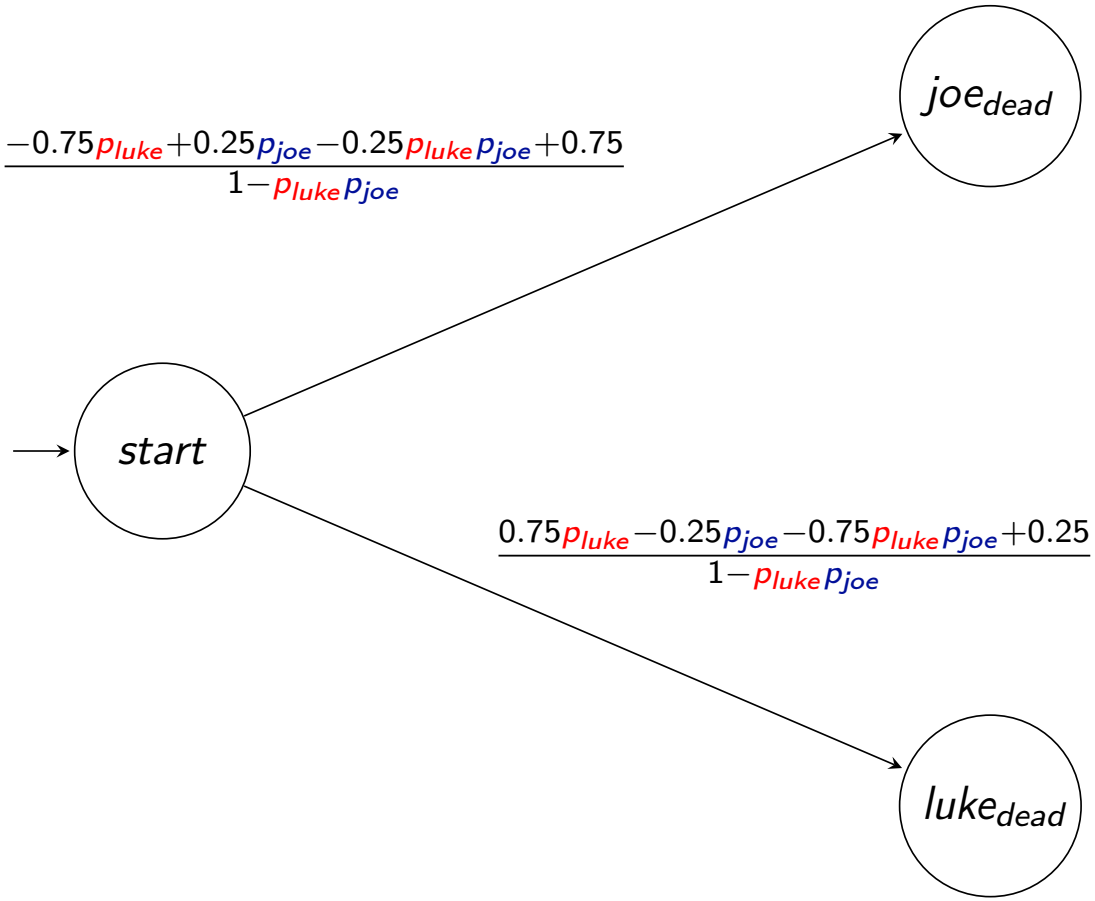
Example


S
$\frac{-0.2872p-0.52q+0.3192pq+0.52}{-0.6712p-0.744q+0.5432pq+0.904}$
5
1
$\frac{-0.384p-0.224q+0.224pq+0.384}{-0.6712p-0.744q+0.5432pq+0.904}$
9
1

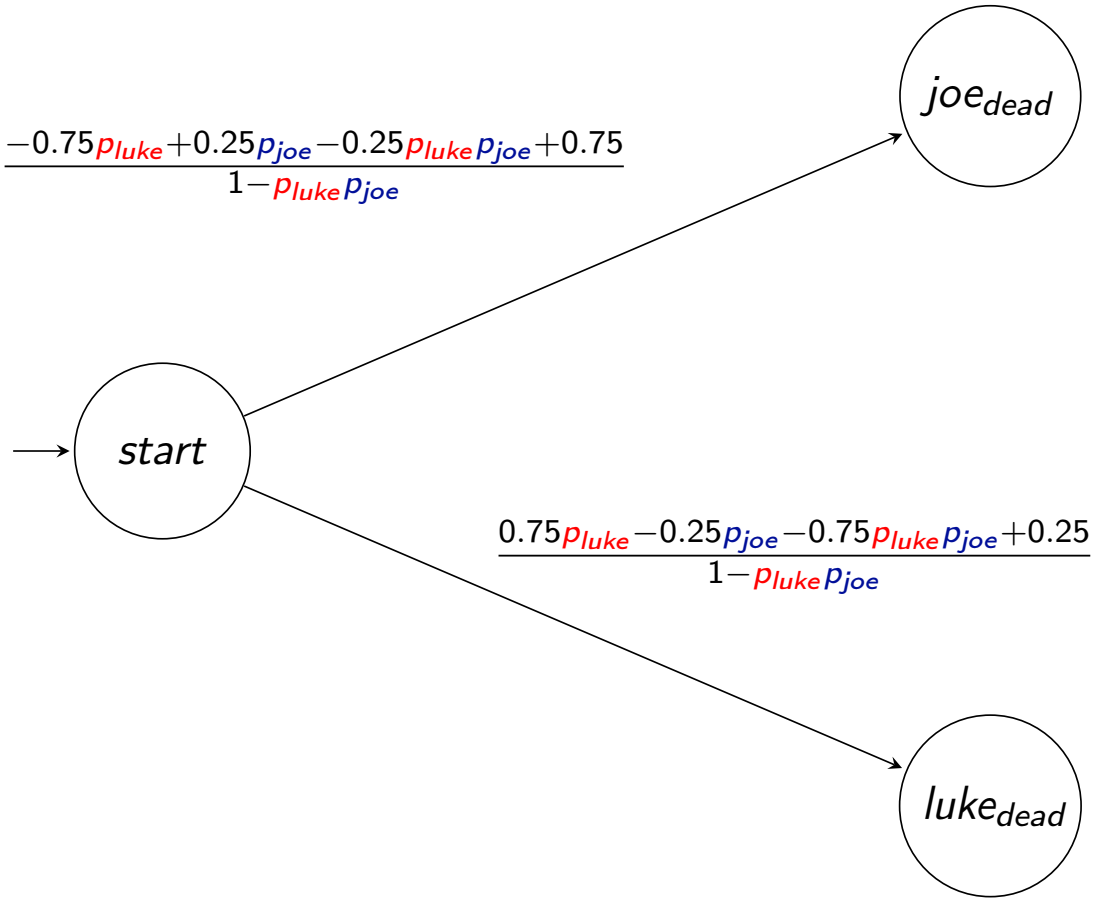
Example: Parametric dueling cowboys



Example: Parametric dueling cowboys



Example: Parametric dueling cowboys



$$\frac{0.75p_{luke} - 0.25p_{joe} - 0.75p_{luke}p_{joe} + 0.25}{1 - p_{luke}p_{joe}} \geq 0.75 \iff p_{luke} \geq \frac{1}{3}p_{joe} + \frac{2}{3}$$

Case studies: BRP

Bounded Retransmission Protocol **(BRP)**

- Sending files in unreliable network
- Parameters : probability of reliability of 2 channels
- Structure : acyclic

Graph		SCC MC		STATE ELIM		PARAM	
States	Trans.	Time	Mem	Time	Mem	Time	Mem
3528	4611	29.05	48.10	4.33	61.17	98.99	32.90
4361	5763	511.50	501.71	6.87	78.49	191.52	58.43
7048	9219	548.73	281.86	25.05	216.05	988.28	142.66
10759	13827	147.31	176.89	85.54	682.24	3511.96	304.07
21511	27651	1602.53	776.48	718.66	3134.59	34322.60	1757.12

Case studies: BRP

Bounded Retransmission Protocol **(BRP)**

- Sending files in unreliable network
- Parameters : probability of reliability of 2 channels
- Structure : acyclic

Graph		SCC MC		STATE ELIM		PARAM	
States	Trans.	Time	Mem	Time	Mem	Time	Mem
3528	4611	29.05	48.10	4.33	61.17	98.99	32.90
4361	5763	511.50	501.71	6.87	78.49	191.52	58.43
7048	9219	548.73	281.86	25.05	216.05	988.28	142.66
10759	13827	147.31	176.89	85.54	682.24	3511.96	304.07
21511	27651	1602.53	776.48	718.66	3134.59	34322.60	1757.12

Case studies: Crowds

Crowds protocol

- Anonymous network communication using random routing
- Parameters:
  - probability of member being "good"
  - probability if "good" member delivers message
- Structure: nested SCCs

Graph		SCC MC		STATE ELIM		PARAM	
States	Trans.	Time	Mem	Time	Mem	Time	Mem
198201	348349	60.90	140.15	243.07	133.91	46380.00	227.66
482979	728677	35.06	478.85	247.75	297.40	TO	—
726379	1283297	223.24	515.61	1632.63	477.10	TO	—
961499	1452537	81.88	1027.78	646.76	589.21	TO	—
1729494	2615272	172.59	2372.35	1515.63	1063.15	TO	—
2888763	5127151	852.76	2345.06	12326.80	2123.96	TO	—

TO: 14h = 50400s

Case studies: Crowds

Crowds protocol

- Anonymous network communication using random routing
- Parameters:
  - probability of member being "good"
  - probability if "good" member delivers message
- Structure: nested SCCs

Graph		SCC MC		STATE ELIM		PARAM	
States	Trans.	Time	Mem	Time	Mem	Time	Mem
198201	348349	60.90	140.15	243.07	133.91	46380.00	227.66
482979	728677	35.06	478.85	247.75	297.40	TO	—
726379	1283297	223.24	515.61	1632.63	477.10	TO	—
961499	1452537	81.88	1027.78	646.76	589.21	TO	—
1729494	2615272	172.59	2372.35	1515.63	1063.15	TO	—
2888763	5127151	852.76	2345.06	12326.80	2123.96	TO	—

TO: 14h = 50400s

Case studies: NAND

NAND multiplexing

- Computation on copies of NAND unit with unreliable hardware
- Parameters:
  - probability of faultiness of units
  - probability of erroneous input
- Structure: acyclic, many paths join in specific states and diverge again

Graph		SCC MC		STATE ELIM		PARAM	
States	Trans.	Time	Mem	Time	Mem	Time	Mem
7393	11207	8.35	114.60	17.02	255.13	5.00	10.67
14323	21567	39.71	366.79	59.60	926.33	15.26	16.89
21253	31927	100.32	795.31	121.40	2050.67	29.51	24.45
28183	42287	208.41	1405.16	218.85	3708.27	50.45	30.47
78334	121512	639.29	3785.11	—	MO	1138.82	111.58

MO: 4GB

Case studies: NAND

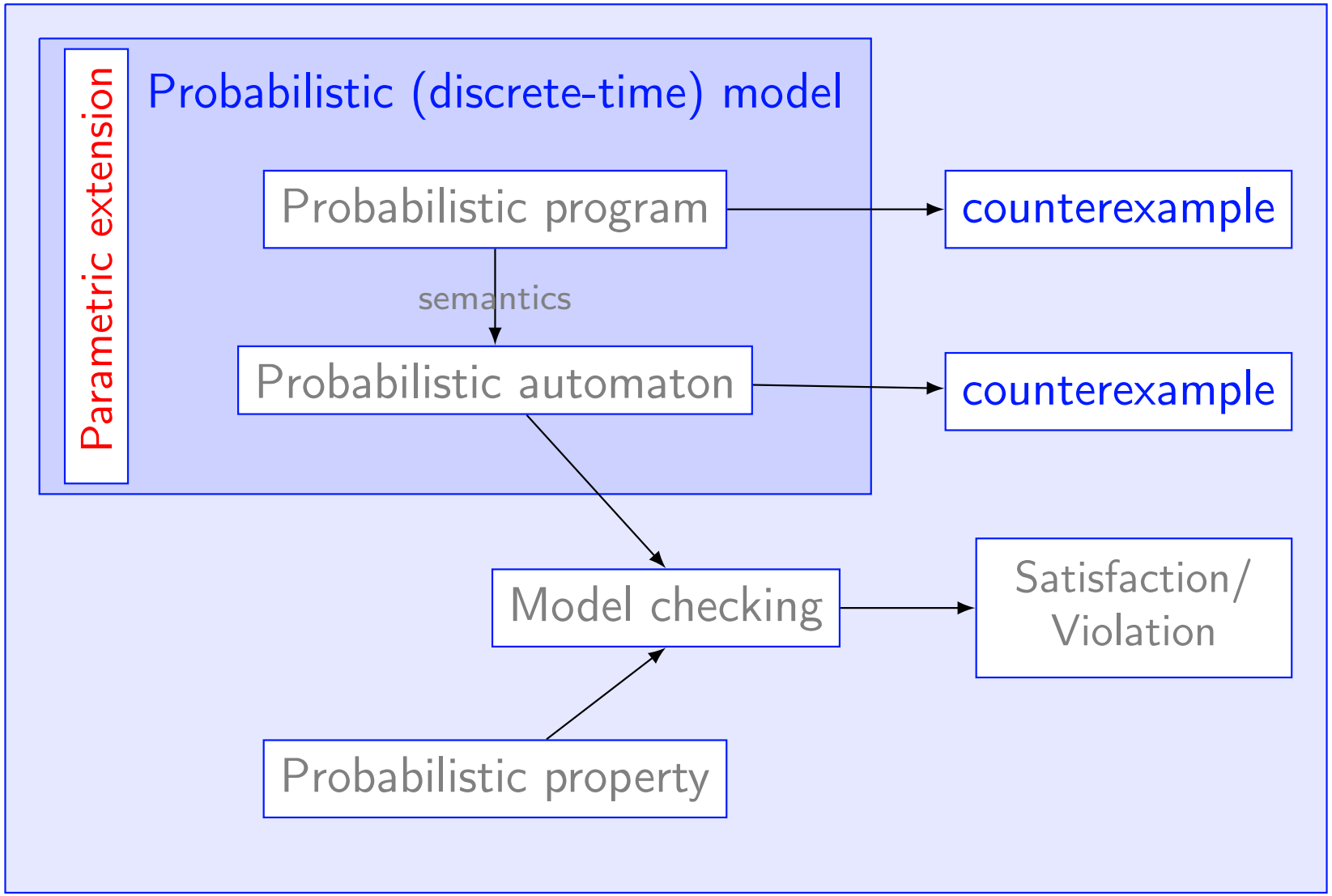
NAND multiplexing

- Computation on copies of NAND unit with unreliable hardware
- Parameters:
  - probability of faultiness of units
  - probability of erroneous input
- Structure: acyclic, many paths join in specific states and diverge again

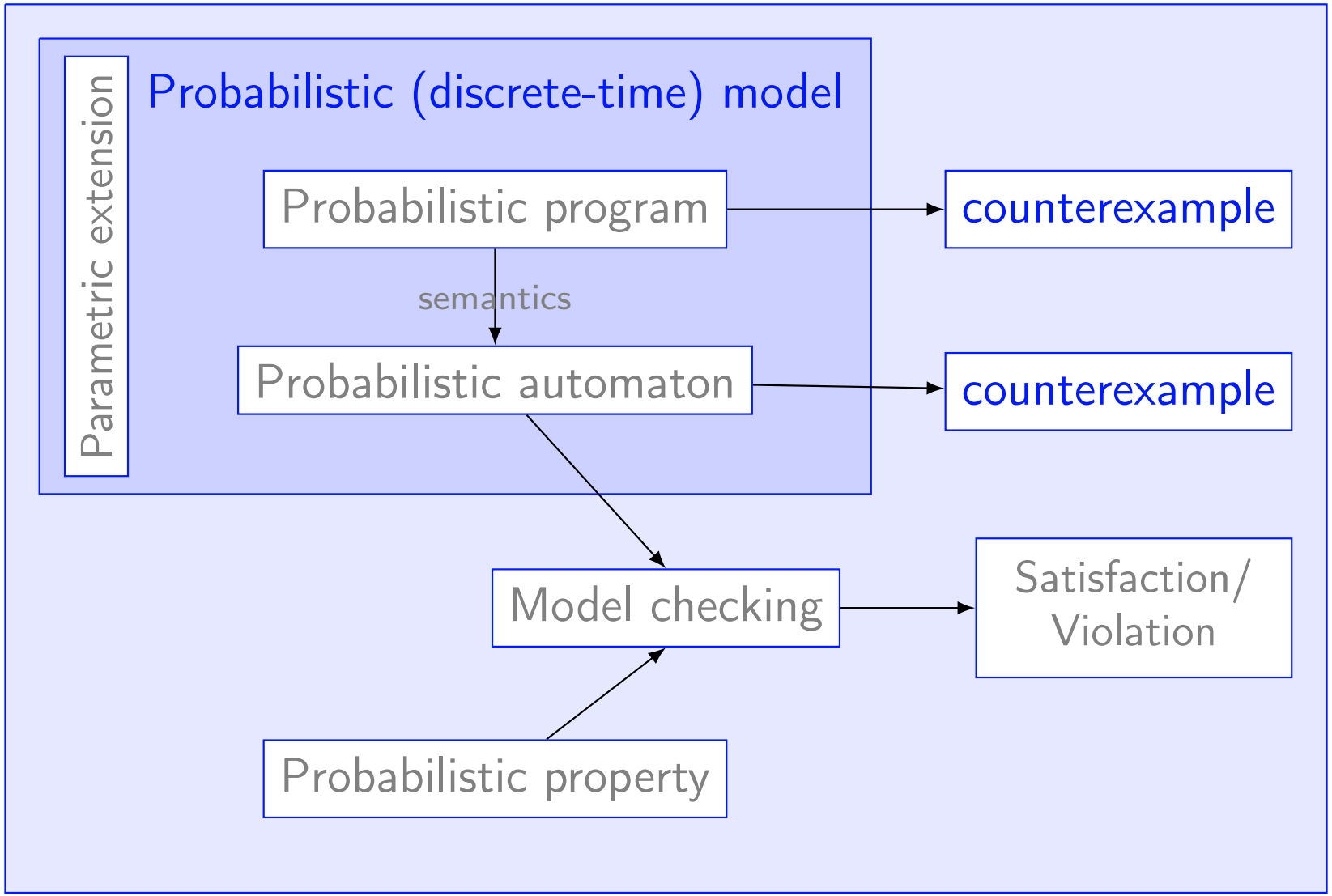
Graph		SCC MC		STATE ELIM		PARAM	
States	Trans.	Time	Mem	Time	Mem	Time	Mem
7393	11207	8.35	114.60	17.02	255.13	5.00	10.67
14323	21567	39.71	366.79	59.60	926.33	15.26	16.89
21253	31927	100.32	795.31	121.40	2050.67	29.51	24.45
28183	42287	208.41	1405.16	218.85	3708.27	50.45	30.47
78334	121512	639.29	3785.11	—	MO	1138.82	111.58

MO: 4GB

Contents


Parametric extension
Probabilistic (discrete-time) model
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property

Contents


Probabilistic (discrete-time) model
Parametric extension
Probabilistic program
counterexample
semantics
Probabilistic automaton
counterexample
Model checking
Satisfaction/
Violation
Probabilistic property