Probabilistic Counterexamples

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Ralf Wimmer

Albert-Ludwigs-Universität Freiburg, Germany
Saarland University, Saarbrücken, Germany

2nd AVACS Autumn School, Oldenburg, October 2, 2015

Acknowledgements

Most of what I present is joint work with

- Nils Jansen,
- Erika Ábrahám,
- Joost-Pieter Katoen, and
- Bernd Becker.

Introduction

Motivation, Foundations

Dave Parker did a good job yesterday, motivating the relevance of probabilistic systems and laying the foundations for counterexamples!

- Here: only a short reminder of the central notions.

Reminder: Probabilistic Model Checking

Discrete-time Markov Chains (DTMCs)

UNI FREIBURG

Definition: DTMCs

Let AP be a finite set of atomic propositions. A **discrete-time Markov chain** M is a tuple $M = (S, s_{\text{init}}, P, L)$ such that

- S is a finite set of states,
- $s_{\text{init}} \in S$ the initial state,
- $P : S \times S \to [0,1]$ the transition probability matrix with $\sum_{s' \in S} P(s,s') \leq 1$ for all $s \in S$, and
- $L : S \to 2^{AP}$ a labeling function, assigning the set of true propositions to each state.

Reminder: Probabilistic Model Checking

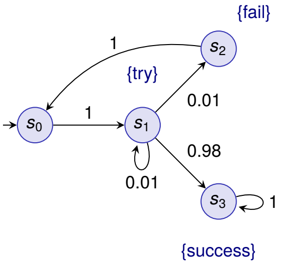
Discrete-time Markov Chains (DTMCs)

UNI FREIBURG

Definition: DTMCs

Let AP be a finite set of atomic propositions. A **discrete-time Markov chain** M is a tuple $M = (S, s_{\text{init}}, P, L)$ such that

- S is a finite set of states,
- $s_{\text{init}} \in S$ the initial state,
- $P : S \times S \rightarrow [0,1]$ the transition probability matrix with $\sum_{s' \in S} P(s,s') \leq 1$ for all $s \in S$, and
- $L : S \rightarrow 2^{AP}$ a labeling function, assigning the set of true propositions to each state.



Reminder: Probabilistic Model Checking

Discrete-time Markov Decision Processes (MDPs)

Definition: MDPs

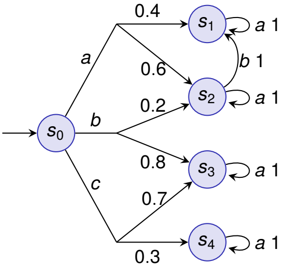
Let AP be a set of atomic propositions. A **discrete-time Markov decision process** M is a tuple $M = (S, s_{\text{init}}, A, P, L)$ such that

- S, s_{init}, and L are as for DTMCs,
- A is a finite set of actions, and
- $P : S \times A \times S \to [0,1]$ is a transition probability matrix such that $\sum_{s' \in S} P(s, \alpha, s') \in \{0,1\}$ for all $s \in S$ and $\alpha \in A$.

Definition: MDPs

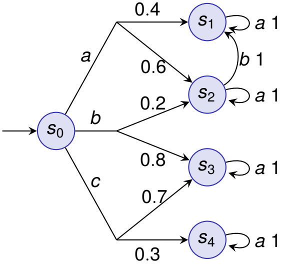
Let AP be a set of atomic propositions. A **discrete-time Markov decision process** M is a tuple $M = (S, s_{\text{init}}, A, P, L)$ such that

- S, s_{init}, and L are as for DTMCs,
- A is a finite set of actions, and
- $P : S \times A \times S \to [0,1]$ is a transition probability matrix such that $\sum_{s' \in S} P(s, \alpha, s') \in \{0,1\}$ for all $s \in S$ and $\alpha \in A$.



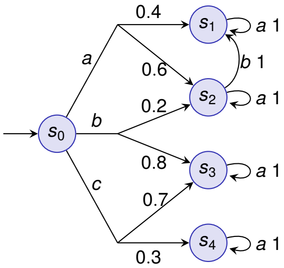
Scheduler

- The non-determinism is resolved by a scheduler.
- It assigns to each finite path a distribution over the actions possible in the last state.



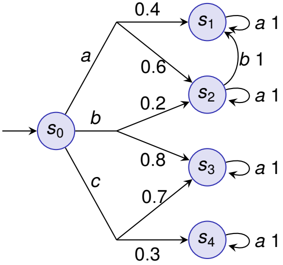
Scheduler

- The non-determinism is resolved by a scheduler.
- It assigns to each finite path a distribution over the actions possible in the last state.
- A deterministic scheduler puts the whole probability into a unique action-distribution pair.
- The decisions made by a memoryless scheduler depend only on the last state of the path.



Scheduler

- The non-determinism is resolved by a scheduler.
- It assigns to each finite path a distribution over the actions possible in the last state.
- A deterministic scheduler puts the whole probability into a unique action-distribution pair.
- The decisions made by a memoryless scheduler depend only on the last state of the path.



Each scheduler for an MDP/PA induces a DTMC.

UNI FREIBURG

Safety of DTMCs

Is the probability to eventually enter an unsafe state (labeled with “unsafe”) at most λ?

$$\mathcal{P}_{\leq\lambda}(\mathcal{F}\,\text{unsafe})$$

Probability computation

Solve the following linear equation system:

$$x_s = \begin{cases} 1 & \text{if } s \vDash \text{unsafe}, \\ 0 & \text{if all unsafe states are unreachable from } s, \\ \sum_{s' \in S} P(s,s') \cdot x_{s'} & \text{otherwise.} \end{cases}$$

Safety of MDPs

Is the maximal probability to reach an unsafe state at most λ?

Safety of MDPs

Is the maximal probability to reach an unsafe state at most λ?

Probability computation

Solve the following linear program:

$$\begin{aligned}
\text{minimize} \quad & \sum_{s \in S} x_s \\
\text{such that} \quad & \\
\text{for } s \in T: \quad & x_s = 1 \\
\text{for } s \text{ with } T \text{ unreachable}: \quad & x_s = 0 \\
\text{otherwise, for } s \in S, a \in A: \quad & x_s \geq \sum_{s' \in S} P(s, \alpha, s') \cdot x_{s'}
\end{aligned}$$

Non-probabilistic Systems

Examples:

- digital circuits
- software
- hybrid systems

Safety: The system will never enter an unsafe state.

Counterexample: Trace (sequence of inputs and successor states) leading from the initial state to an unsafe state.

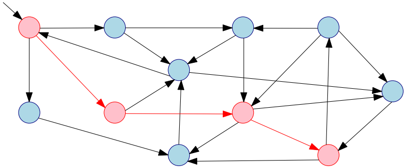
Obtaining Non-probabilistic Cex

UNI FREIBURG

By-product of model checking:

- **bounded model checking (BMC):** Satisfying assignment of the BMC-formula corresponds to a counterexample
- **state space traversal (explicit):** Store the current path during depth-first search
- **state space traversal (symbolic):** Store intermediate state sets during forward traversal and extract a cex by walking backward.
- **LTL model checking:** Accepting run of the Büchi automaton

Why Counterexamples?



"It is impossible to overestimate the importance of the counterexample feature. The counterexamples are invaluable in debugging complex systems. Some people use model checking just for this feature."

Edmund Clarke, Turing-Award Winner 2007

Applications of cex:

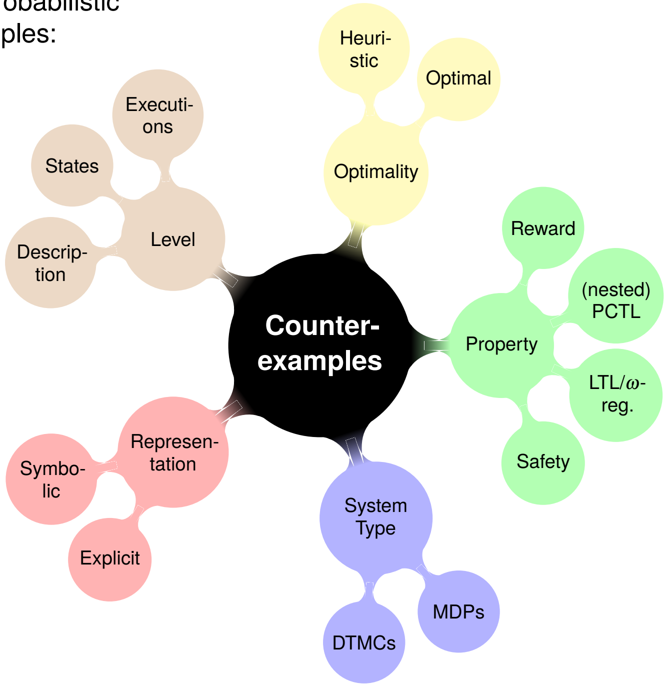
- System debugging (fault reproduction / diagnosis)
- Counterexample-guided abstraction refinement (CEGAR)

Counterexamples for Probabilistic Systems

Challenges:

- Algorithms only yield probabilities, but no counterexamples.
- A single trace to an error state typically does not suffice.

Aspects of probabilistic counterexamples:



Overview

Introduction
- Probabilistic Model Checking
- Non-Probabilistic Counterexamples

Path-based Counterexamples

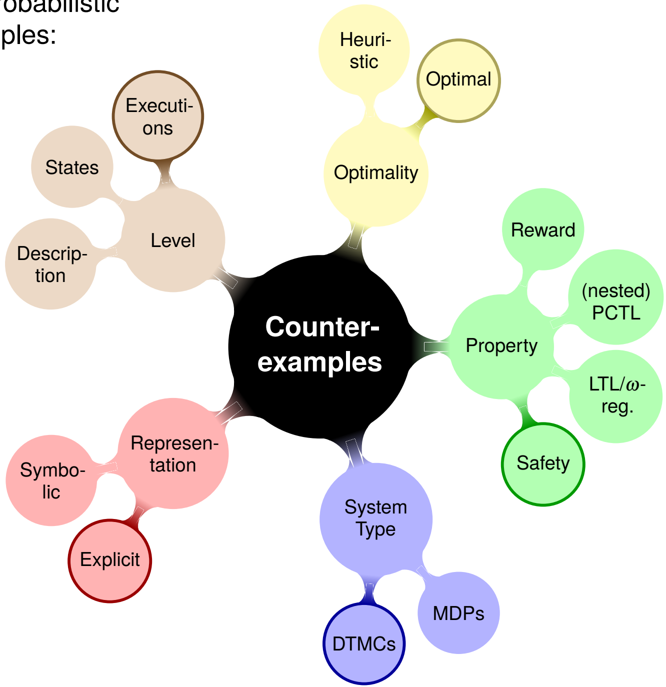
Computation of Minimal Critical Subsystems

Symbolic Computation of Critical Subsystems

High-level counterexamples

Path-based Counterexamples

Aspects of probabilistic counterexamples:



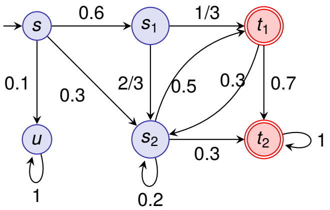
Adaptation of Non-Probabilistic Cex

- **Non-prob. cex:** 1 trace
- **Prob. cex:** set of traces with enough probability

Adaptation of Non-Probabilistic Cex

- **Non-prob. cex:** 1 trace
- **Prob. cex:** set of traces with enough probability

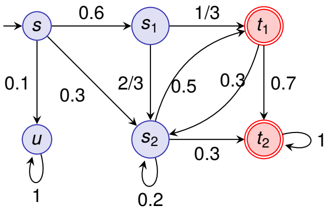
$$\mathcal{P}_{\leq 0.5}(\mathcal{F}\, unsafe)$$



Adaptation of Non-Probabilistic Cex

- **Non-prob. cex:** 1 trace
- **Prob. cex:** set of traces with enough probability

$$\mathcal{P}_{\leq 0.5}(\mathcal{F}\, unsafe)$$



Counterexample:

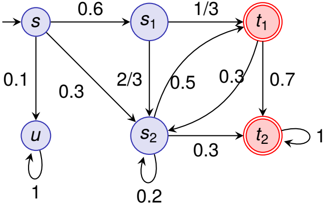
- $s \to s_1 \to t_1$
  $s \to s_1 \to s_2 \to t_1$
  $s \to s_1 \to s_2 \to t_2$
  Prob: 0.52

Definitions

Consider a violated safety property $\mathcal{P}_{\leq\lambda}(\mathcal{F}\,\text{unsafe})$.

- **Evidence:** Any finite path π starting in s_{init} and ending upon the first visit of an unsafe state.
- **Strongest evidence:** evidence π^* such that $\Pr(\pi^*) \geq \Pr(\pi)$ for all evidences π.
- **Counterexample:** Set C of evidences such that $\Pr(C) > \lambda$
- **Minimal couterexample:** Counterexample C^* such that $|C^*| \leq |C|$ for all cex C.
- **Smallest counterexample:** Counterexample C^* such that $\Pr(C^*) \geq \Pr(C)$ for all minimal cex C.

Example



Evidences:

- $s \to s_1 \to t_1$, prob = 0.2
- $s \to s_1 \to s_2 \to t_1$, prob = 0.2
- $s \to s_2 \to t_1$, prob = 0.15
- $s \to s_1 \to s_2 \to t_2$, prob = 0.12
- $s \to s_2 \to t_2$, prob = 0.09

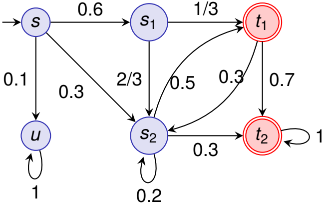
No evidences:

- $s_1 \to s_2 \to t_1$
- $s \to s_1 \to t_1 \to t_2$

Strongest evidences:

- $s \to s_1 \to t_1$
- $s \to s_1 \to s_2 \to t_1$

Example

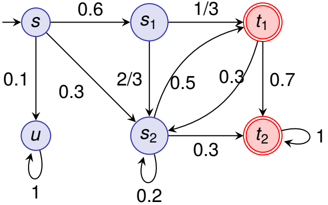


$$\mathcal{P}_{\leq 0.5}(\mathcal{F}\,\text{unsafe})$$

Counterexamples:

- $s \to s_1 \to t_1$
  $s \to s_1 \to s_2 \to t_1$
  $s \to s_1 \to t_1$
  Prob: 0.55
- $s \to s_1 \to t_1$
  $s \to s_1 \to s_2 \to t_1$
  $s \to s_1 \to s_2 \to t_2$
  Prob: 0.52
- $s \to s_1 \to s_2 \to t_1$
  $s \to s_1 \to s_2 \to t_2$
  $s \to s_2 \to t_1$
  $s \to s_2 \to t_2$
  Prob: 0.56

Example

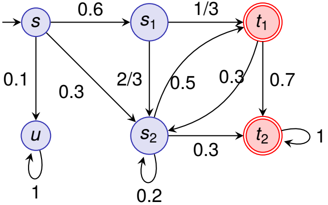


$$\mathcal{P}_{\leq 0.5}(\mathcal{F}\,\text{unsafe})$$

Minimal Counterexamples:

- $s \to s_1 \to t_1$
  $s \to s_1 \to s_2 \to t_1$
  $s \to s_1 \to t_1$
  Prob: 0.55
- $s \to s_1 \to t_1$
  $s \to s_1 \to s_2 \to t_1$
  $s \to s_1 \to s_2 \to t_2$
  Prob: 0.52

Example



$$\mathcal{P}_{\leq 0.5}(\mathcal{F}\,\text{unsafe})$$

Smallest Counterexamples:

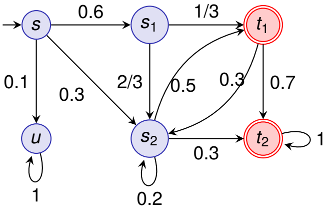
- $s \rightarrow s_1 \rightarrow t_1$
  $s \rightarrow s_1 \rightarrow s_2 \rightarrow t_1$
  $s \rightarrow s_1 \rightarrow t_1$
  Prob: 0.55

Computation of Smallest Cex

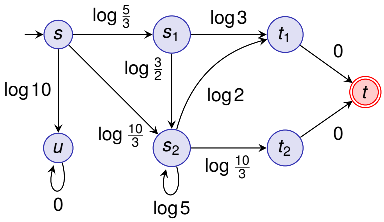
Transformation into a shortest-paths problem:

1. Add a single deadlock target state t; redirect all out-going transitions from unsafe states to t
2. Define weighted digraph $G = (S, E, w)$:

$$(s,s') \in E \;\Leftrightarrow\; P(s,s') > 0 \qquad \text{and} \qquad w(s,s') = -\log P(s,s')$$



$\rightarrow$



Shortest Paths

Lemma

The k shortest path from s_{init} to t in the weighted digraph corresponds to the k-most probable evidence in the DTMC.

Shortest Paths

Lemma

The k shortest path from s_{init} to t in the weighted digraph corresponds to the k-most probable evidence in the DTMC.

The computation of a smallest cex is a ***k*-shortest paths** problem in a weighted digraph with non-negative weights.

Available Algorithms:

- Eppstein (SIAM J. Comput., 1998)
- Jiménez/Marzal (Proc. of WAE, 1999)
- K* by Aljazzar/Leue (Artif. Intell., 2011)

Challenges

Counterexample = k shortest paths

Does this solve the counterexample problem?

Clearly: **NO!**

Limiting factors:

- size of the DTMC
- size of the path set
- models with non-determinism (MDPs)

Challenges

UNI FREIBURG

Counterexample = k shortest paths

Does this solve the counterexample problem?

Clearly: **NO!**

Limiting factors:

- *size of the DTMC*
  - sometimes millions or billions of states
- size of the path set

- models with non-determinism (MDPs)

Challenges

Counterexample = k shortest paths

Does this solve the counterexample problem?

Clearly: **NO!**

Limiting factors:

- *size of the DTMC*
  - sometimes millions or billions of states
- *size of the path set*
  - number of paths often larger than the number of states
- models with non-determinism (MDPs)

Challenges

Counterexample = k shortest paths

Does this solve the counterexample problem?

Clearly: **NO!**

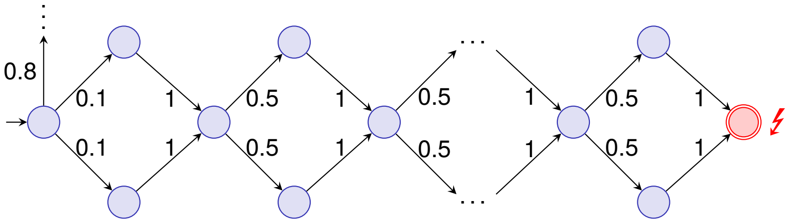
Limiting factors:

- *size of the DTMC*
  - ▶ sometimes millions or billions of states
- *size of the path set*
  - ▶ number of paths often larger than the number of states
- *models with non-determinism (MDPs)*
  - ▶ all paths must resolve the non-determinism in the same way

Size of Counterexamples

Property:

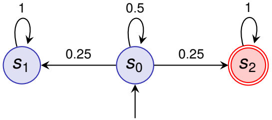
$$\mathcal{P}_{\leq 0.15}(\mathcal{F}\, unsafe)$$



- Probability of each path: $0.1 \cdot (0.5)^{n-1}$
- Number of paths: 2^n (n = number of branchings)
- Number of paths needed: $\frac{0.15}{0.2} \cdot 2^n + 1$

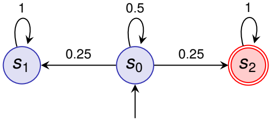
$\Rightarrow$ exponential in the number of states.

Counterexamples can be even infinite sets



Property: $\mathcal{P}_{<0.5}(\mathcal{F}\, unsafe)$

Counterexamples can be even infinite sets



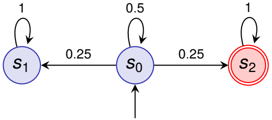
Property: $\mathcal{P}_{<0.5}(\mathcal{F}\,\mathit{unsafe})$

Consider set C of all paths leading to state s_2:

$$C = \{(s_0) \to s_2,\ \ (s_0)^2 \to s_2,\ \ (s_0)^3 \to s_2, \ldots\}$$

Probability of C: $\sum_{i=0}^{\infty}(0.5)^i \cdot 0.25$

Counterexamples can be even infinite sets



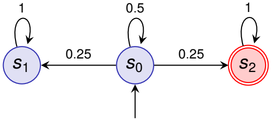
Property: $\mathcal{P}_{<0.5}(\mathcal{F}\,\mathit{unsafe})$

Consider set C of all paths leading to state s_2:

$$C = \{(s_0) \rightarrow s_2,\ \ (s_0)^2 \rightarrow s_2,\ \ (s_0)^3 \rightarrow s_2, \ldots\}$$

Probability of C: $\sum_{i=0}^{\infty}(0.5)^i \cdot 0.25 \overset{\text{geom. ser.}}{=} \frac{1}{1-0.5} \cdot 0.25$

Counterexamples can be even infinite sets



Property is violated!

Property: $\mathcal{P}_{<0.5}(\mathcal{F}\, unsafe)$

Consider set C of all paths leading to state s_2:

$$C = \{(s_0) \to s_2,\ \ (s_0)^2 \to s_2,\ \ (s_0)^3 \to s_2, \ldots\}$$

Probability of C: $\sum_{i=0}^{\infty}(0.5)^i \cdot 0.25 \overset{\text{geom. ser.}}{=} \frac{1}{1-0.5} \cdot 0.25 = 0.5$

Representation of prob. cex

Counterexamples can be represented

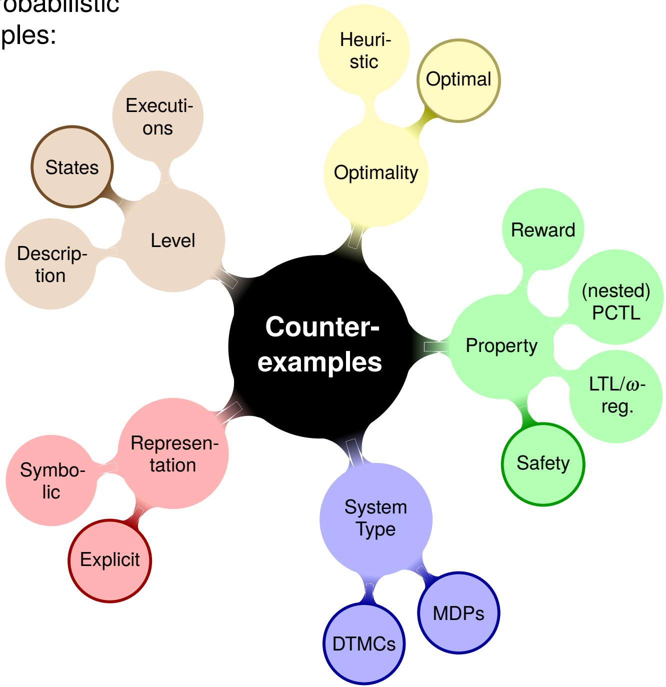
- by enumeration of the paths,
- by regular expressions, trees, ...
- critical subsystems [Aljazzar/Leue, 2009; Jansen et al., 2011].

Critical subsystem

Subset S' of the states such that the probability of reaching an unsafe-state **visiting only states from** S' is already beyond λ.

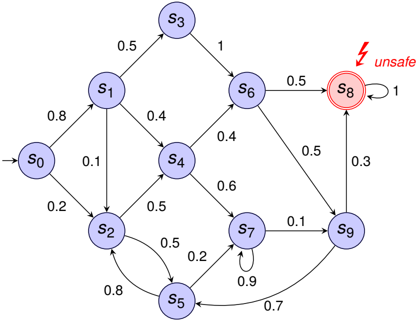
Computation of Minimal Critical Subsystems

Aspects of probabilistic counterexamples:



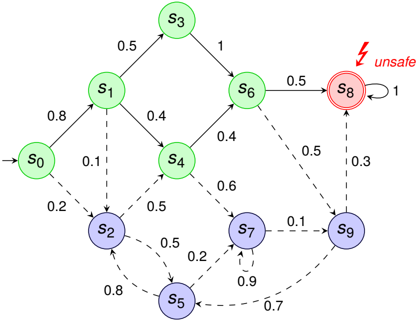
Critical subsystems for DTMCs: Example

$$\mathcal{P}_{\leq 0.25}(\mathcal{F}\,\mathit{unsafe})$$



Critical subsystems for DTMCs: Example

$$\mathcal{P}_{\leq 0.25}(\mathcal{F}\,\mathit{unsafe})$$



Minimal Critical Subsystems

Formulate minimal critical subsystems as an optimization problem:

- 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

Minimal Critical Subsystems

Formulate minimal critical subsystems as an optimization problem:

- 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

Mixed-integer linear program

$$\begin{aligned}
\text{minimize} \quad & \left(-\tfrac{1}{2}p_{s_{\text{init}}} + \sum_{s\in S} x_s\right) \\
\text{such that} \quad & \\
& p_{s_{\text{init}}} > \lambda \\
\forall s \in T: \quad & x_s = p_s \\
\forall s \in S \setminus T: \quad & p_s \leq x_s \\
\forall s \in S \setminus T: \quad & p_s \leq \sum_{s'\in S} P(s,s') \cdot p_{s'}
\end{aligned}$$

Optimizations

The computation time can be reduced by adding redundant constraints:

- Each state (except s_{init}) has a predecessor state in the subsystem
- Each state (except unsafe states) has a successor state in the subsystem
- Generalize this to strongly connected components
- Require that each state in the subsystem is reachable from s_{init}
- Require that each state in the subsystem can reach an unsafe state

► **Trade-off between additional constraints and size of search space**

Some results for DTMCs

Benchmarks:

- Crowds protocol
  - Ramdomized protocol for anonymous surfing
- Synchronous leader election
  - Randomized protocol to select a unique leader in a symmetric ring of computers.

Experimental setup:

- Time limit: 2 hours
- Memory limit: 4 GB
- Solver: Gurobi 6

Some results for DTMCs

Model	$\lvert S \rvert$	$\lvert E_M \rvert$	$\lvert T \rvert$	λ	$\lvert S_{\text{MCS}} \rvert$	$\lvert E_{\text{MCS}} \rvert$	Time
crowds2-3	183	243	26	0.09	22	27	0.06 (0.11)
crowds2-4	356	476	85	0.09	22	27	0.30 (0.24)
crowds2-5	612	822	196	0.09	22	27	0.56 (0.24)
crowds3-3	396	576	37	0.09	37	51	0.38 (0.30)
crowds3-4	901	1321	153	0.09	37	51	0.89 (0.58)
crowds3-5	1772	2612	425	0.09	37	51	1.51 (0.87)
crowds5-4	3515	6035	346	0.09	72	123	12.51 (4.89)
crowds5-6	18817	32677	3710	0.09	72	123	100.26 (23.52)
crowds5-8	68740	120220	19488	0.09	72	123	1000.79 (145.84)
leader3-2	22	29	1	0.5	15	18	0.21 (0.13)
leader3-3	61	87	1	0.5	33	45	0.02 (0.06)
leader3-4	135	198	1	0.5	70	101	0.07 (0.09)
leader4-2	55	70	1	0.5	34	41	0.24 (0.17)
leader4-3	256	336	1	0.5	132	171	0.49 (0.37)
leader4-4	782	1037	1	0.5	395	522	1.88 (1.21)
leader4-5	1889	2513	1	0.5	946	1257	4.06 (2.80)
leader4-6	3902	5197	1	0.5	1953	2600	8.70 (5.92)

MILP formulation for MDPs

MILP formulation for MDPs

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

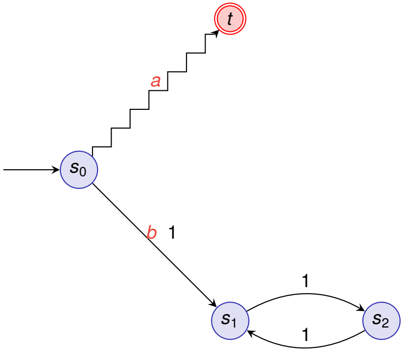
MILP formulation for MDPs

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

$$
\begin{aligned}
\text{minimize} \quad & -\tfrac{1}{2} p_{s_{\text{init}}} + \textstyle\sum_{s \in S} x_s \\
\text{such that} \quad & \\
& p_{s_{\text{init}}} > \lambda \\
\textit{targets}: \quad & x_s = p_s \\
\textit{non-target } s: \quad & p_s \leq x_s \qquad x_s = \textstyle\sum_{a \in A} \sigma_{s,a} \\
\textit{non-target } s,\ \textit{action } a: \quad & p_s \leq (1 - \sigma_{s,a}) + \textstyle\sum_{s' \in S} P(s,a,s') \cdot p_{s'}
\end{aligned}
$$

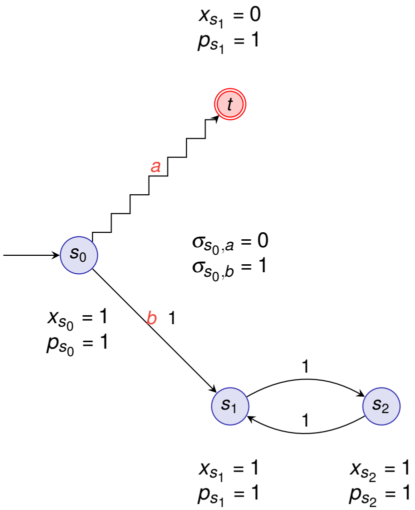
MILP formulation for MDPs: Problematic states

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



MILP formulation for MDPs: Problematic states

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



MILP formulation for MDPs

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

minimize	$-\frac{1}{2} p_{s_\text{init}} + \sum_{s \in S} x_s$
such that	
	$p_{s_\text{init}} > \lambda$
targets :	$x_s = p_s$
non-target s :	$p_s \leq x_s \qquad x_s = \sum_{a \in A} \sigma_{s,a}$
non-target s, action a :	$p_s \leq (1 - \sigma_{s,a}) + \sum_{s' \in S} P(s,a,s') \cdot p_{s'}$
probl. s, s′ ∈succ(s,a) :	$\boldsymbol{2t_{s,s'} \leq x_s + x_{s'}}$ $\boldsymbol{r_s < r_{s'} + (1 - t_{s,s'})}$ $\boldsymbol{(1 - x_s) + (1 - \sigma_{s,a}) + \sum_{s' \in succ(s,a)} t_{s,s'} \geq 1}$

Some results for MDPs

Model	$\lvert S \rvert$	$\lvert E \rvert$	prob.	λ	$\lvert S_{min} \rvert$	basic	best opt.
consensus-2-2	272	400	1	0.1	15	– TO – ($\geq$ 8)	2 167
consensus-2-4	528	784	1	0.1	$\leq$ 35	– TO – ($\geq$ 9)	– TO – ($\geq$ 12)
csma-2-2	1 038	1 054	1	0.1	195	– TO – ($\geq$ 184)	638
csma-2-4	7 958	7 988	1	0.1	410	– TO – ($\geq$ 408)	1 342
csma-2-6	66 718	66 788	1	0.1	415	2 364	2 364
aleader-3	364	573	1	0.5	$\leq$ 66	– TO – ($\geq$ 18)	– TO – ($\geq$ 27)
aleader-4	3 172	6 252	1	0.5	$\leq$ 215	– TO – ($\geq$ 10)	– TO – ($\geq$ 10)

Extensions of the MILP approach

- LTL properties both for DTMCs and MDPs
  - LTL $\rightarrow$ deterministic Rabin automaton (DRA)
  - DRA $\otimes$ DTMC/MDP $\rightarrow$ DTMC/MDP
  - Minimize projection onto the original state space

Extensions of the MILP approach

- LTL properties both for DTMCs and MDPs
  - LTL $\rightarrow$ deterministic Rabin automaton (DRA)
  - DRA $\otimes$ DTMC/MDP $\rightarrow$ DTMC/MDP
  - Minimize projection onto the original state space
- Expected reward properties

Extensions of the MILP approach

- LTL properties both for DTMCs and MDPs
  - LTL $\rightarrow$ deterministic Rabin automaton (DRA)
  - DRA $\otimes$ DTMC/MDP $\rightarrow$ DTMC/MDP
  - Minimize projection onto the original state space
- Expected reward properties
- High-level counterexamples (see last chapter)

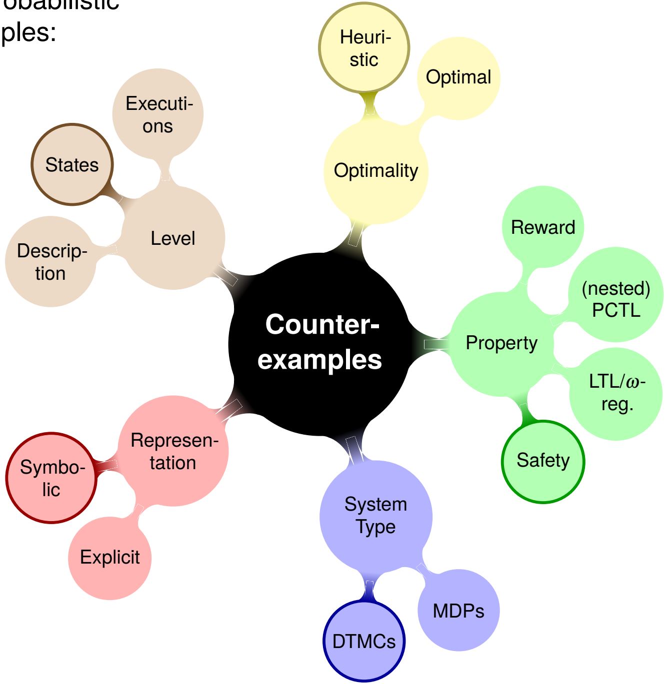
Other approaches for computing small critical subsystems

Approaches:

- heuristic search (variant of A*) (Aljazzar/Leue)
- hierarchical abstraction of SCCs (Jansen et al.)
- *symbolic methods using MTBDDs*

Symbolic Computation of Critical Subsystems

Aspects of probabilistic counterexamples:



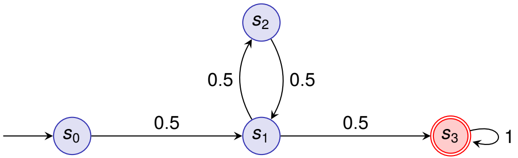
Symbolic representation of DTMCs

Multi-terminal binary decision diagrams (MTBDDs):

- directed acyclic graphs with a root node
- terminal nodes: labeled with a real number
- internal nodes: two successors, high and low, labeled with a boolean variable

Each assignment of the variables induces a path in the MTBDD to a terminal node, whose label is the function value.
▶ functions $f : \{0, 1\}^n \to \mathbb{R}$

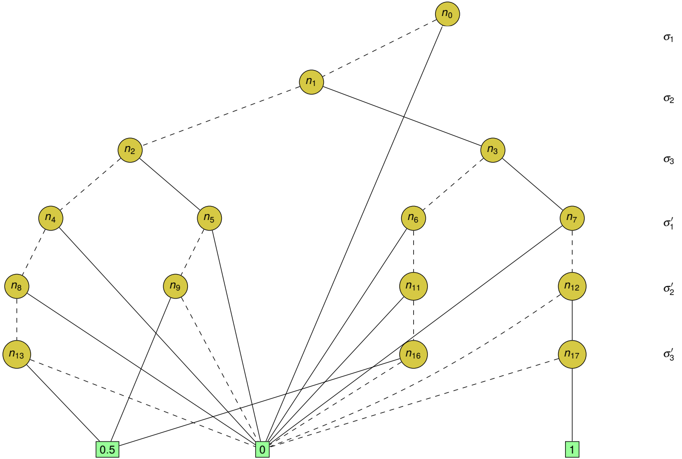
Example: DTMC



Encoding of the states:

s_0	s_1	s_2	s_3
000	001	010	011

Example: BDD-encoding



MTBDD-based representation

- often (not always) much smaller than explicit representations
- efficient algorithms for (point-wise) addition, multiplication, matrix-multiplication … available
- ▶ in practise MTBDDs allow for representing very large systems

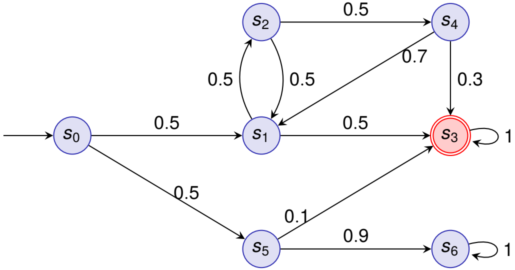
Counterexample computation using MTBDDs

Idea

- Start with the states of a most probable path from the initial to a target state
- extend the system with further paths / path fragments until it becomes a counterexample

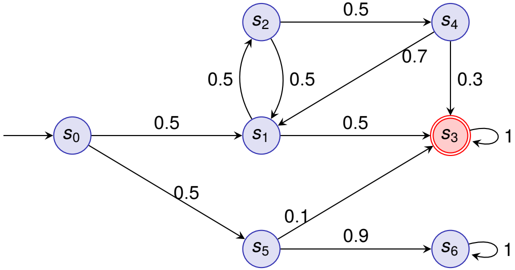
- **Global search**: all paths go from initial to target states
- **Fragment search**: paths start and end at an arbitrary state of the subsystem and contain at least one new state

Example

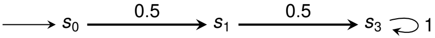


Global search:

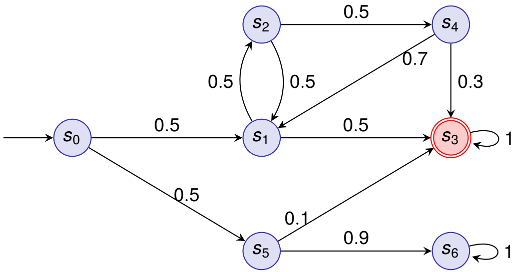
Example



Global search:



Example

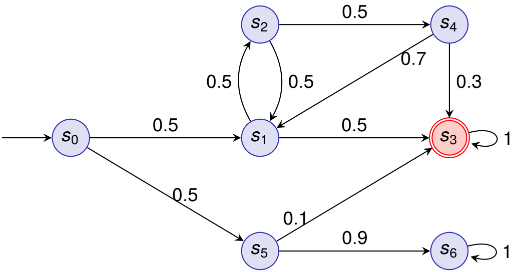


Global search:

$$\longrightarrow s_0 \xrightarrow{0.5} s_1 \xrightarrow{0.5} s_3 \circlearrowleft 1$$

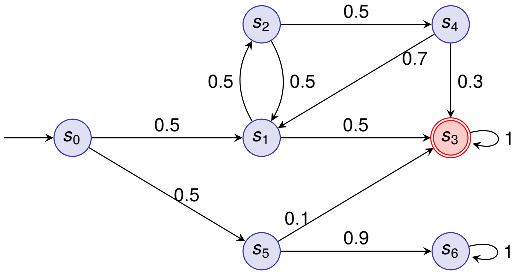
$$\longrightarrow s_0 \xrightarrow{0.5} s_1 \xrightarrow{0.5} s_2 \xrightarrow{0.5} s_1 \xrightarrow{0.5} s_3 \circlearrowleft 1$$

Example

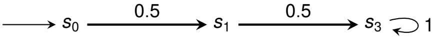


Local search:

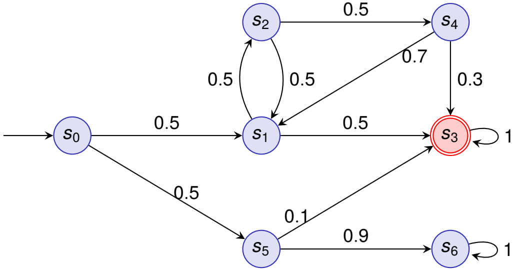
Example



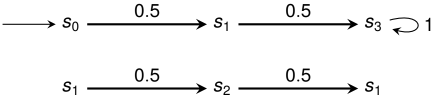
Local search:



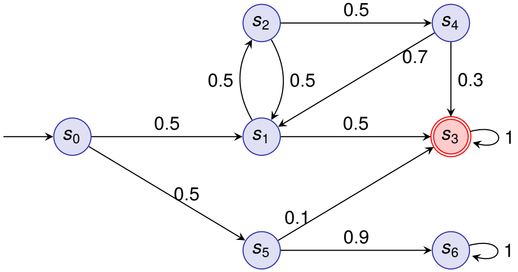
Example



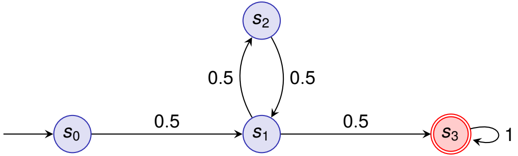
Local search:



Example: Result



Resulting subsystem:



The basic algorithm

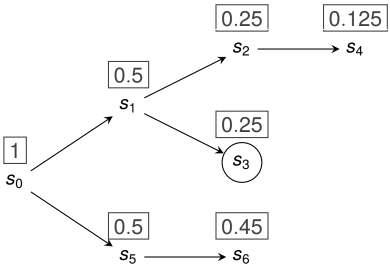
OBDD states, newStates := $\emptyset$
MTBDD subsys := $\emptyset$

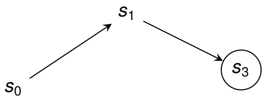
```
while modelCheck(subsys, T) ≤ λ do
    newStates := findNextPath(dtmc, Subsys);
    Subsys := Subsys ∪ newStates
end while
return Subsys
```

Finding paths

Use a symbolic version of **Dijkstra's shortest path algorithm** to find a most probable path to a target state (Siegle et al.).

- FloodingDijkstra(transitions, start set, target set)





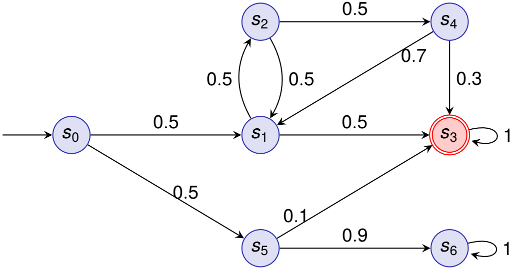
Global search

Extend the subsystem with paths from the initial to a target state

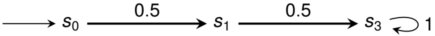
- FloodingDijkstra(transitions, init, targets)

How to exclude already found paths?

Example: Global search

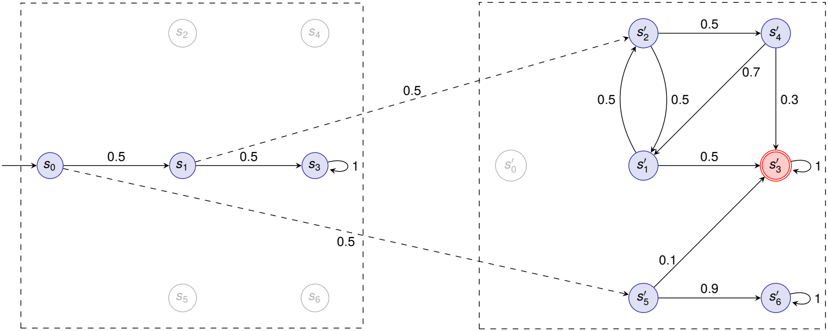


First path:



Example: Global search

Exclude all found transitions by doubling the DTMC:



Shortest path in the new graph is shortest path in the old graph containing at least one new state.

Local Search

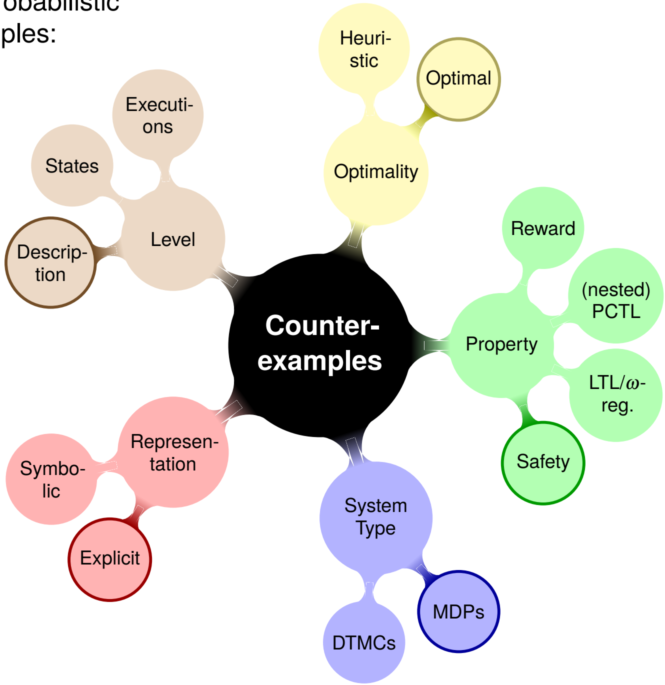
```
procedure LocalSearch(MTBDD trans, BDD init, BDD targets,
                 BDD subsys)
    if subsys= ∅ then
        return FloodingDijkstra(trans, init, targets);
    else
        subsysStates = toStateBDD(subsys);
        return FloodingDijkstra(trans \ subsys,
                                subsysStates, subsysStates);
    end if
end procedure
```

Results

- Largest instance: crowds-20-30 with $\approx 10^{16}$ states
  - $\approx$ 3000 seconds
  - 873 MB memory
  - subsystem with 76 007 states.
- Subsystem size typically not far from minimum.
- Global search slightly faster, fragment search yields slightly smaller subsystems.
- currently restricted to safety and expected reward properties of DTMCs.

High-level counterexamples

Aspects of probabilistic counterexamples:

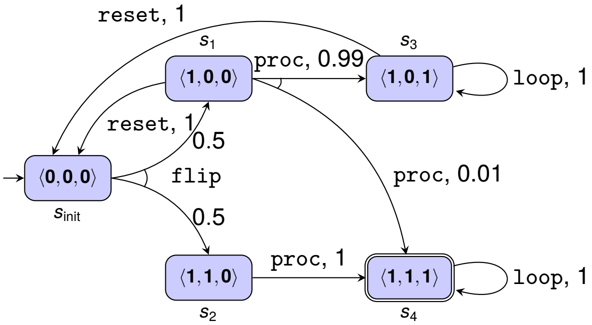


PRISM's guarded command language

module coin
f: **bool init** 0;
c: **bool init** 0;
[flip] $\neg f \rightarrow 0.5 : (f' = 1)\&(c' = 1) + 0.5 : (f' = 1)\&(c' = 0);$
[reset] $f \wedge \neg c \rightarrow 1 : (f' = 0);$
[proc] $f \rightarrow 0.99 : (f' = 1) + 0.01 : (c' = 1);$
endmodule

module processor
p: **bool init** 0;
[proc] $\neg p \rightarrow 1 : (p' = 1);$
[loop] $p \rightarrow 1 : (p' = 1);$
[reset] $\mathit{true} \rightarrow 1 : (p' = 0)$
endmodule

The induced MDP



$$\mathcal{M} \not\models \mathcal{P}_{\leq 0.5}(\Diamond (f = 1 \wedge c = 1 \wedge p = 1))$$

Counterexamples for PRISM models

Goal:

- Compute a minimal subset of the commands such that the induced system is already erroneous (**minimal critical command set**)

Counterexamples for PRISM models

Goal:

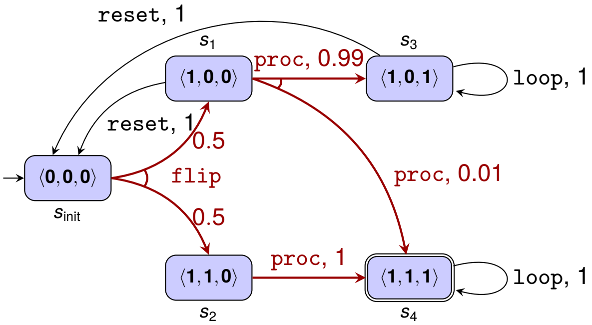
- Compute a minimal subset of the commands such that the induced system is already erroneous (**minimal critical command set**)

module coin
  f: **bool init** 0;
  c: **bool init** 0;
  [flip] $\neg f \rightarrow 0.5 : (f' = 1)\&(c' = 1) + 0.5 : (f' = 1)\&(c' = 0);$
  [reset] $f \wedge \neg c \rightarrow 1 : (f' = 0);$
  [proc] $f \rightarrow 0.99 : (f' = 1) + 0.01 : (c' = 1);$
endmodule

module processor
  p: **bool init** 0;
  [proc] $\neg p \rightarrow 1 : (p' = 1);$
  [loop] $p \rightarrow 1 : (p' = 1);$
  [reset] $true \rightarrow 1 : (p' = 0)$
endmodule

$$\mathcal{M} \not\models \mathcal{P}_{\leq 0.5}(\Diamond (f = 1 \wedge c = 1 \wedge p = 1))$$

The induced MDP



$$\mathcal{M} \not\models \mathcal{P}_{\leq 0.5}(\Diamond(f = 1 \land c = 1 \land p = 1))$$

Computation of minimal critical command sets

1. Compose the modules of the PRISM program
2. Generate the corresponding MDP
3. Label all transitions with the command(s) they are created from
4. Compute a minimal critical labeling:
   - SMT + binary search
   - Mixed integer linear programming (QEST'13)
   - **MAXSAT**

Composition and state space generation

module coin
  f: **bool init** 0;
  c: **bool init** 0;
c_1: [flip] $\neg f \rightarrow 0.5 : (f' = 1)\&(c' = 1) + 0.5 : (f' = 1)\&(c' = 0);$
c_2: [reset] $f \wedge \neg c \rightarrow 1 : (f' = 0);$
c_3: [proc] $f \rightarrow 0.99 : (f' = 1) + 0.01 : (c' = 1);$
endmodule
module processor
  p: **bool init** 0;
c_4: [proc] $\neg p \rightarrow 1 : (p' = 1);$
c_5: [loop] $p \rightarrow 1 : (p' = 1);$
c_6: [reset] $true \rightarrow 1 : (p' = 0)$
endmodule

$\Downarrow$

module `coin` $\|$ `processor`
  f: **bool init** 0;
  c: **bool init** 0;
  p: **bool init** 0;
c_1: [flip] $\neg f \rightarrow 0.5 : (f' = 1)\,\&\,(c' = 1) + 0.5 : (f' = 1)\,\&\,(c' = 0);$
c_2, c_6: [reset] $f \wedge \neg c \rightarrow 1 : (f' = 0)\,\&\,(p' = 0);$
c_3, c_4: [proc] $f \wedge \neg p \rightarrow 0.99 : (f' = 1)\,\&\,(p' = 1) + 0.01 : (c' = 1)\,\&\,(p' = 1);$
c_5: [loop] $p \rightarrow 1 : (p' = 1);$
endmodule

Composition and state space generation

module `coin` ∥ `processor`

f: **bool init** 0;

c: **bool init** 0;

p: **bool init** 0;

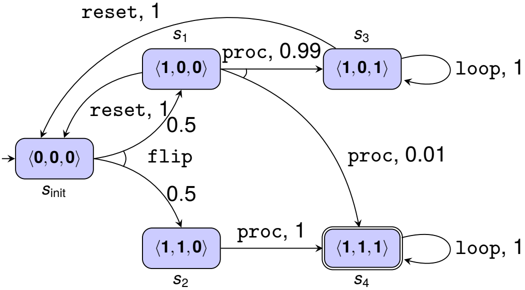
c_1: [flip] $\neg f \rightarrow 0.5 : (f' = 1) \,\&\, (c' = 1) + 0.5 : (f' = 1) \,\&\, (c' = 0)$;

c_2, c_6: [reset] $f \wedge \neg c \rightarrow 1 : (f' = 0) \,\&\, (p' = 0)$;

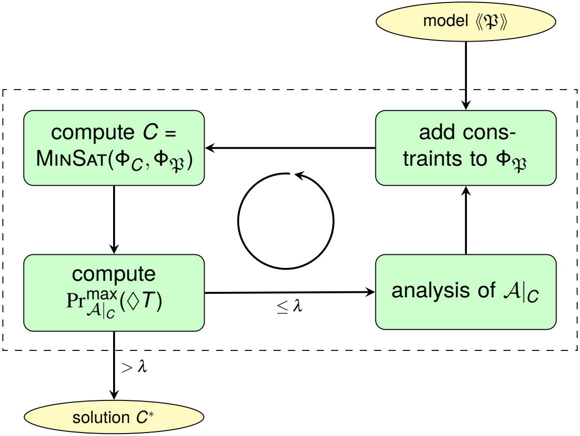
c_3, c_4: [proc] $f \wedge \neg p \rightarrow 0.99 : (f' = 1) \,\&\, (p' = 1) + 0.01 : (c' = 1) \,\&\, (p' = 1)$;

c_5: [loop] $p \rightarrow 1 : (p' = 1)$;

endmodule



Idea: MAXSAT approach



MAXSAT

UNI FREIBURG

Definition: MAXSAT

Given two sets of clauses:

- φ_h (hard constraints)
- φ_s (soft constraints)

find an assignment which satisfies **all** hard constraints and **as many** soft constraints **as possible**.

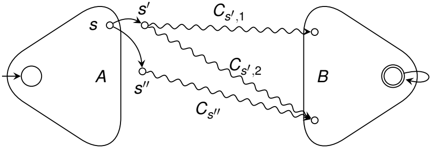
Several solvers available: MaxAntom, Z3, ...

Initial constraint system

- **Guaranteed commands:**
  Commands occuring on each path from s_{init} to T are contained in C^*.
- **Proper synchronization**:
  Each synchronizing command $c \in C^*$ needs a matching partner from each module synchronizing with c.
- **Predecessors and successors**:
  At least one state $s \in S \setminus T$, in which $c \in C^*$ is enabled needs a successor state with an activated command.

  At least one state $s \in S \setminus \{s_{\text{init}}\}$, in which $c \in C^*$ is enabled needs a predecessor state with an activated command leading to s.

Extending the constraint system



Example: T unreachable from s_{init}

- Some command appearing on an arbitrary cut between A and B must be contained in the subsystem

Evaluation

model	states	trans.	λ/p^*	comm.	$\lvert C^*\rvert$	MaxSat Time	MaxSat Mem.	MaxSat enum.
`coin`(2, 2)	272	492	0.4 / 0.56	10 (4)	9	**0.08**	**0.02**	54%
`coin`(4, 4)	43136	144352	0.4 / 0.54	20 (8)	17	**1876**	**0.07**	50%
`coin`(4, 6)	63616	213472	0.4 / 0.53	20 (8)	17	**6231**	**0.09**	50%
`coin`(6, 2)	1258240	6236736	0.4 / 0.59	30 (12)	–	TO	$>$ 1.54	–
`csma`(2, 4)	7958	10594	0.5 / 0.999	38 (21)	36	**2.26**	**0.04**	0.09%
`csma`(4, 2)	761962	1327068	0.4 / 0.78	68 (22)	53	**18272**	**0.92**	3.9E-9%
`fw`(1)	1743	2199	0.5 / 1	64 (6)	24	**16.14**	**0.05**	1.4E-10%
`fw`(10)	17190	29366	0.5 / 1	64 (6)	24	**90.47**	**0.07**	1.4E-10%
`fw`(36)	212268	481792	0.5 / 1	64 (6)	24	**1542**	**0.34**	1.4E-10%
`wlan`(0, 2)	6063	10619	0.1 / 0.184	42 (22)	33	**1.6**	**0.03**	0.02%
`wlan`(2, 4)	59416	119957	4E-4 / 7.9E-4	48 (26)	39	**50.27**	**0.07**	0.01%
`wlan`(6, 6)	5007670	11475920	1E-7 / 2.2E-7	52 (30)	43	**5035**	**3.86**	0.01%

Conclusion

- Different kinds of counterexamples available
  - path-based counterexamples
  - critical subsystems
  - critical command sets
- Both optimal and heuristic computation methods
- Symbolic methods scale relatively well to large DTMCs

Open Research Questions

So far, there are few concrete applications of probabilistic cex:

- Probabilistic CEGAR (Hermanns et al., CAV'08; Chadha/Viswanathan, TOCL 2010)
- Fault trees from counterexamples (Fischer-Leitner/Leue, IJCCBS 2013)

Open challenges:

- Demonstrate usefulness for debugging
- Application of subsystems and high-level cex in abstraction refinement
- Counterexamples for continuous-time probabilistic models
- Application for model repair.

Some References

UNI FREIBURG

Overview paper on cex:

- E. Ábrahám, B. Becker, C. Dehnert, N. Jansen, J.-P. Katoen, R. Wimmer: *Counterexample Generation for Discrete-Time Markov Models – An Introductory Survey*. Proc. of SFM, LNCS 8483, Springer 2014.

Research papers:

- T. Han, J.-P. Katoen, B. Damman: *Counterexample Generation in Probabilistic Model Checking*, IEEE Trans. on Software Engineering 35(2), 2009
- R. Wimmer, N. Jansen, E. Ábrahám, J.-P. Katoen, B. Becker: *Minimal Counterexamples for Linear-Time Probabilistic Verification*, Theoretical Computer Science 549:61–100, 2014
- N. Jansen, R. Wimmer, E. Ábrahám, B. Zajzon, J.-P. Katoen, B. Becker, and J. Schuster: *Symbolic Counterexample Generation for Large Discrete-Time Markov Chains*, Science of Computer Programming 91(A):90–114, 2014
- R. Wimmer, N. Jansen, A. Vorpahl, E. Ábrahám, J.-P. Katoen: *High-Level Counterexamples for Probabilistic Automata*, Logical Methods in Computer Science 11(1:15):1–23, 2015
- C. Dehnert, N. Jansen, R. Wimmer, E. Ábrahám, J.-P. Katoen: *Fast Debugging of PRISM Models*, Proc. of ATVA, LNCS vol. 8837, Springer 2014.

Aspects of probabilistic counterexamples: