# Synthesising Discrete Parameters in Probabilistic Models

**Sebastian Junges**

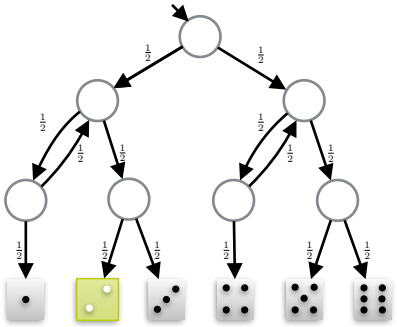
Joint work with: Roman Andriushchenko, Milan Ceska, Christian Hensel, Nils Jansen, Joost-Pieter Katoen, Simon Stupinsky

BRNO UNIVERSITY OF TECHNOLOGY

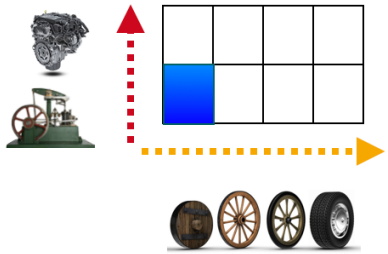
RWTH AACHEN UNIVERSITY

Radboud University

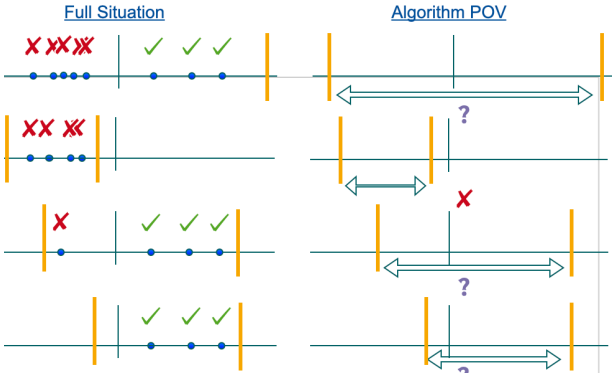
# Overview



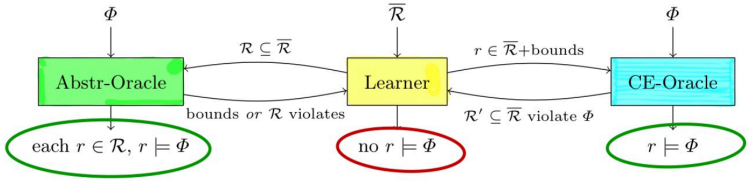
1. Probabilistic Systems & Parameter Synthesis Recap



2. Discrete setting w topology changes

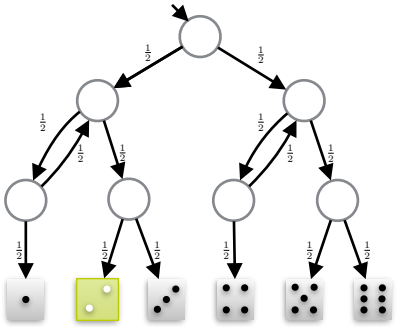



3. Various Algorithms

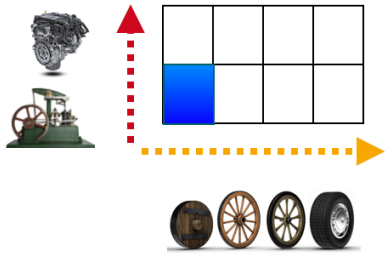



4. PAYNT and Examples

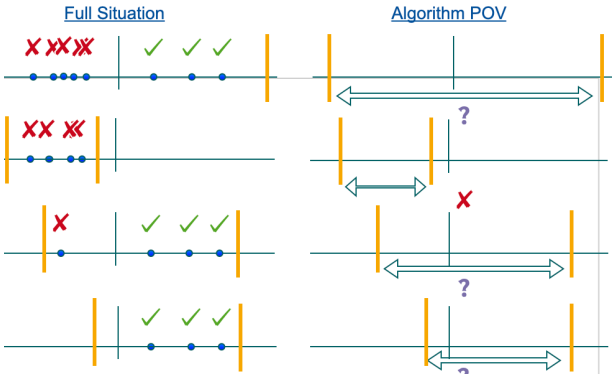
# Overview



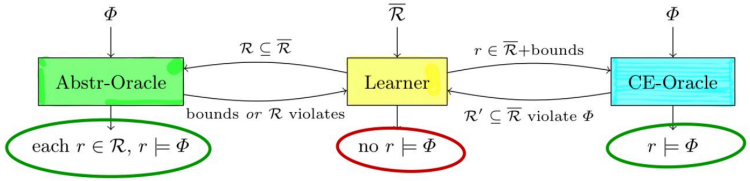
1. Probabilistic Systems & Parameter Synthesis Recap



2. Discrete setting w topology changes



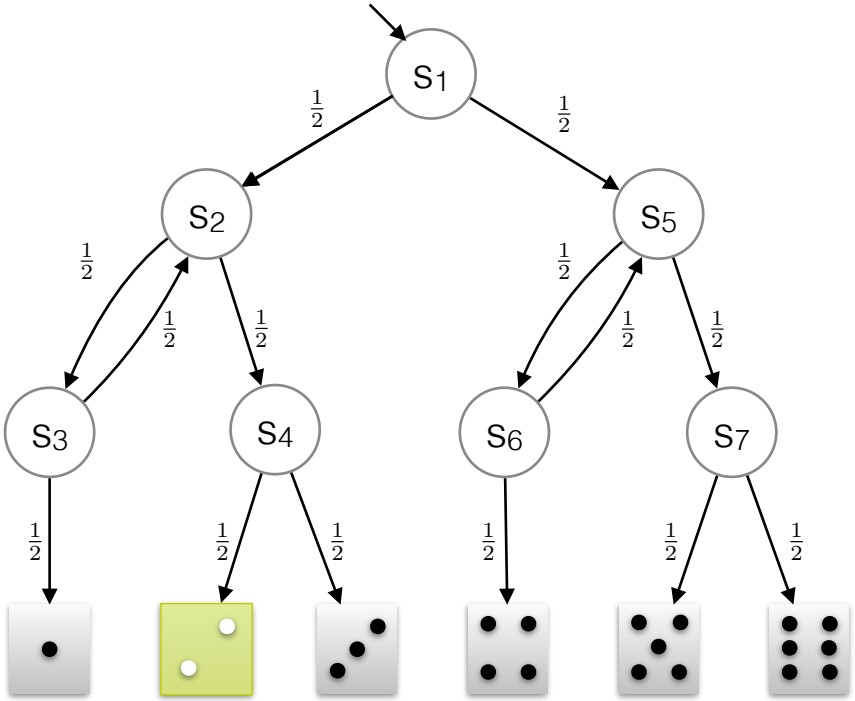

3. Various Algorithms

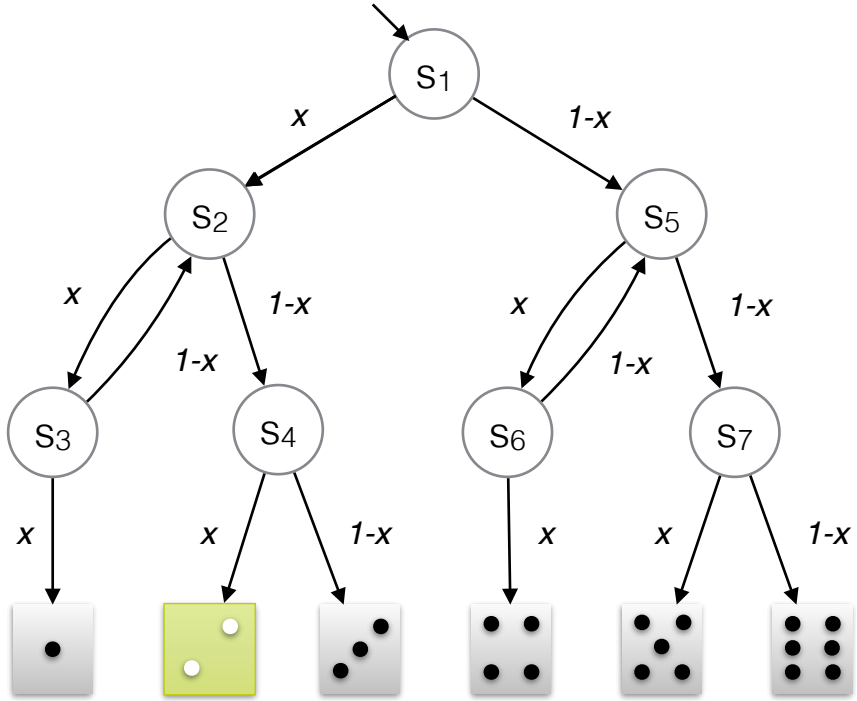



4. PAYNT and Examples

Typically $x=0$ and $x=1$ are excluded

## Markov chains with parametric probabilities




Probability to reach 

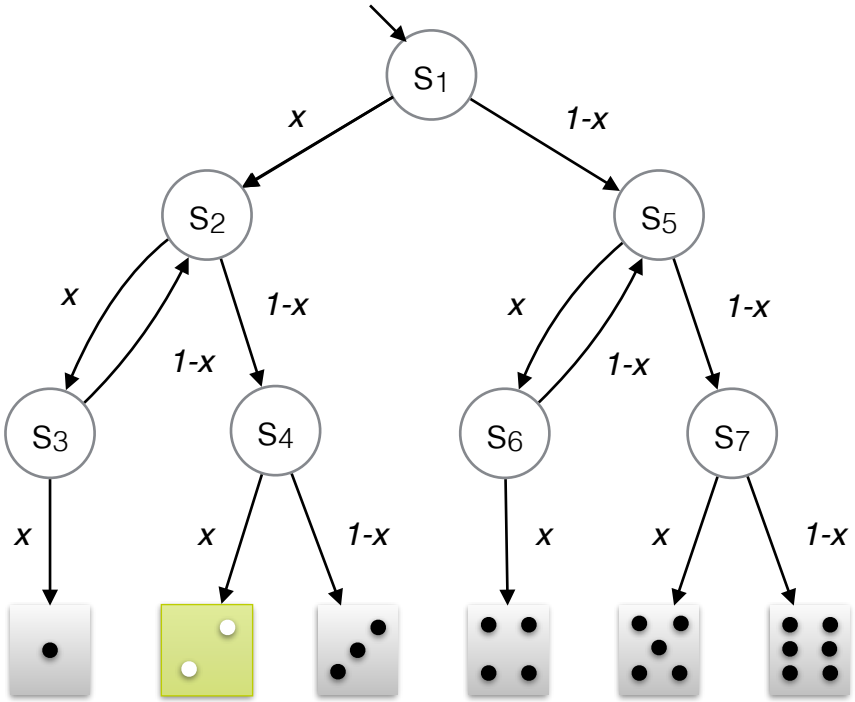
Expected # steps to 

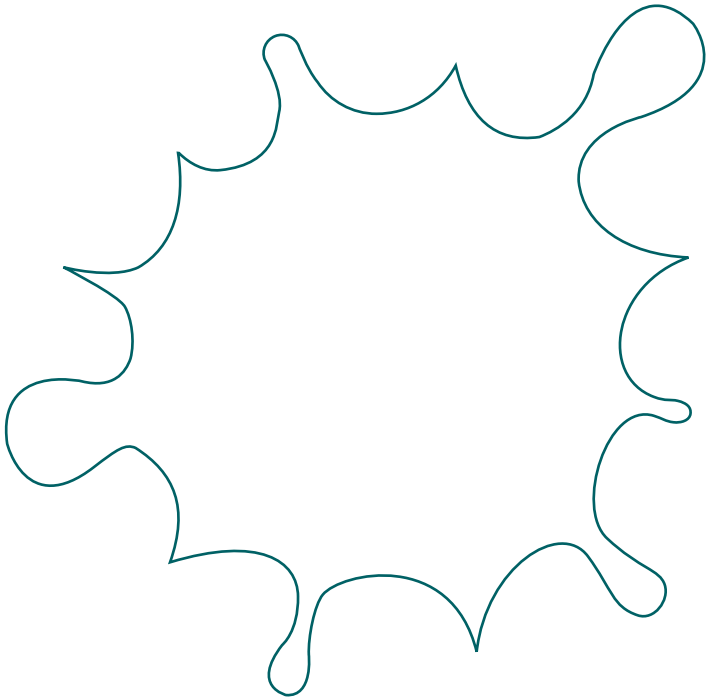
For which $x$ is probability to reach  in [9/60, 11/60]?

$(x^2 - x^3) / (x^2 - x + 1)$ in [9/60, 11/60]?

# Induced Markov chains

## Parametric Markov chains induce an infinite set of Markov chains



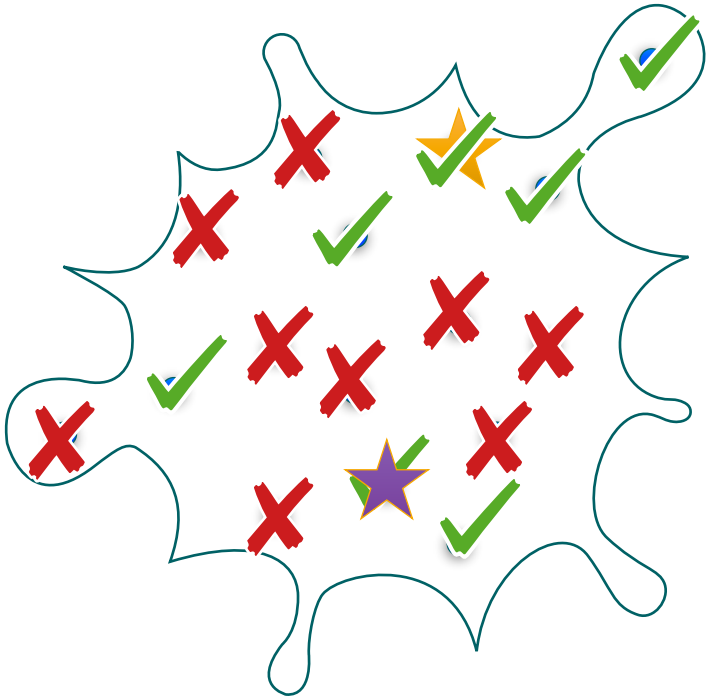
# Synthesis Goals

**Inputs:**

a pMC describing a family + a property ***f*** eg. can *G* be reached with probability > *p*?

**Synthesis goals:**

1. Find a realisation (an MC) satisfying ***f***.
2. Find all realisations satisfying ***f***.
3. Find the realisation with the maximal probability to reach *G*.



**Cost-based variants:**

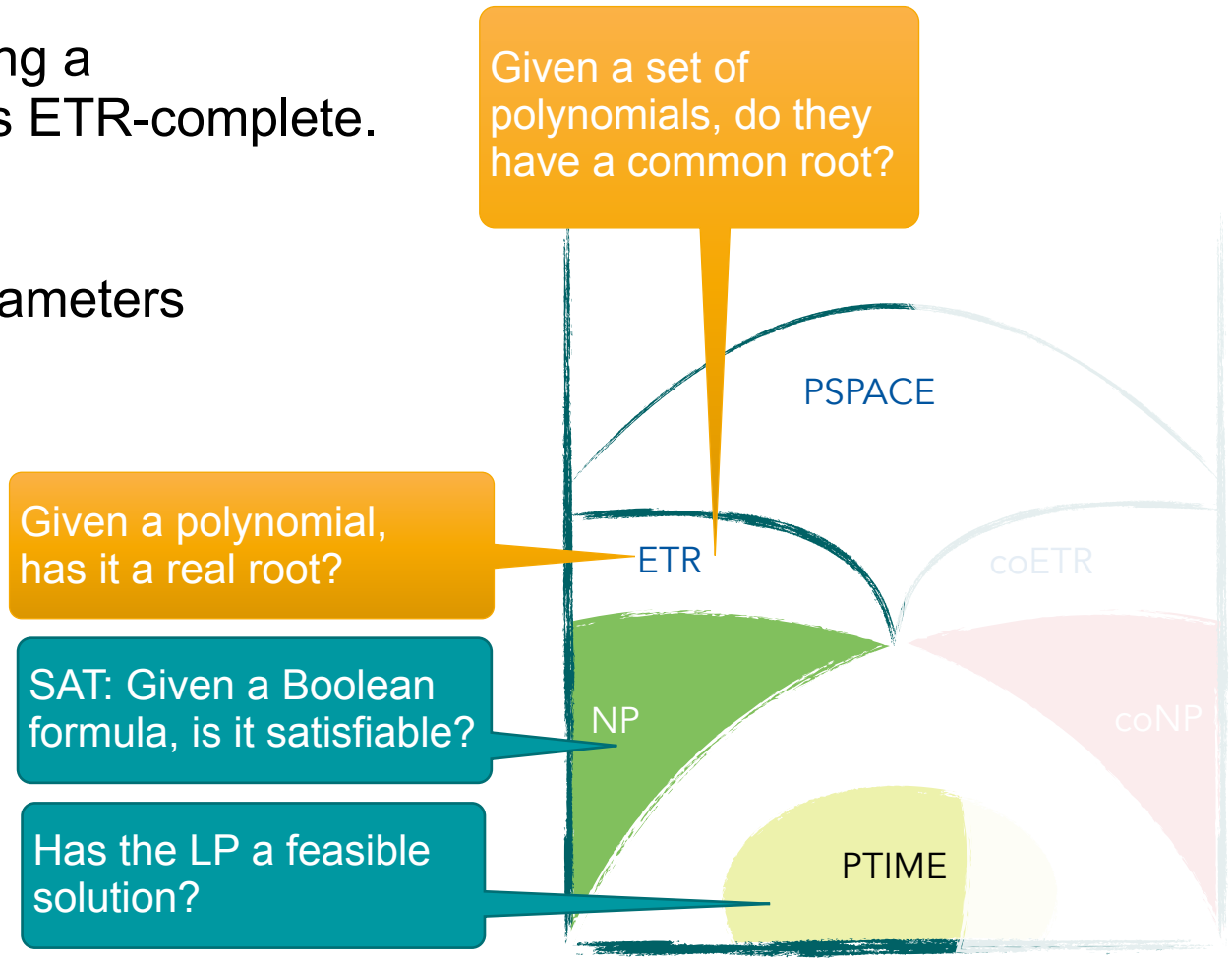
4. Find the cheapest realisation satisfying ***f***.
5. Find all within-budget realisations satisfying ***f***.

## Feasibility is ETR-Complete

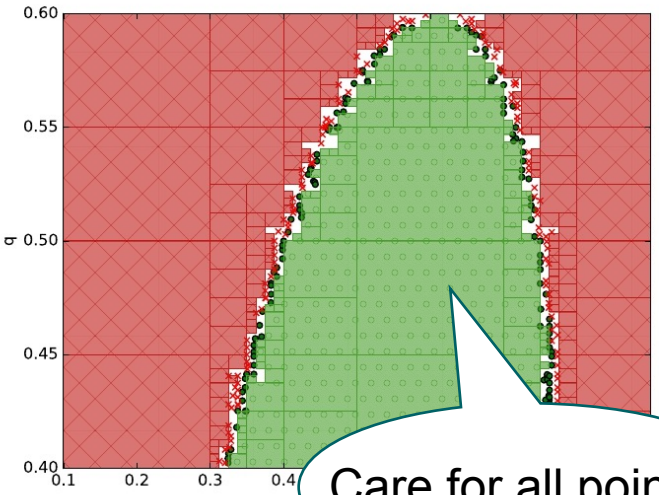
[Hutschenreiter et al, 2017, JCSS'21]

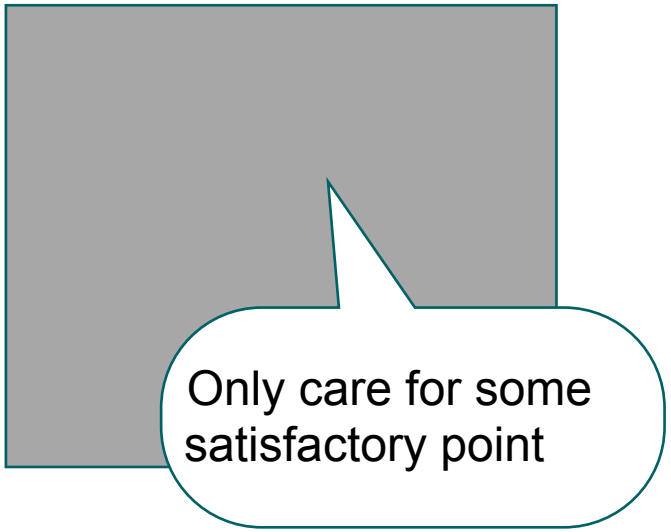
The problem of finding a feasible realisation is ETR-complete.

It is only exponential in the number of parameters



# Practical Parameter Synthesis




Several variants of encoding via SMT solvers [CAV15]

Parameter lifting: abstraction-refinement [ATVA16]
surveyed in [Arxiv19]

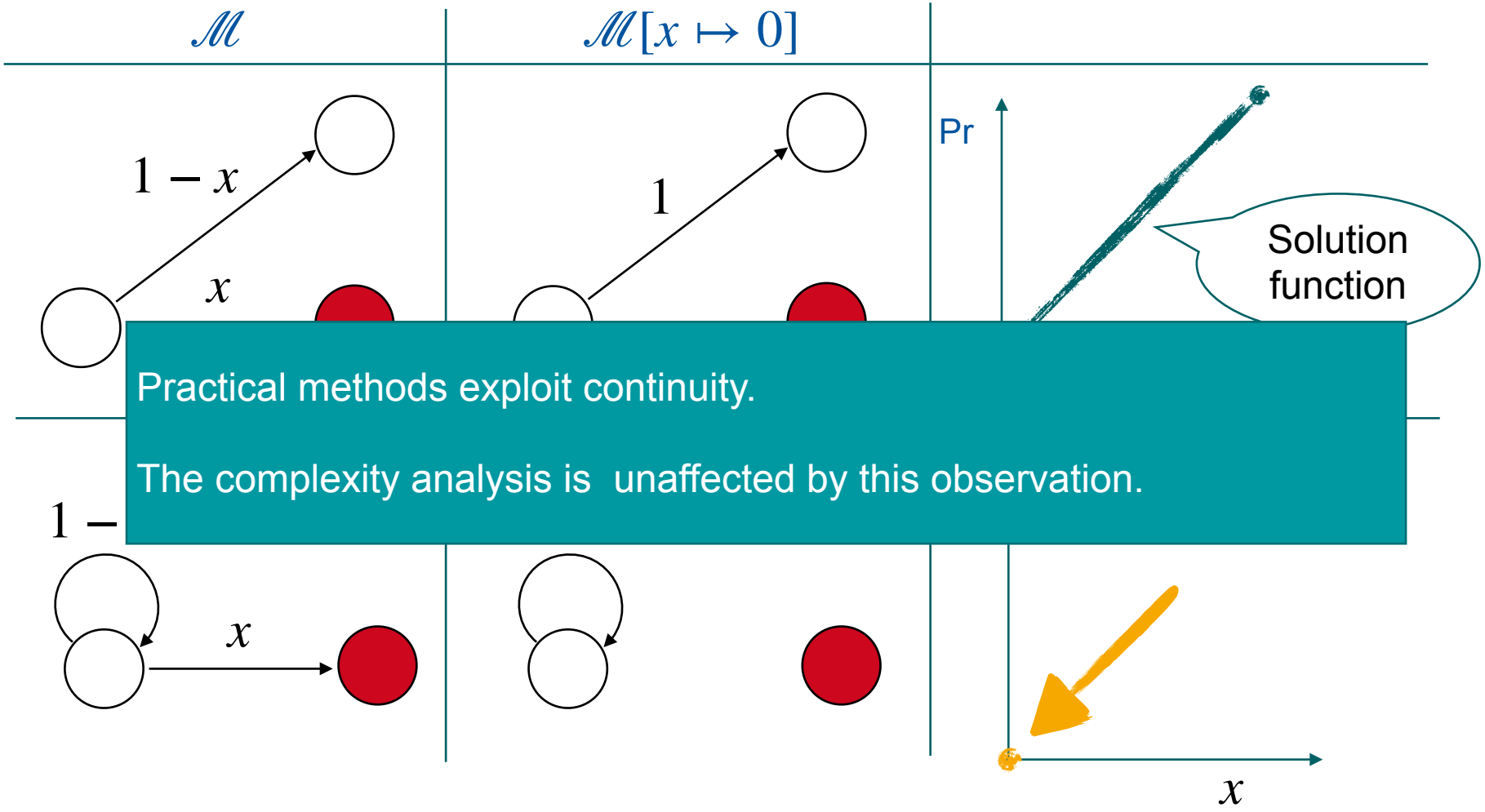
Abstraction-refinement w local monotonicity [TACAS21]

Sampling based methods such as particle swarm or hill-climbing [Chen et al.'14, VMCAI'22]

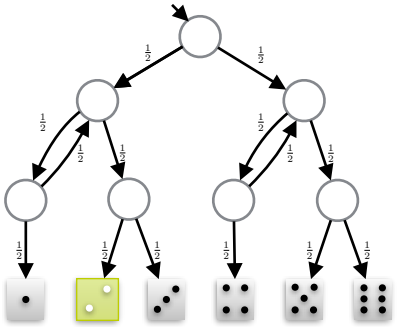
Iterative convex optimisation schemes [TACAS'17]
[ATVA'18, TAC'22]

# Graph preservation

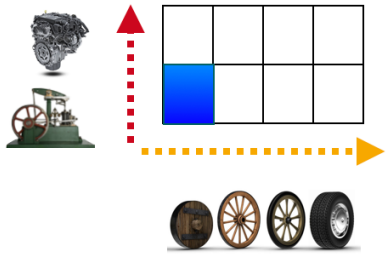
$x \mapsto 0$ is **not** graph preserving



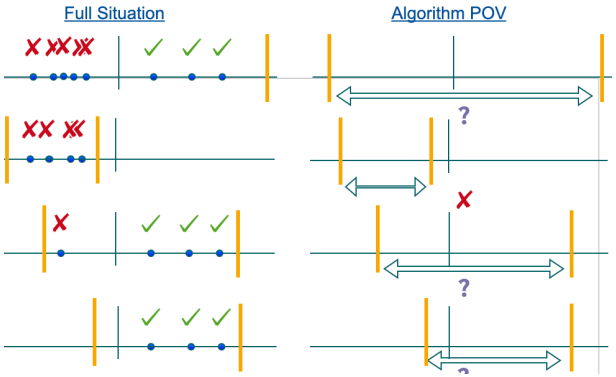
# Overview



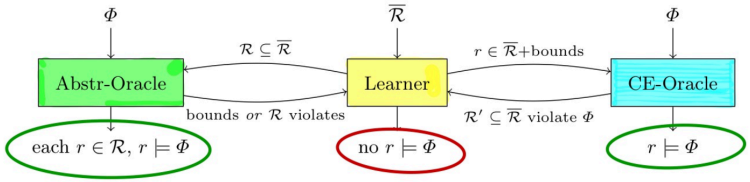
1. Probabilistic Systems & Parameter Synthesis Recap



2. Discrete setting w topology changes



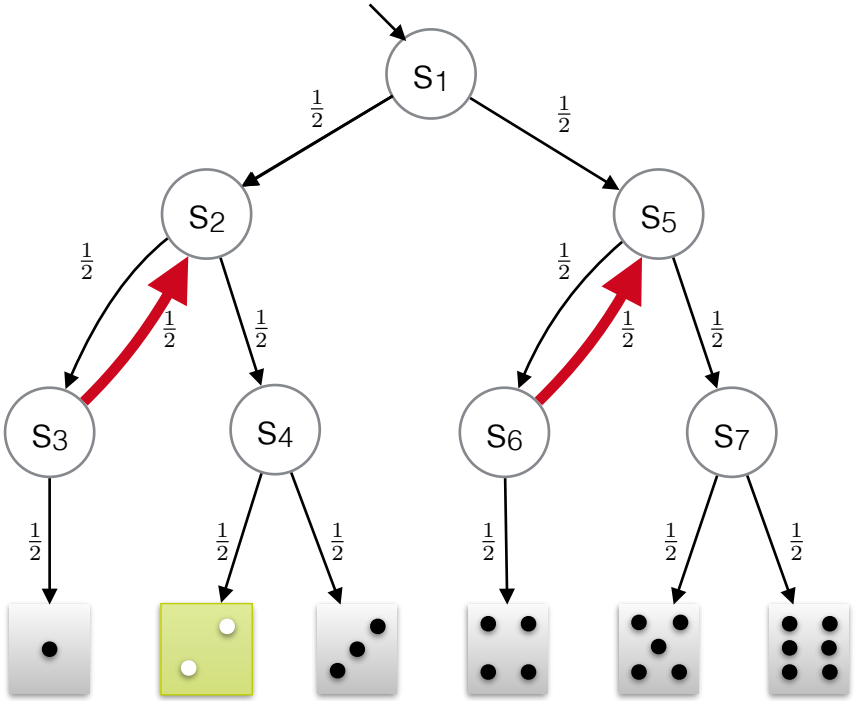

3. Various Algorithms

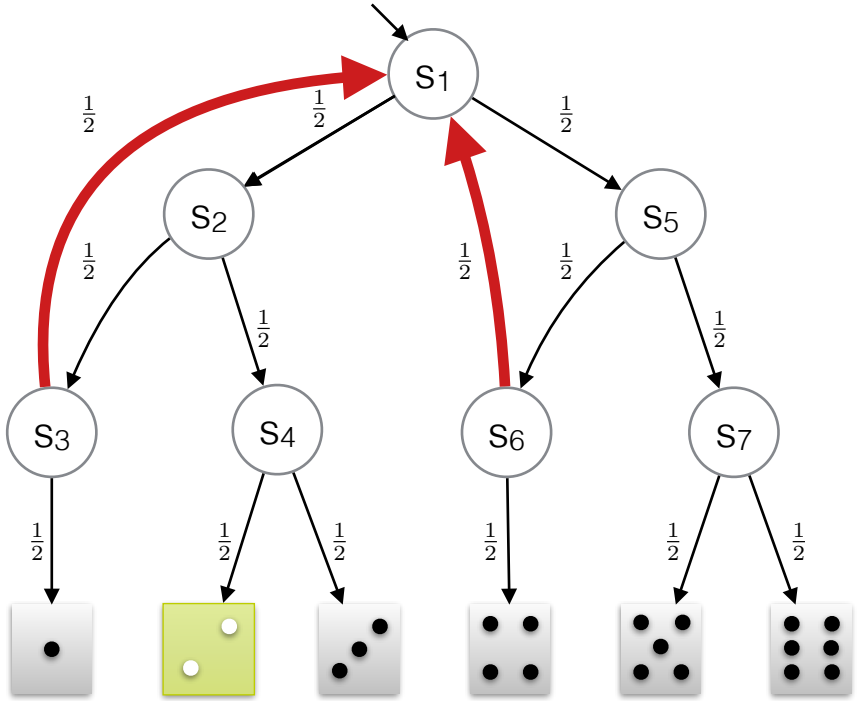



4. PAYNT and Examples

# Focus on topology changes

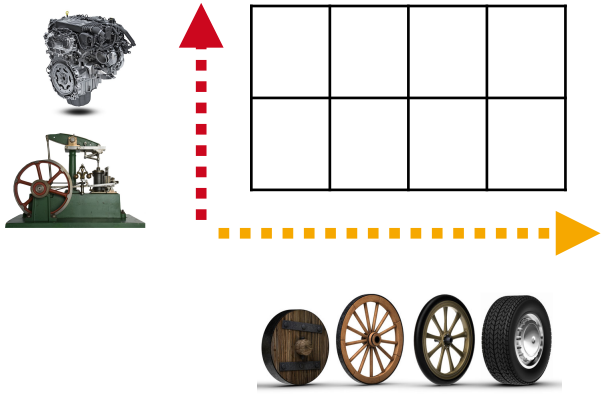
## Two variants of Knuth-Yao



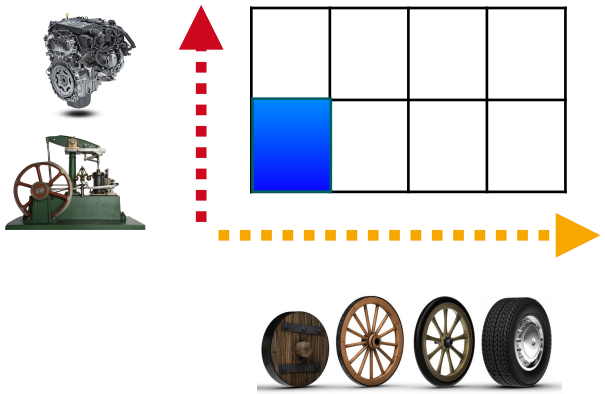
# A discrete setting

## Product line



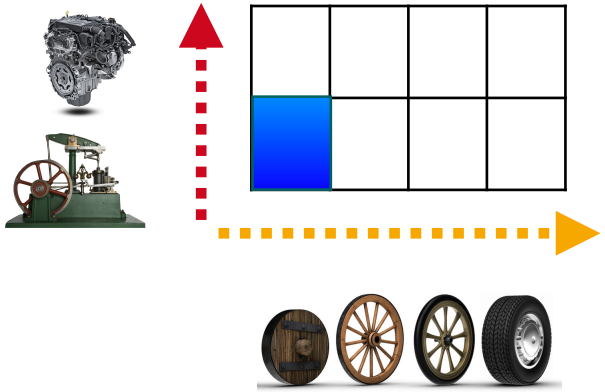
# A discrete setting

## Product line

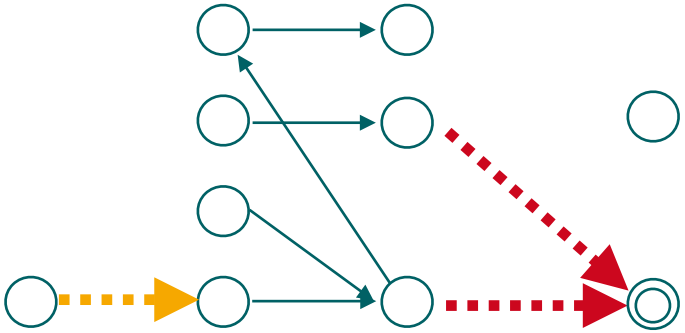


# A discrete setting

## Product line

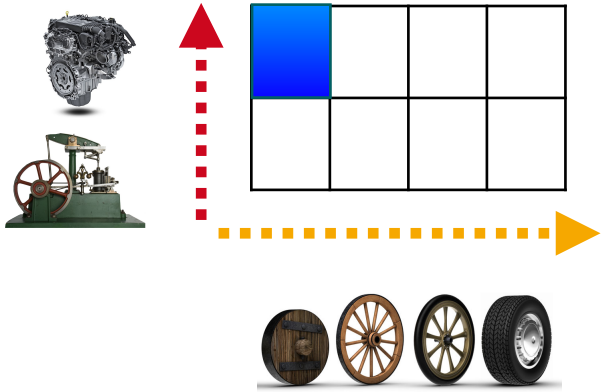


**Markov chain to evaluate reliability:**

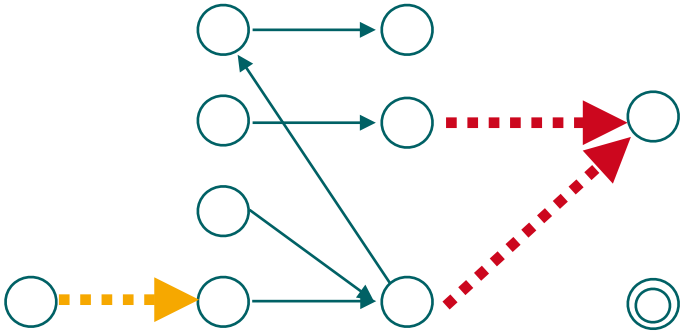


# A discrete setting

## Product line

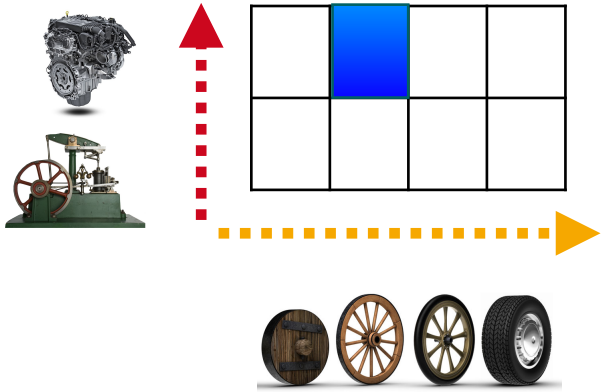


**Markov chain to evaluate reliability:**

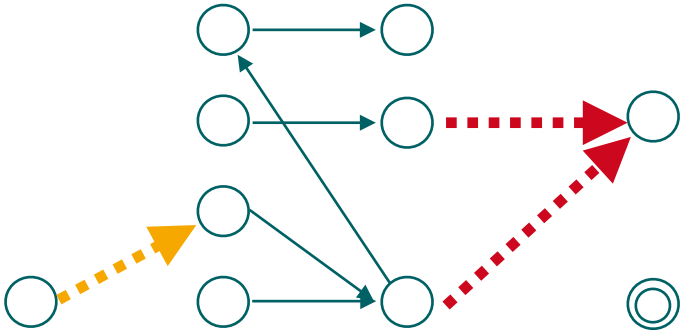


# A discrete setting

## Product line

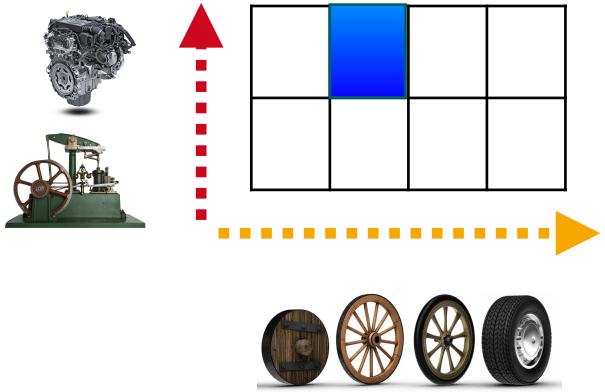


**Markov chain to evaluate reliability:**

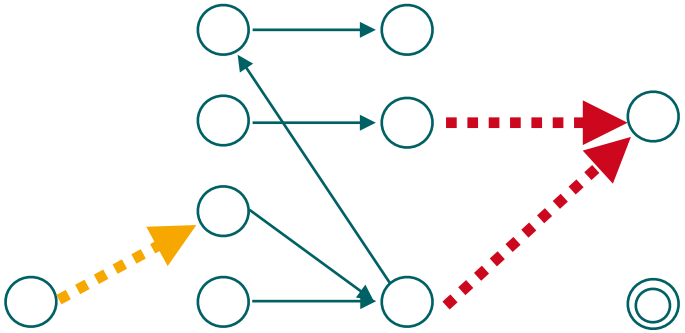


# A discrete setting

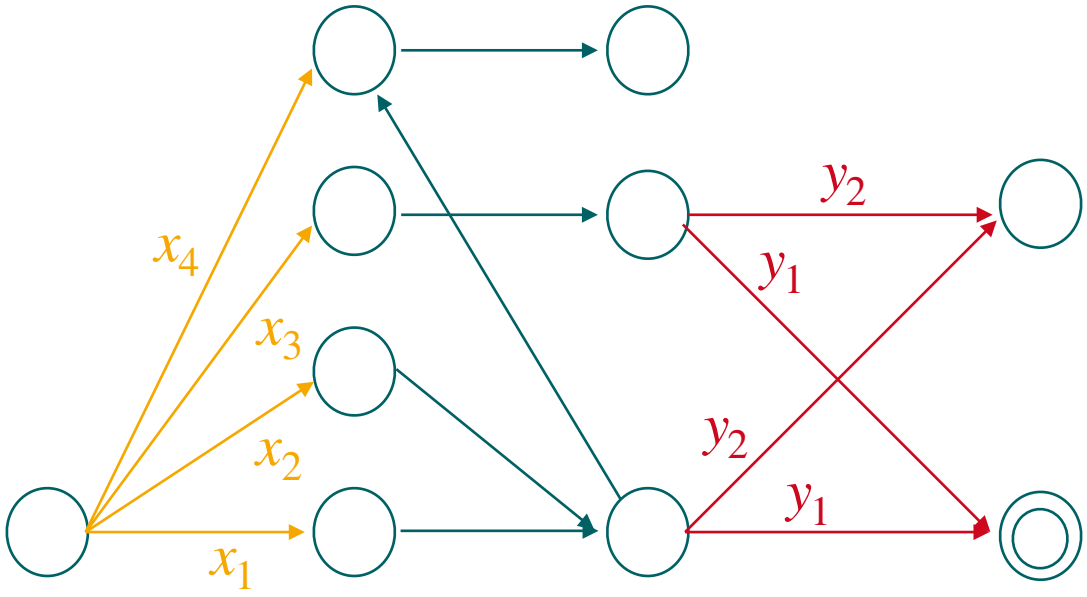
## Product line



**Markov chain to evaluate reliability:**



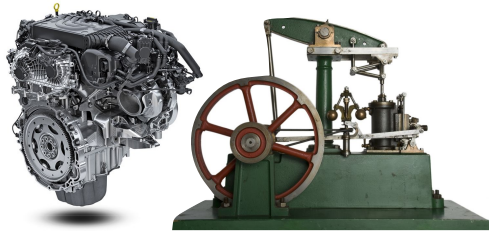
# Modelling families with pMCs




$x_1$ $x_2$ $x_3$ $x_4$

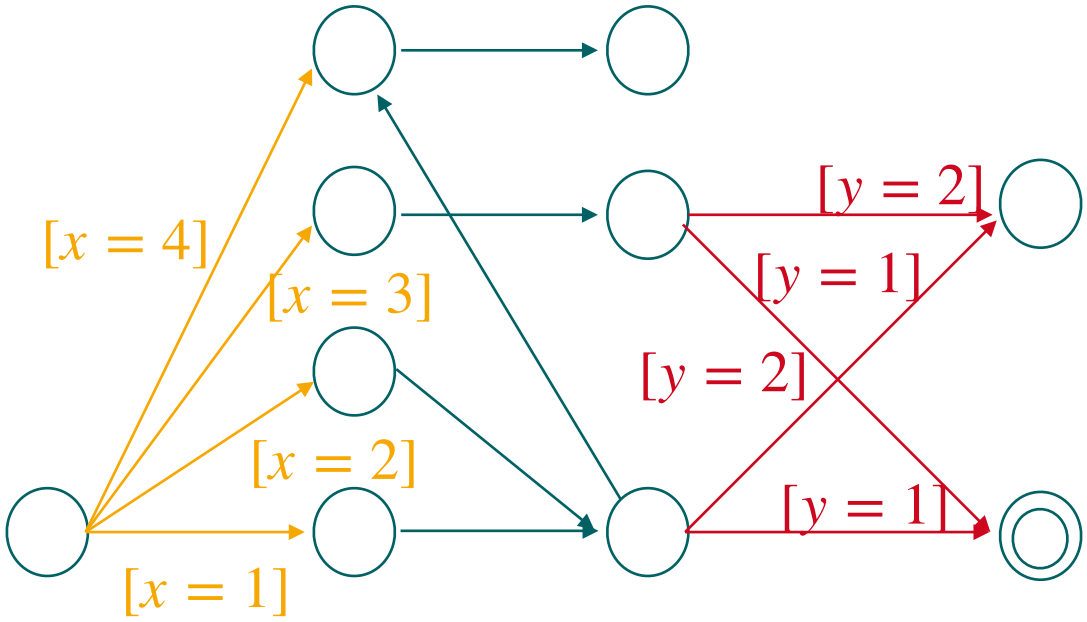
$$x_i \in \{0,1\}, \sum x_i = 1$$

$$y_i \in \{0,1\}, \sum y_i = 1$$



$y_1$ $y_2$

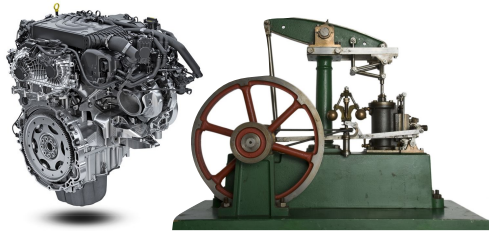
# Reparameterization to “family MCs”




$x = 1$ $x = 2$ $x = 3$ $x = 4$

$x \in \{1,2,3,4\}$

$y \in \{1,2\}$



$y = 1$ $y = 2$

# Family Generators

✓ **Parametric Markov chains** [Daws, Lanotte *et al*, Hahn *et al*, Bartocci *et al*, Brim *et al*,]

- ➡ Parameter-labeled transitions —> uncountably many MCs
- ➡ Topology preserving

✓ **ProFeat** [Chrszon *et al*, Form Asp Comp 2018]

- ➡ Extensions to PRISM modelling language
- ➡ Analysis support

✓ **Modal transition systems** [Larsen, Thomsen, LICS 1988]

- ➡ *May* and *must* transitions
- ➡ Synthesis with dependencies (qualitative) [Benes *et al*, ATVA 2011, LPAR 2012]
- ➡ Probabilistic extensions [Delahaye *et al*, Inf Comp 2013]

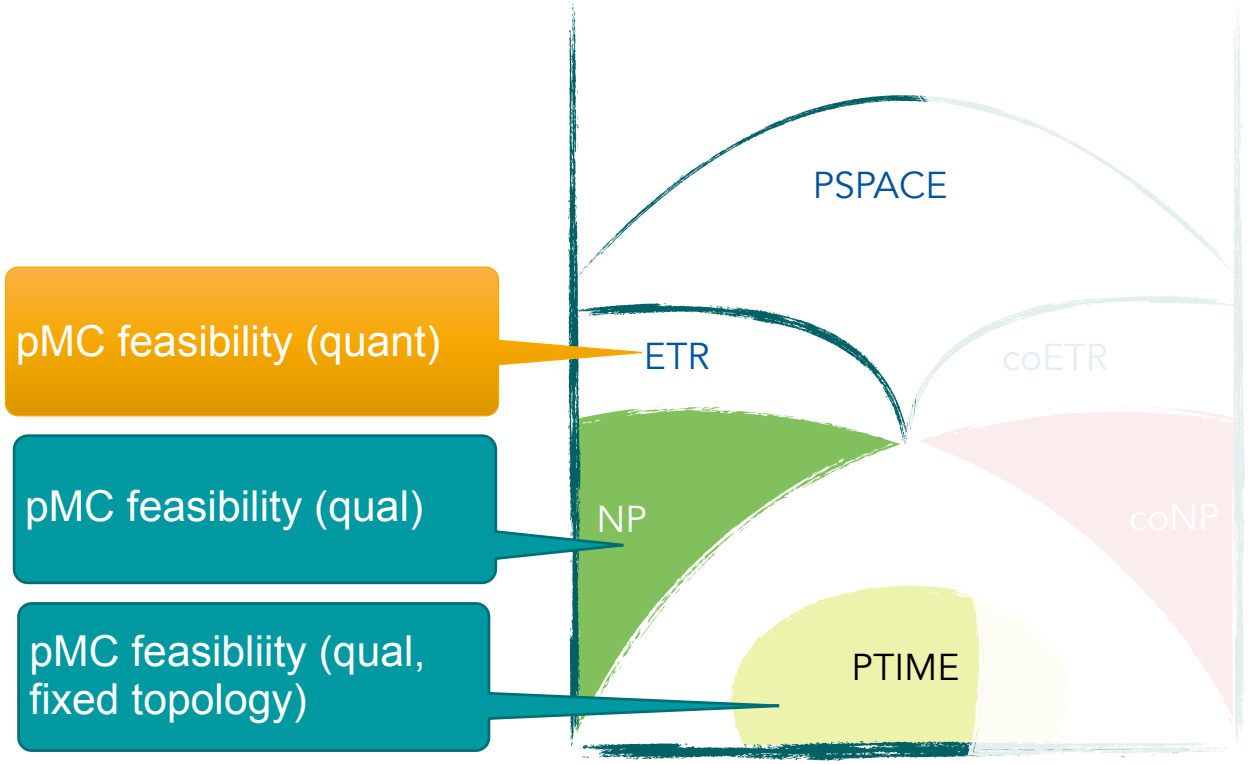
✓ **QFLan** [Vandin *et al*, SPLC 2015, FM 2018]

- ➡ Focus on product lines

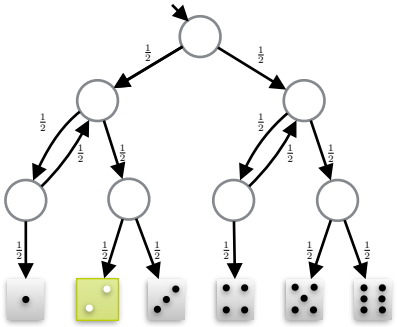
# Complexity

[FAoC'21, JCSS'21]

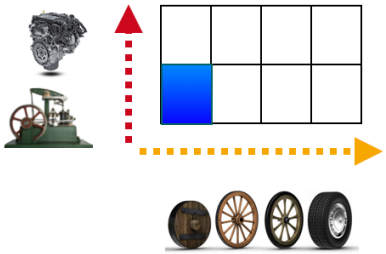
- Feasibility is now NP-complete
- even for "Probability to reach G = 1?"  (Remember: topology changes)



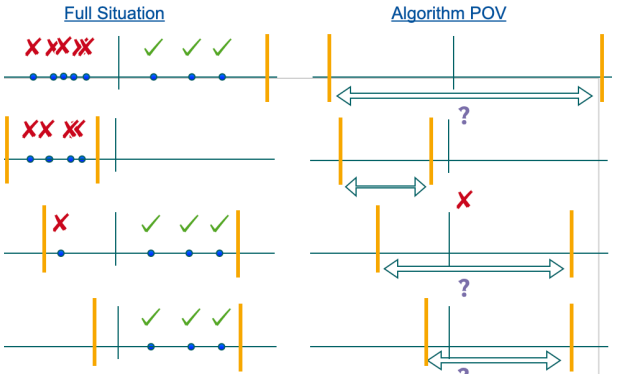
# Overview



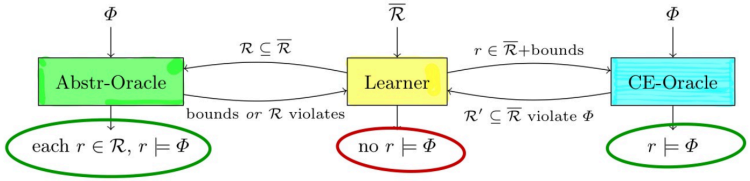
1. Probabilistic Systems & Parameter Synthesis Recap



2. Discrete setting w topology changes



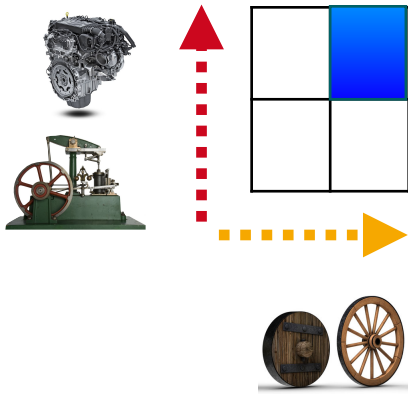

3. Various Algorithms

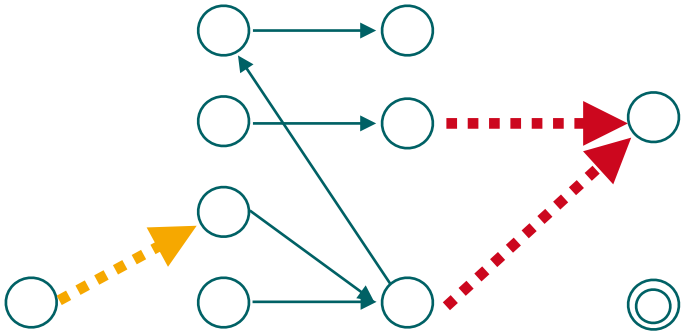



4. PAYNT and Examples

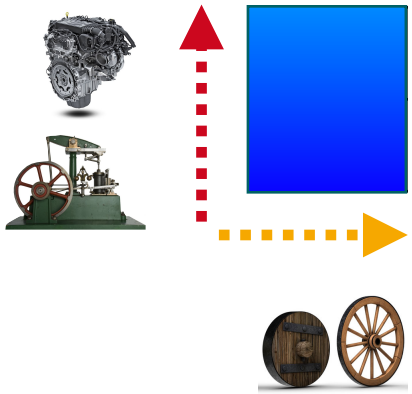
# A discrete setting

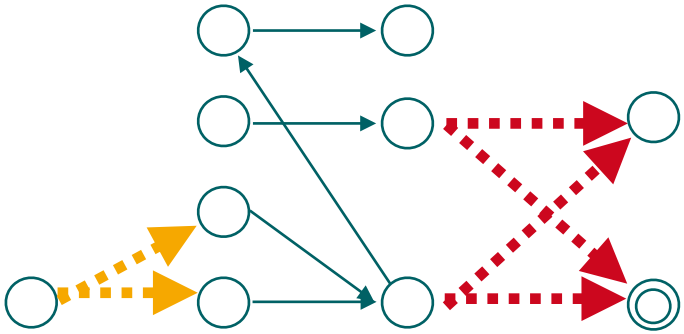
## Product line



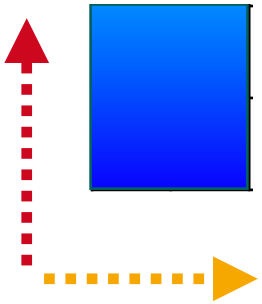


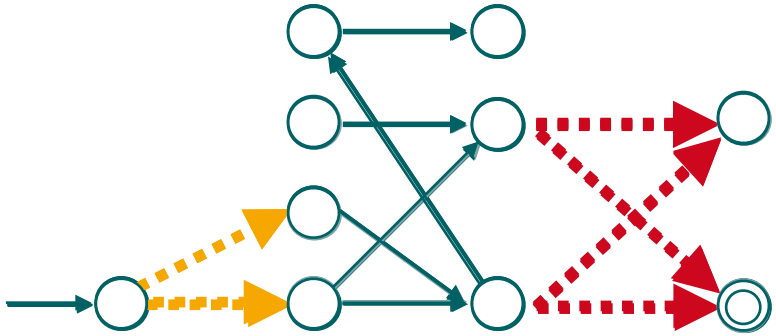
# Families





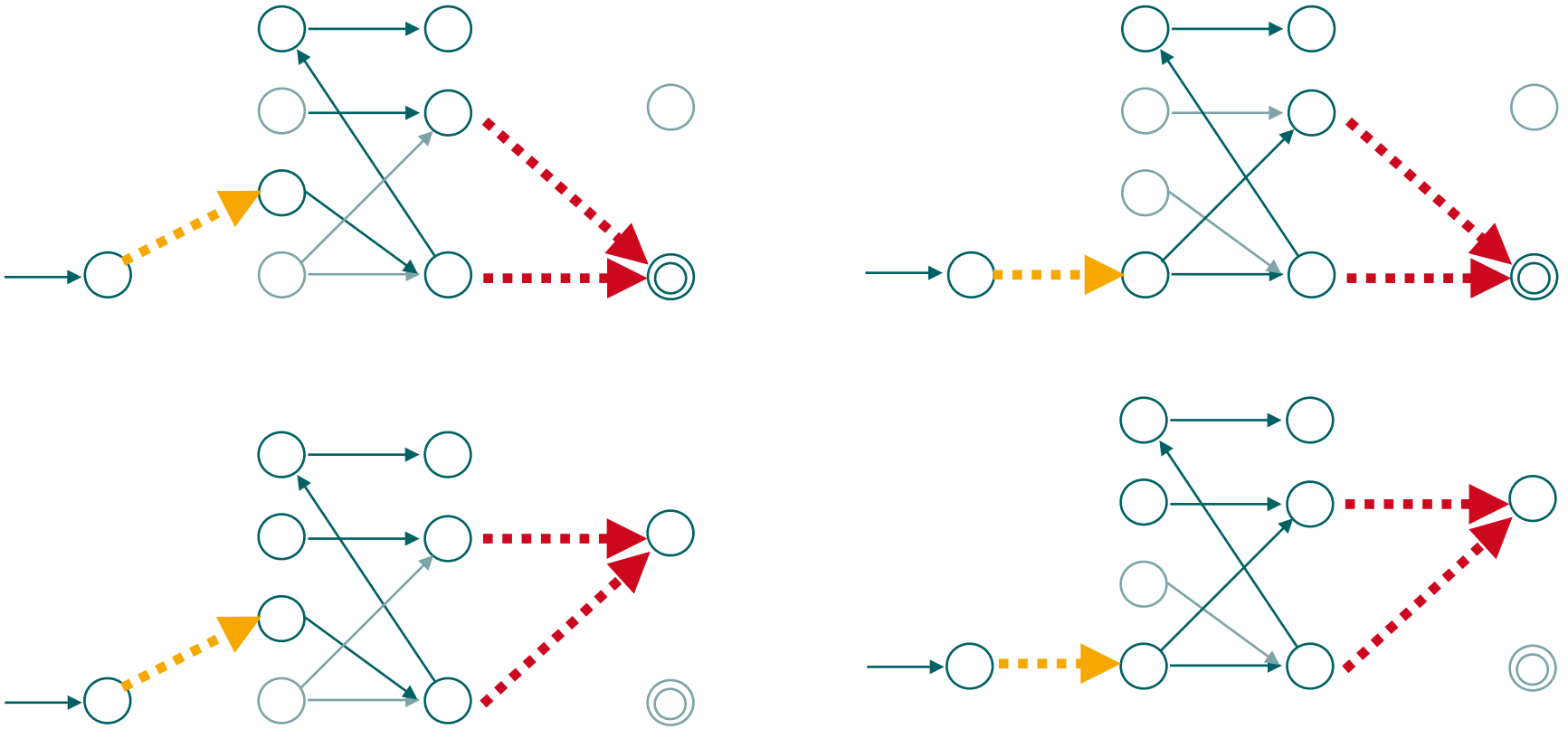
# Families





# All-in-one MDP (disjoint union of all MCs)

[Chrszon et al, Form Asp Comp 2018]

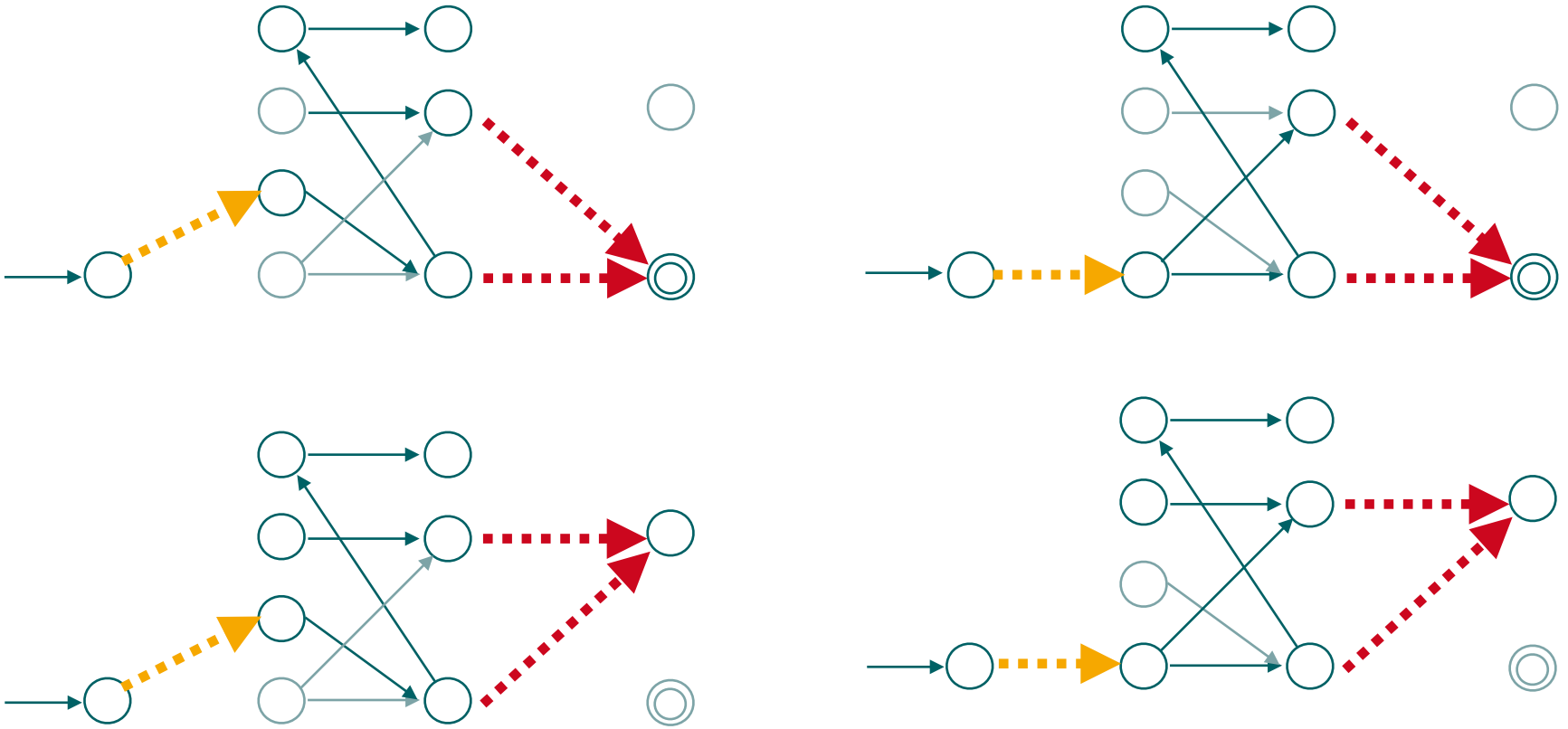


# Algorithms

**Abstraction-Refinement**

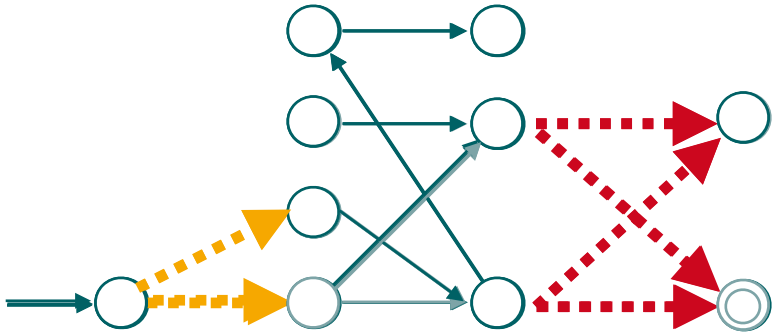
# All-in-one MDP (disjoint union of all MCs)

[Chrszon et al, Form Asp Comp 2018]



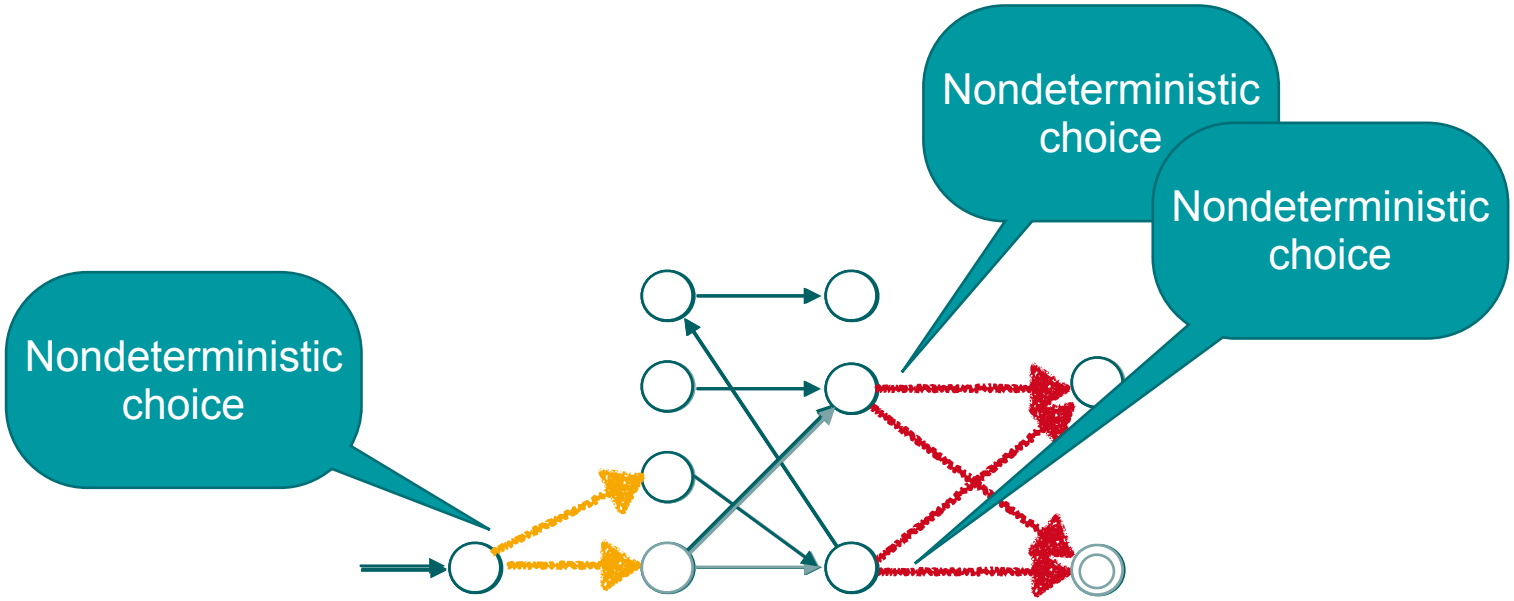
# “forget” in which instantiation we are

[TACAS’19]



# “forget” in which instantiation we are

[TACAS’19]

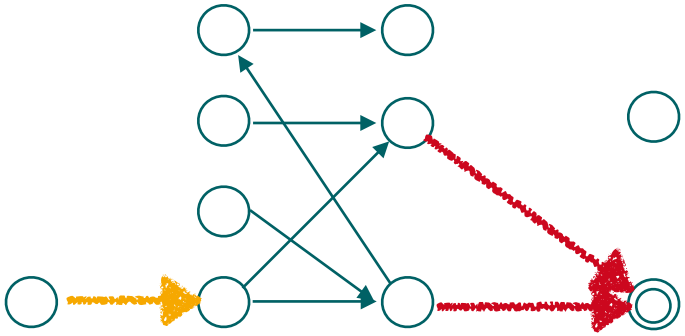



We obtain a quotient MDP

# Consistent strategies

[TACAS'19]

Induced MC of quotient MDP, and an instantiation of the family:

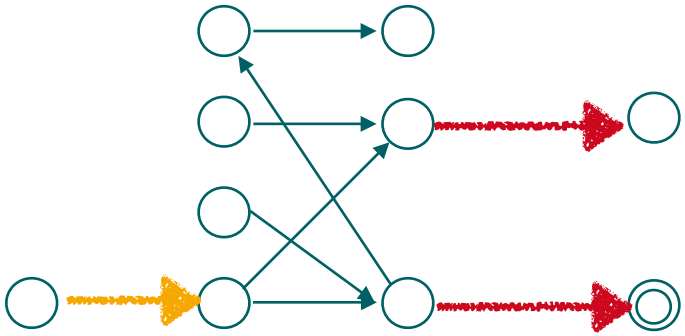


All instantiations are induced by some deterministic memoryless strategy….

# Inconsistent strategies

[TACAS'19]

Induced MC of quotient MDP, not an instantiation:

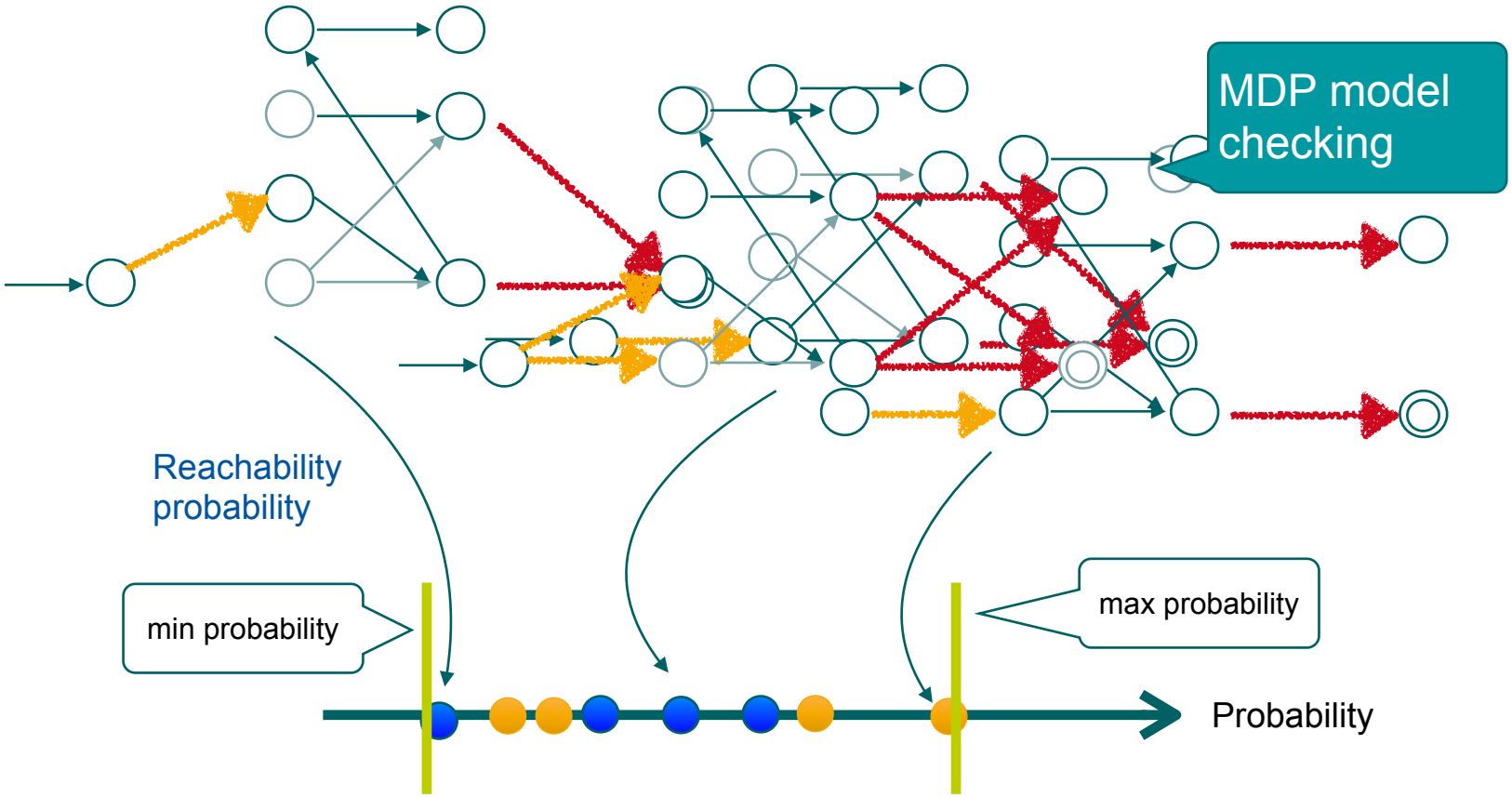


Some strategies are *inconsistent* wrt the parameters,
(other strategies are *consistent*)

# Analysing the quotient

[TACAS'19]

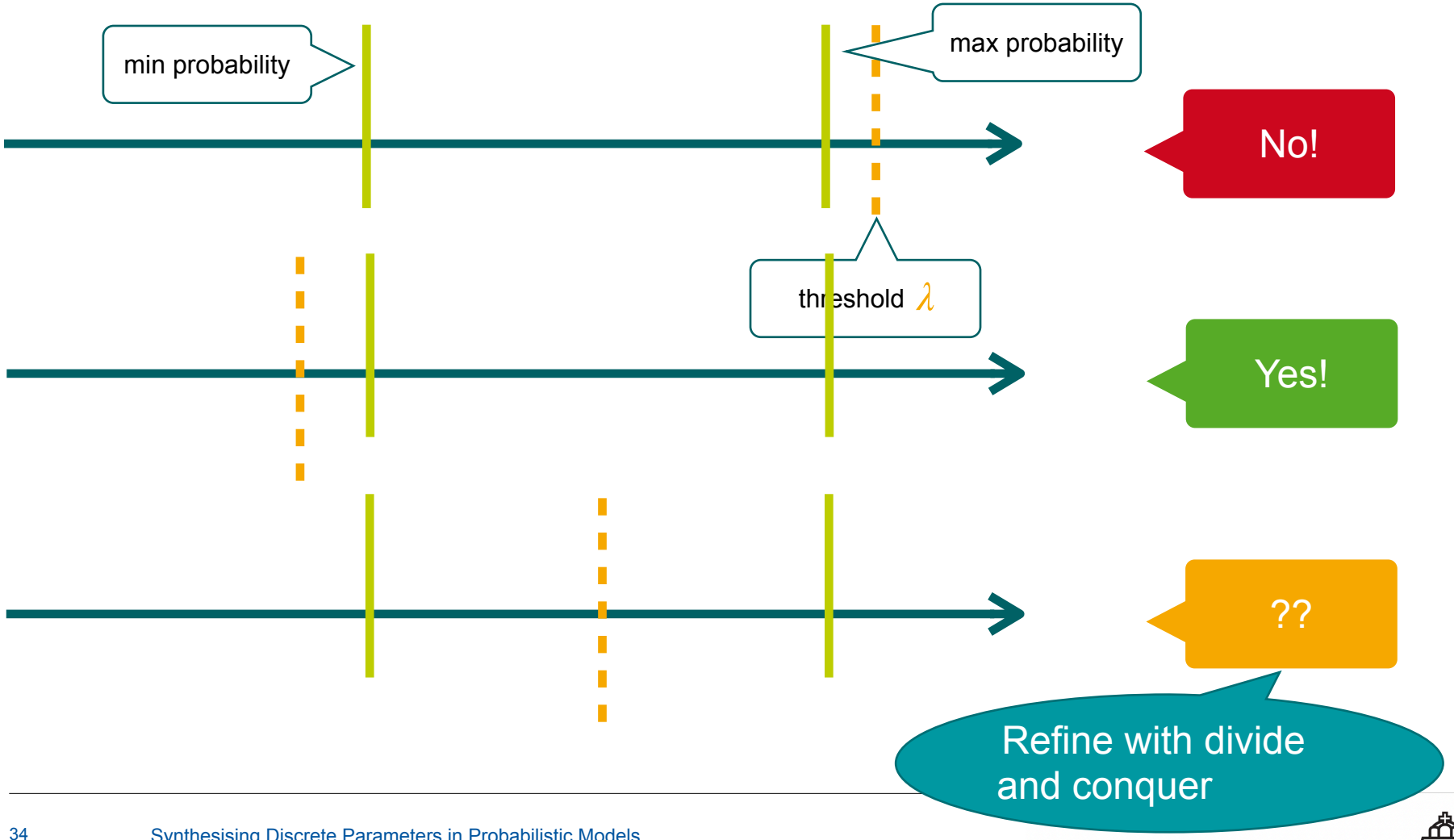
**Consistent strategies are subset of all strategies.**



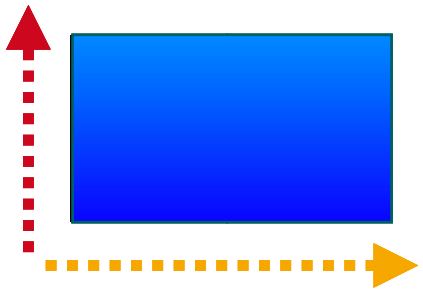
# Analysing the quotient

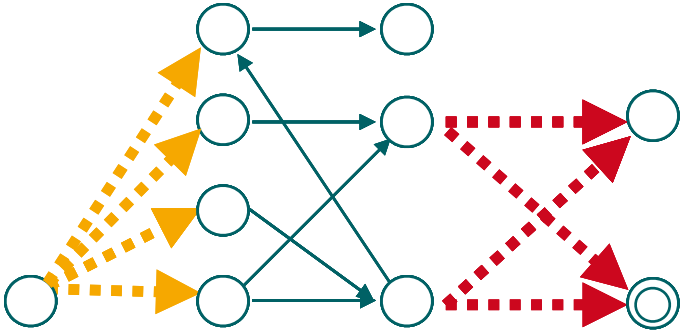
[TACAS'19]

**Does an instantiation exist s.t. induced probability is more than $\lambda$ ?**

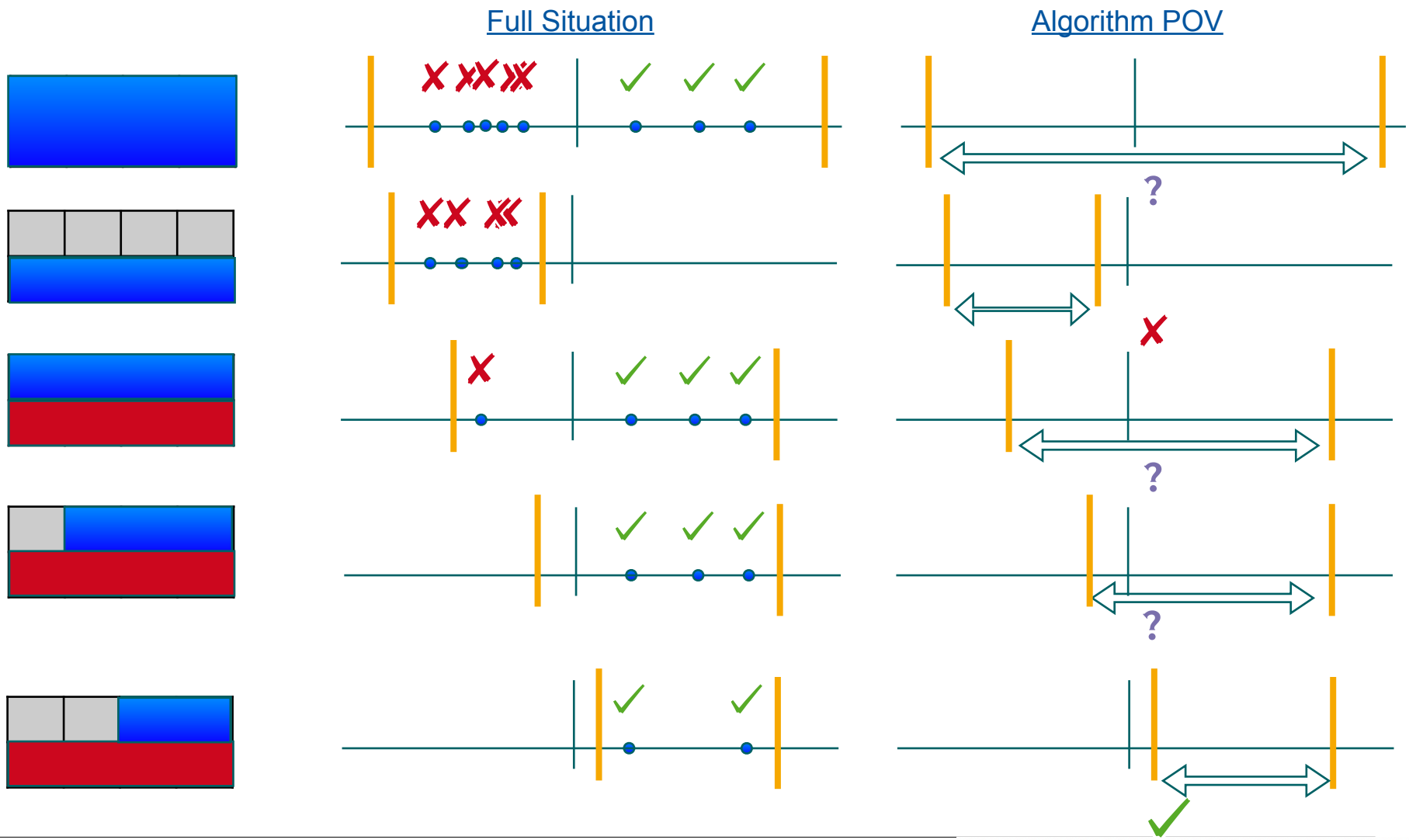


# Divide and Conquer Refinement





# Abstraction-Refinement (by splitting)

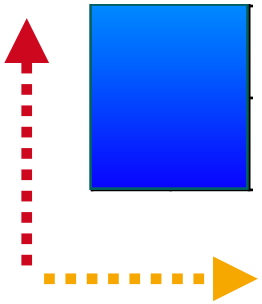


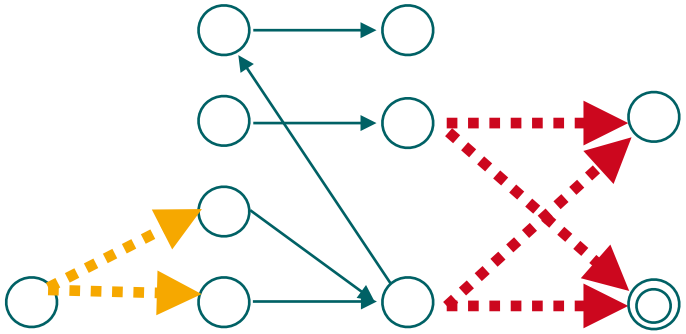
# Algorithms

**Counterexample-Guided Inductive Synthesis**

# Families

[FM'19]

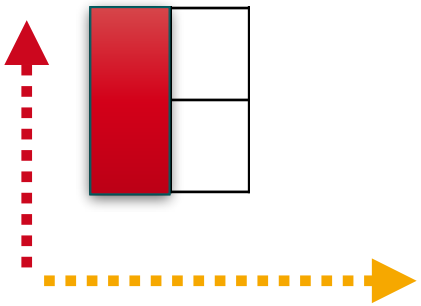




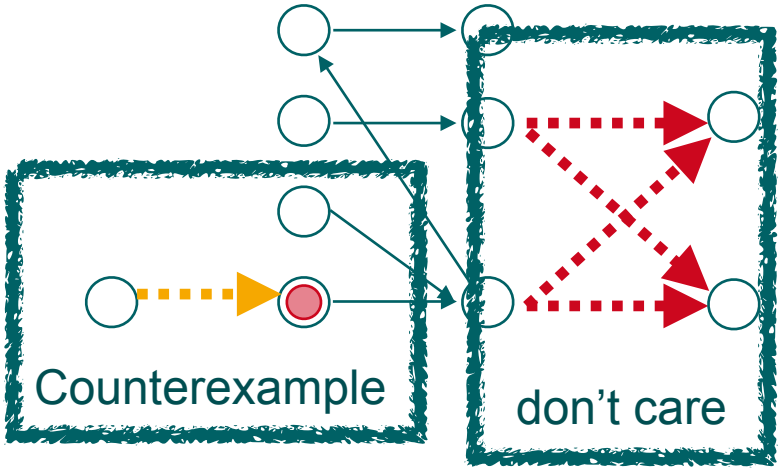
# Counterexample Reasoning Idea

[FM'19]

We reach the red state with probability at most $\lambda$

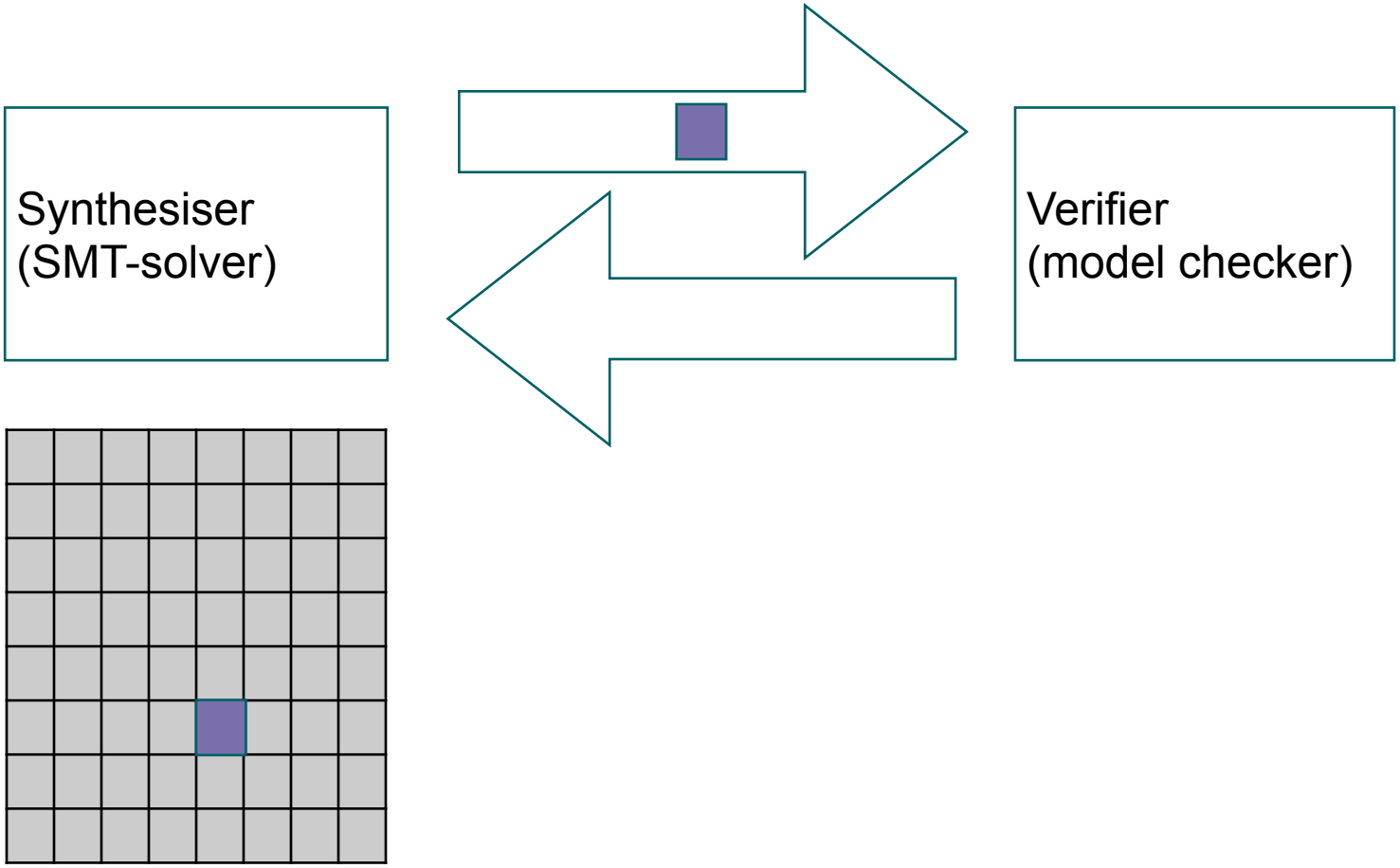


**No!**



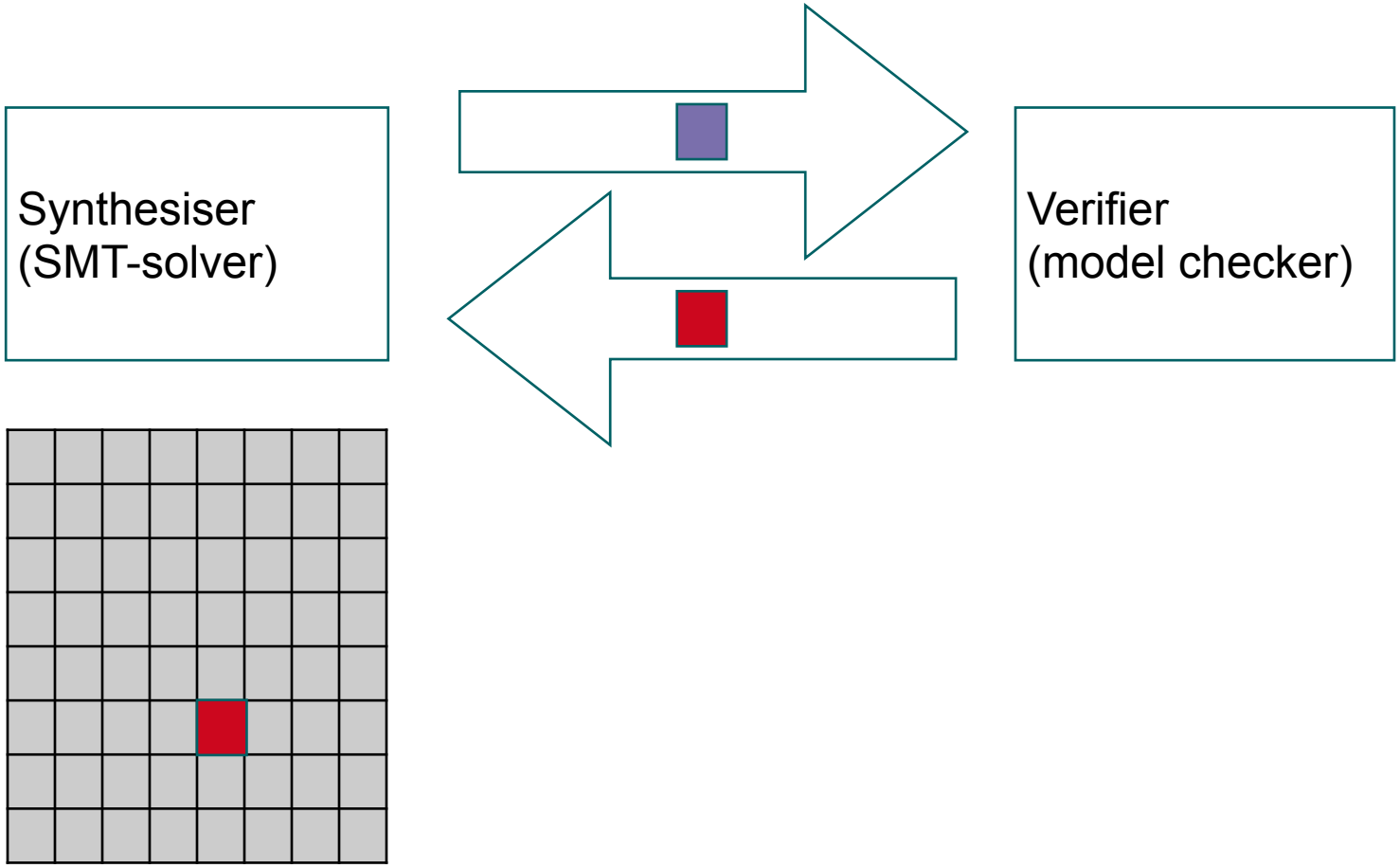
# Counterexample-Guided Inductive Synthesis

[FM'19]



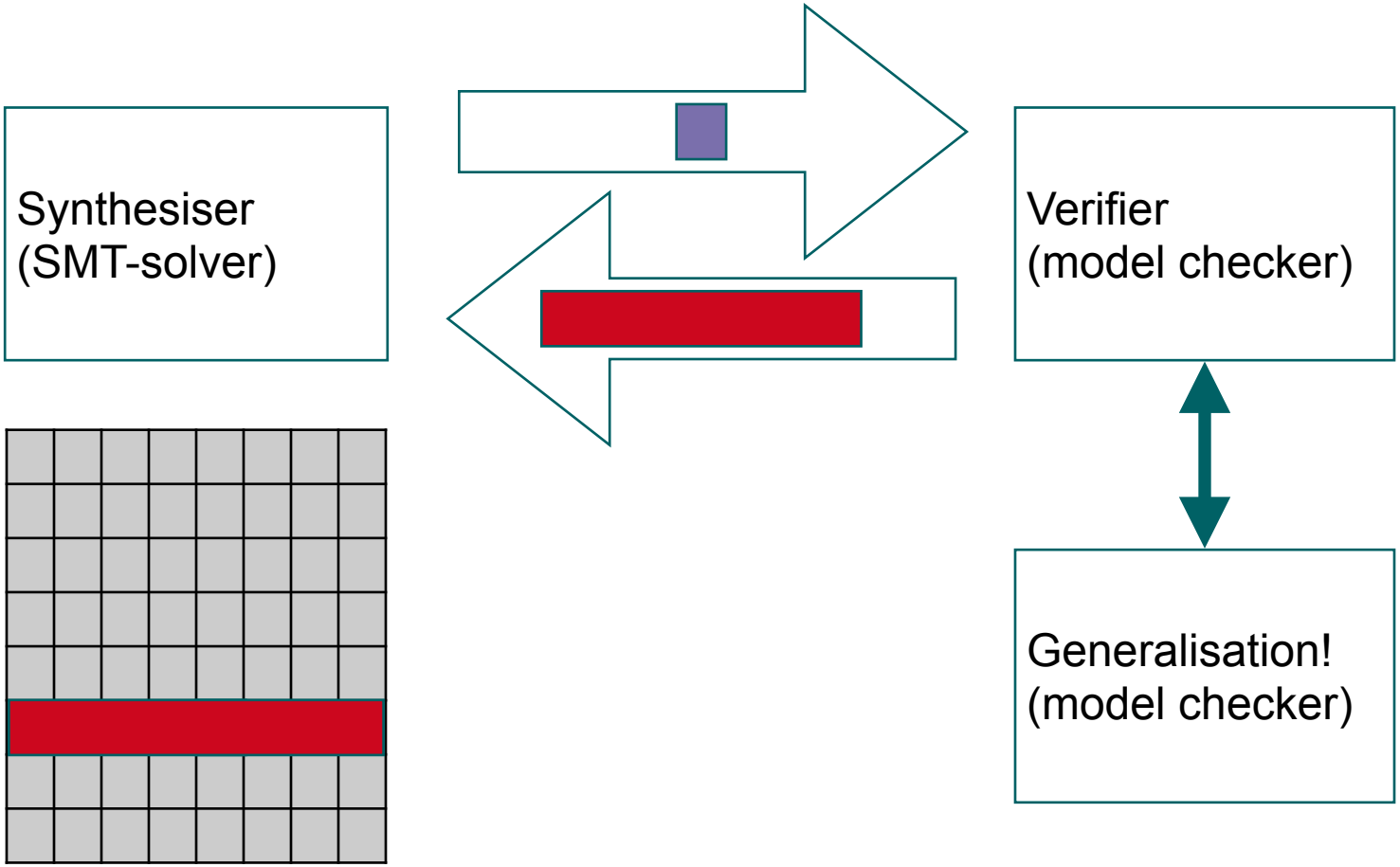
# Counterexample-Guided Inductive Synthesis

[FAOC'21]

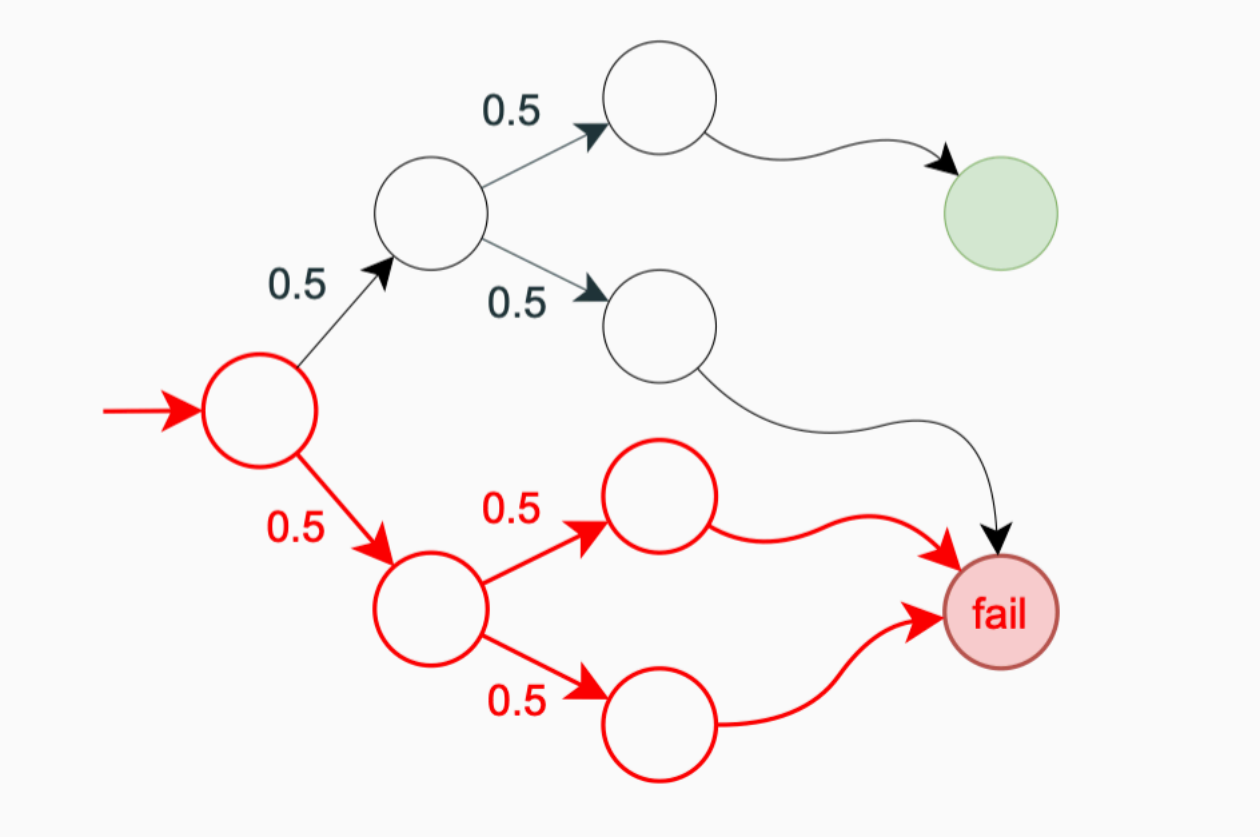


# Counterexample-Guided Inductive Synthesis

[FAOC'21]



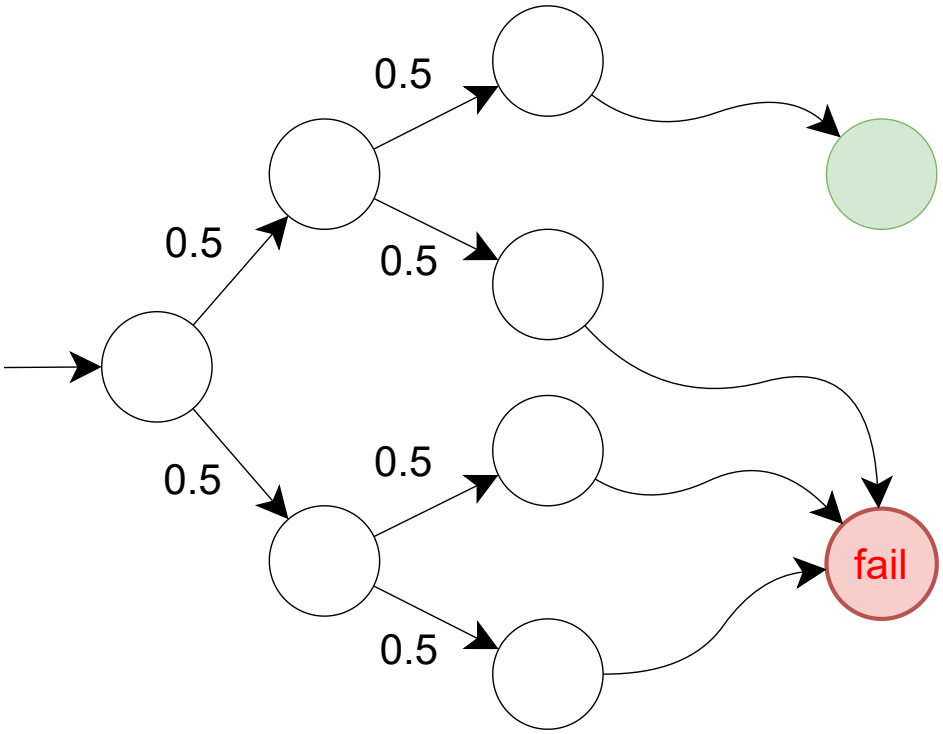
# Counterexamples



# Generating Counterexamples

## Greedy

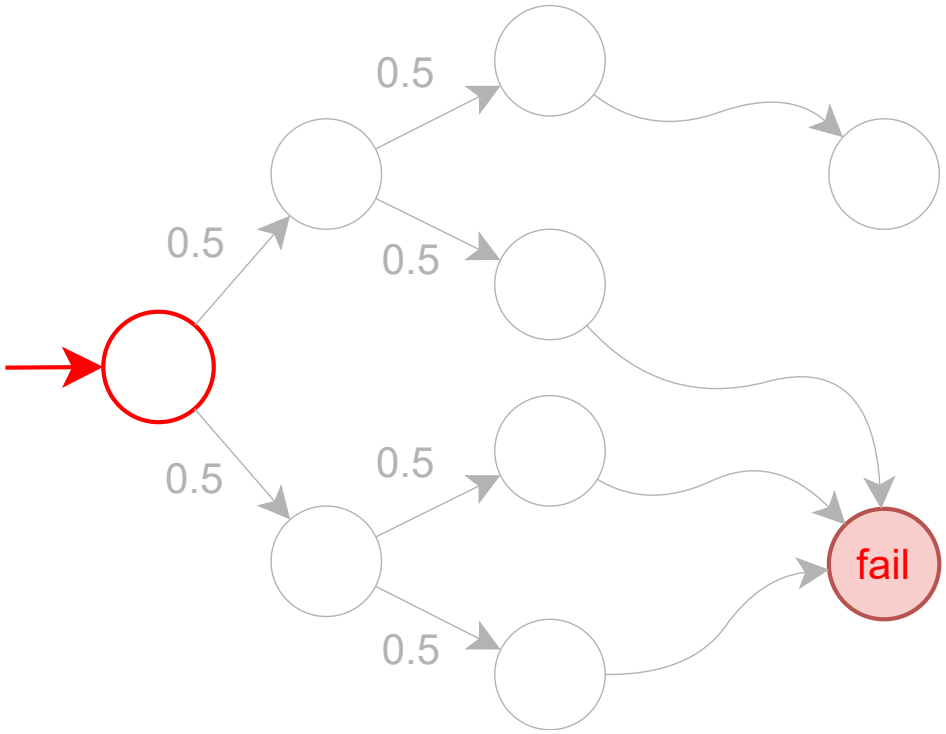
Possible counterexample generation for $\mathbb{P}_{\leq 0.4}(\Diamond \text{fail})$



# Generating Counterexamples

## Greedy

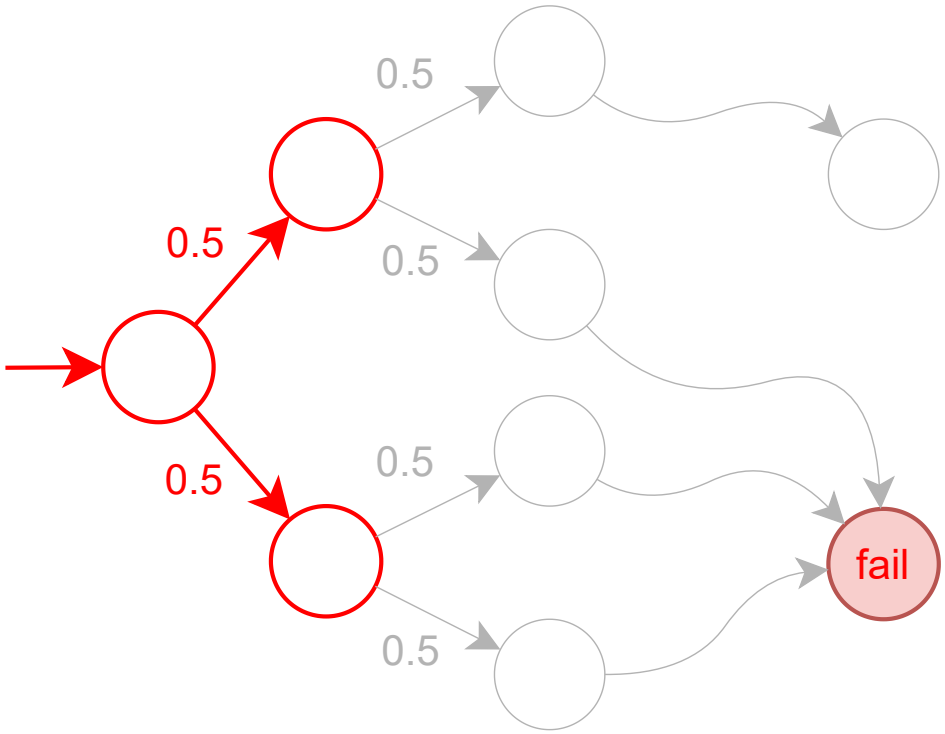
Possible counterexample generation for $\mathbb{P}_{\leq 0.4}(\Diamond \text{fail})$



# Generating Counterexamples

## Greedy

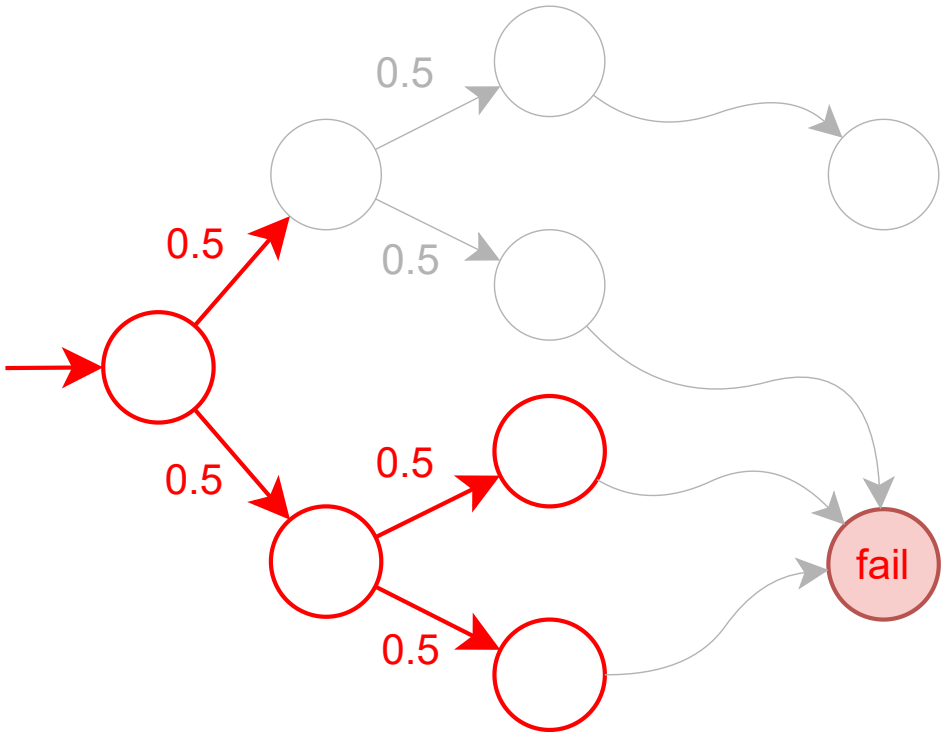
Possible counterexample generation for $\mathbb{P}_{\leq 0.4}(\Diamond \text{fail})$



# Generating Counterexamples

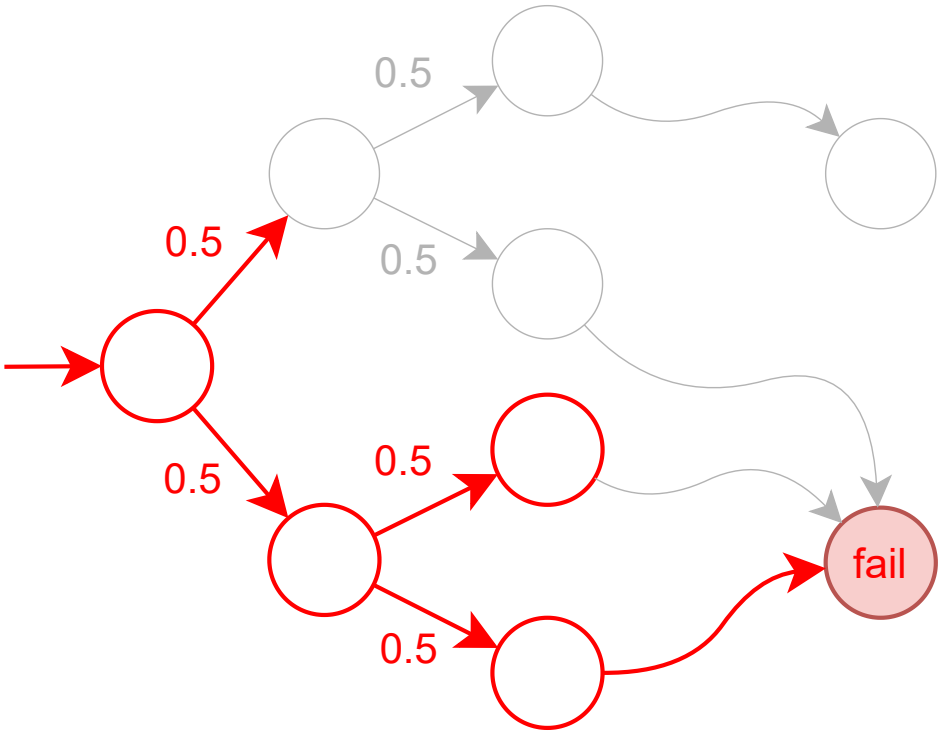
## Greedy

Possible counterexample generation for $\mathbb{P}_{\leq 0.4}(\Diamond \text{fail})$



# Generating Counterexamples

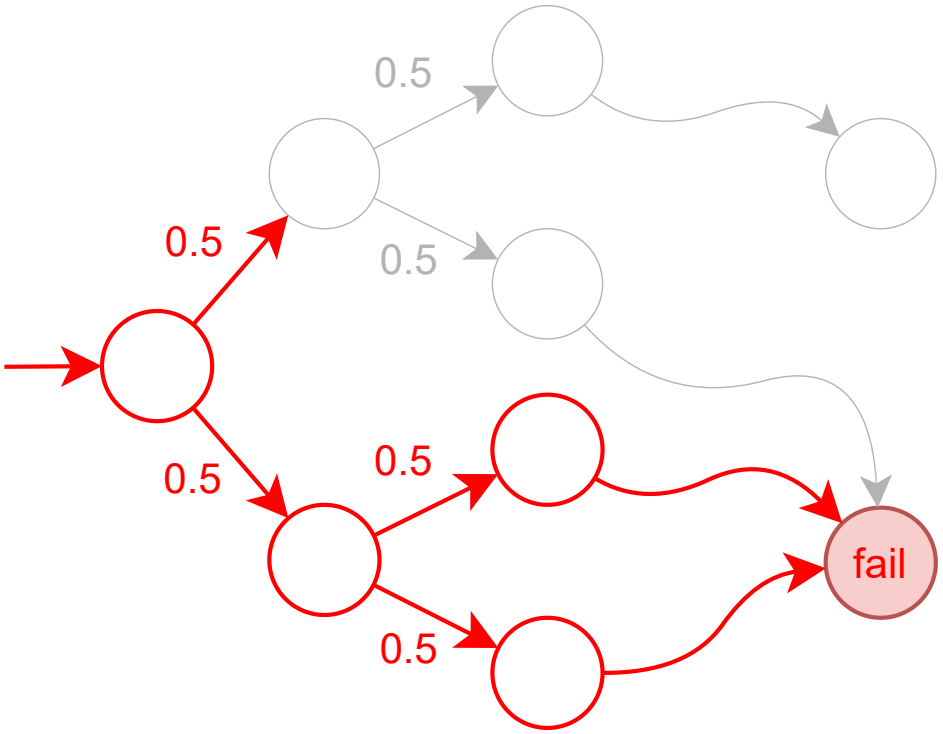
## Greedy



# Generating Counterexamples

## Greedy

Possible counterexample generation for $\mathbb{P}_{\leq 0.4}(\Diamond \text{fail})$



# Counterexamples

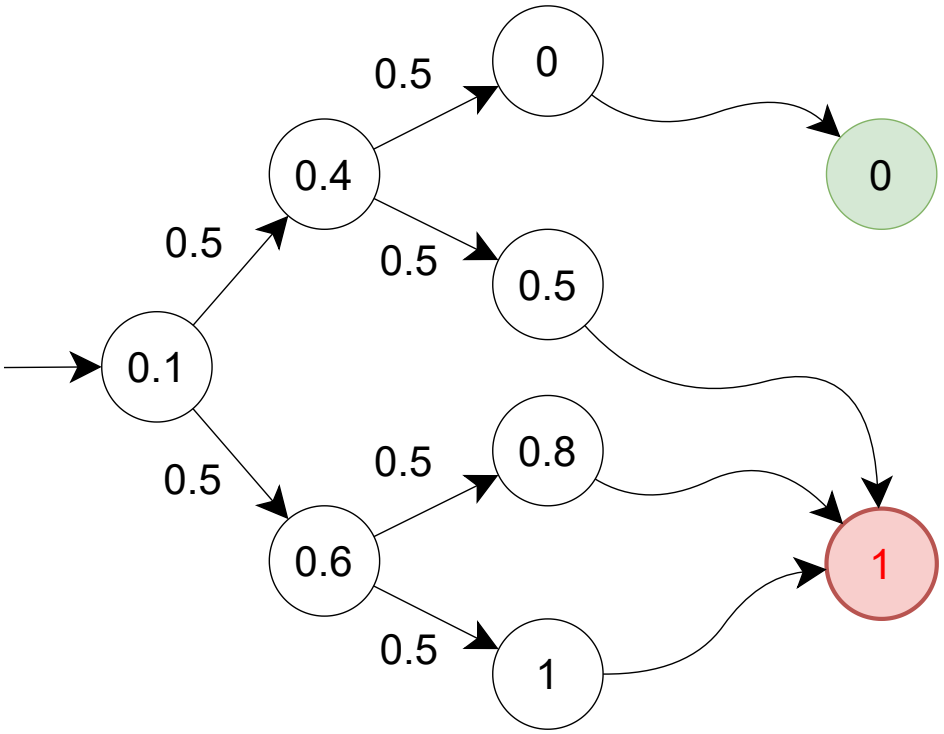
## Overview

- Counterexamples should minimize holes (“Hole-aware counterexamples”)
- **Algorithms**:
  - Greedy (outlined above)
  - MaxSAT approach from [Dehnert et al., 2014]
  - SWITTS [Jantsch et al, 2020]
- **Weakness**: Counterexamples make no assumptions about possible extensions
  They assume that with probability 0 the system goes to fail.
- **Idea**: Use lower bounds on the probability to fail in any family member.
  Bounds can be computed via abstraction-refinement

## Counterexamples modulo family

[TACAS'21]

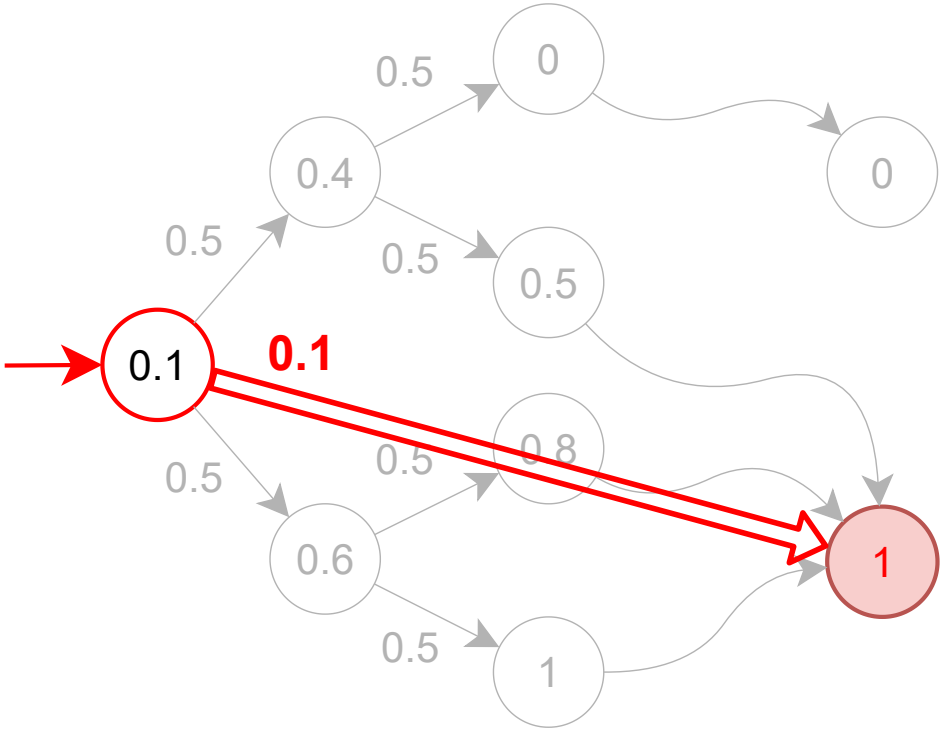
- use val($s$) as a **probabilistic shortcut** from state $s$ to the target
- val($s$) = minimum reachability probability through all possible continuations of $s$ **within the family**



## Counterexamples modulo family

[TACAS'21]

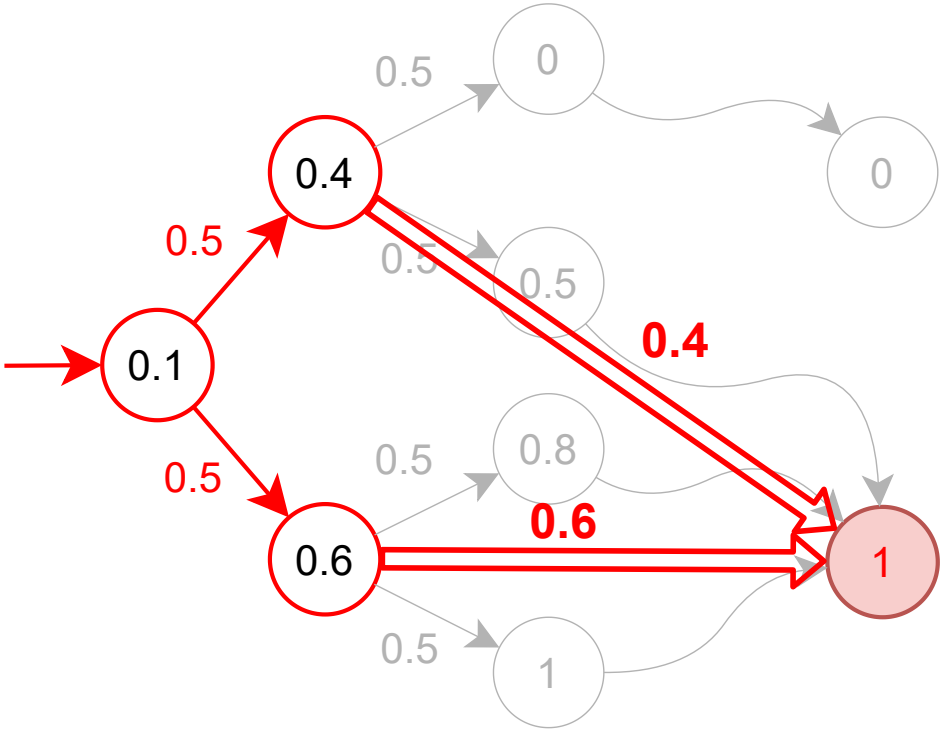
- use val($s$) as a **probabilistic shortcut** from state $s$ to the target
- val($s$) = minimum reachability probability through all possible continuations of $s$ **within the family**



## Counterexamples modulo family

[TACAS'21]

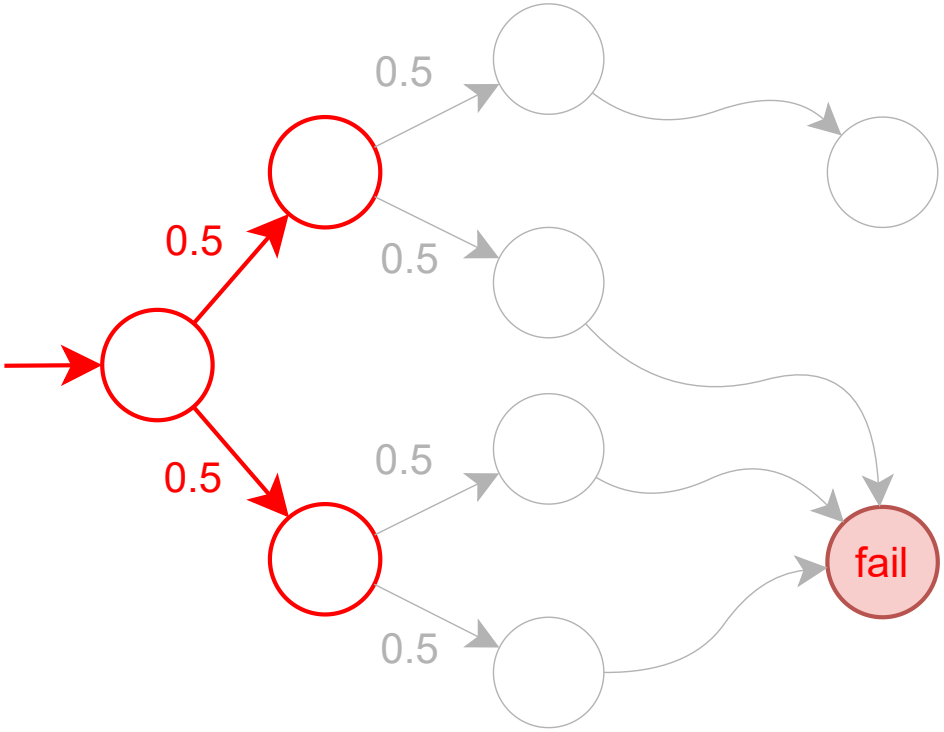
- use val($s$) as a **probabilistic shortcut** from state $s$ to the target
- val($s$) = minimum reachability probability through all possible continuations of $s$ **within the family**



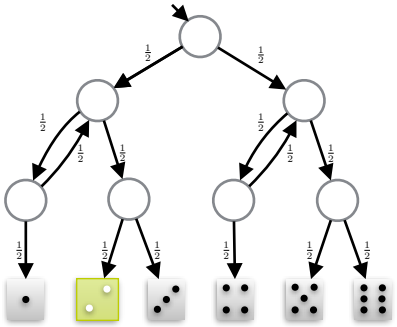
## Counterexamples modulo family

[TACAS'21]

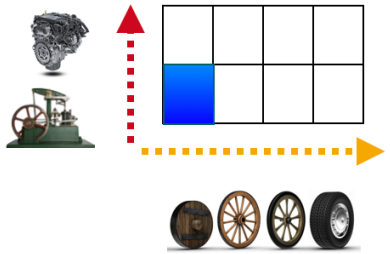
- use val($s$) as a **probabilistic shortcut** from state $s$ to the target
- val($s$) = minimum reachability probability through all possible continuations of $s$ **within the family**



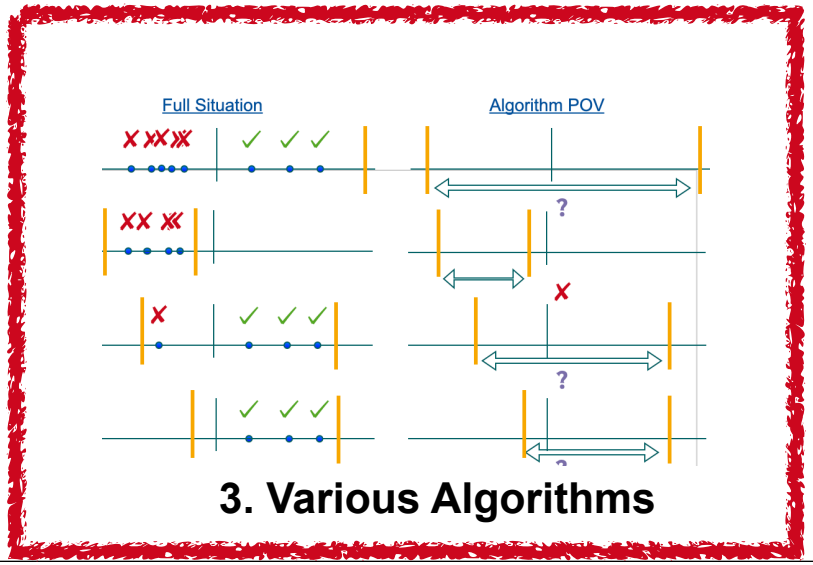
# Overview



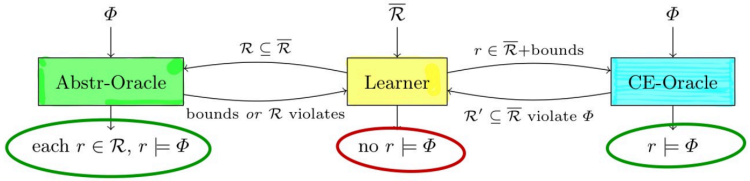
1. Probabilistic Systems & Parameter Synthesis Recap



2. Discrete setting w topology changes



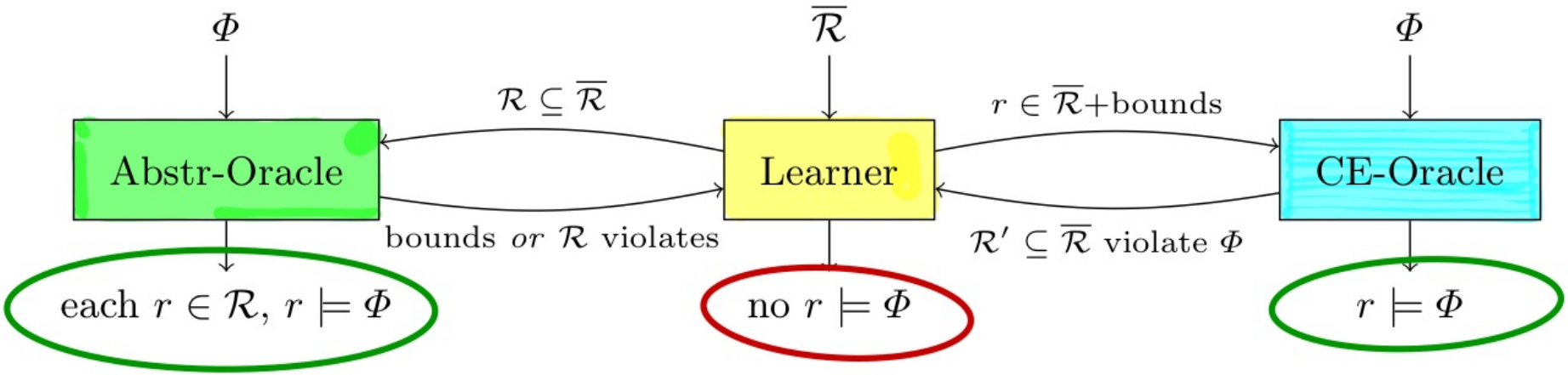

3. Various Algorithms




4. PAYNT and Examples

# The Best of Both Worlds

[TACAS'21]



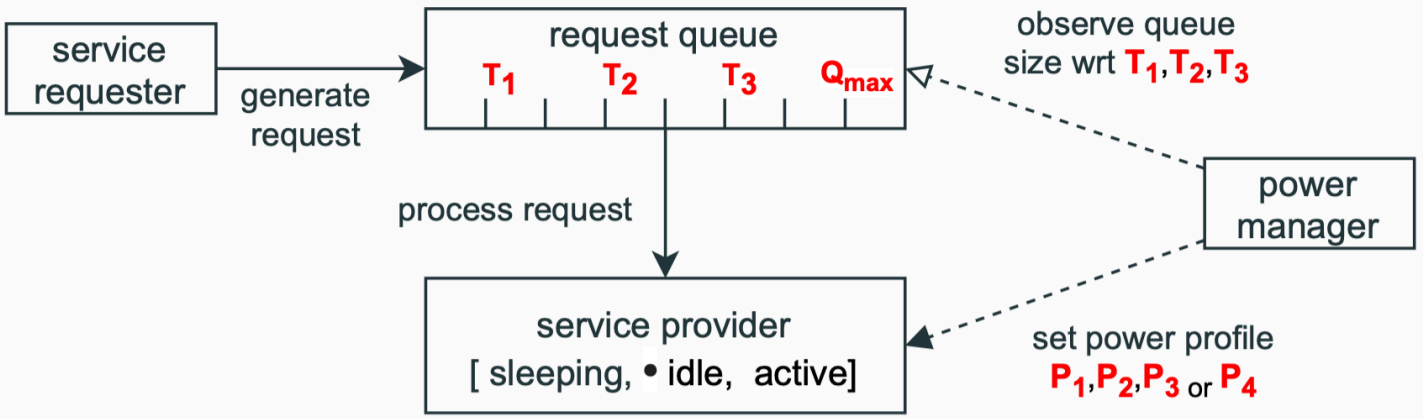

Implemented in PAYNT

using Python, building upon Z3 and the probabilistic model checker Storm

# PAYNT example (1)

Details: see
https://github.com/randriu/synthesis

[CAV'21]

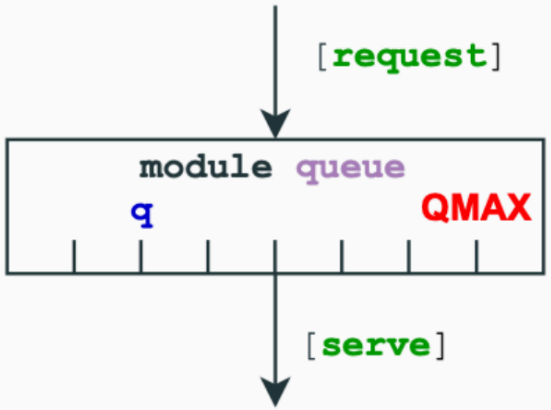



- specification:
  - expected number of lost requests must be at most 1
  - expected power consumption is minimal
- problem: how to choose $Q_{max}, T_1, T_2, T_3, P_1, P_2, P_3, P_4$ in order to satisfy the specification?

# PAYNT example (2)

[CAV'21]

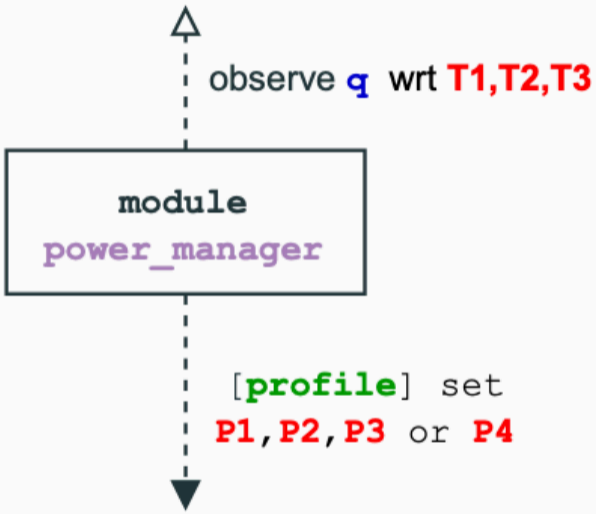
```
R{"lost"}<=1 [F finished];
R{"power"}min=? [F finished];
```




```
...

hole int QMAX in {1,2,3,4,5,6,7,8,9,10};

module queue
    q : [0..QMAX] init 0;
    lost : [0..1] init 0;

    [request] q < QMAX -> (q'=q+1) & (lost'=0);
    [request] q = QMAX -> (lost'=1);

    [serve] q > 0 -> (q'=q-1) & (lost'=0);
endmodule

...
```

# PAYNT example (3)

[CAV'21]




```
...

hole double T1 in {0.0,0.1,0.2,0.3};
hole double T2 in {0.4,0.5,0.6};
hole double T3 in {0.7,0.8,0.9};

// 0 - sleep, 1 - idle, 2 - active
hole int P1 in {0,1,2};
hole int P2 in {0,1,2};
hole int P3 in {0,1,2};
hole int P4 in {0,1,2};

module power_manager
    pm : [0..2] init 0;

    [profile] q <= T1*QMAX -> (pm'=P1);
    [profile] q > T1*QMAX & q <= T2*QMAX -> (pm'=P2);
    [profile] q > T2*QMAX & q <= T3*QMAX -> (pm'=P3);
    [profile] q <= T3*QMAX -> (pm'=P4);
endmodule

...
```

# Using PAYNT

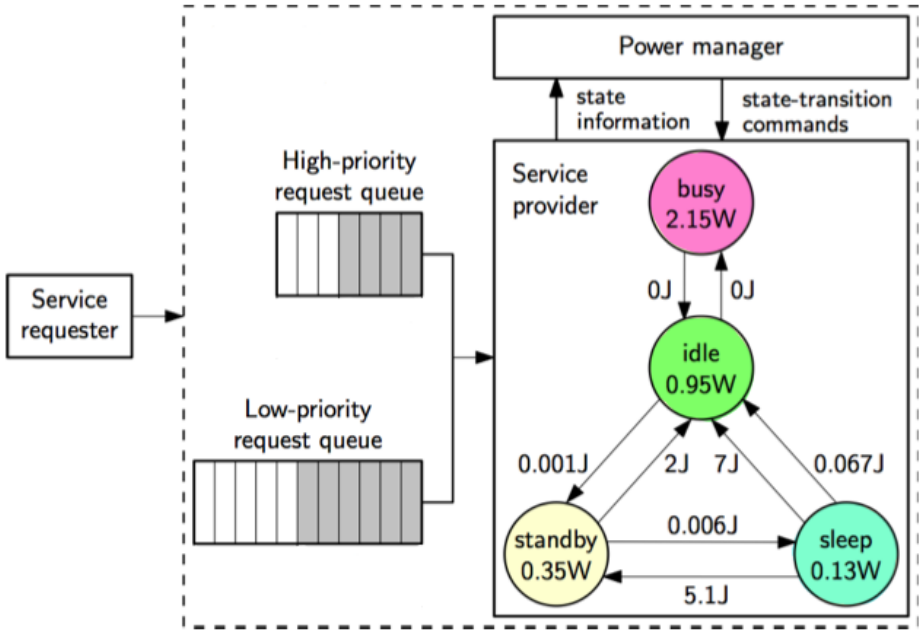
[CAV'21]

```
> python3 paynt/paynt.py --project sketch_path \
--properties sketch.properties
```

```
method: hybrid, synthesis time: 200.0 s
number of holes: 8, family size: 29160
super MDP size: 1502, average MDP size: 903, MDP
checks: 238, iterations: 125
average DTMC size: 234, DTMC checks: 26574,
iterations: 13287 optimal: 9100.064246

hole assignment:
P1=1,P2=2,P3=2,P4=2,T1=0.1,T2=0.4,T3=0.7,QMAX=5
```

# Sketching Probabilistic Programs

[Benini *et al*, IEEE CAD 1999]



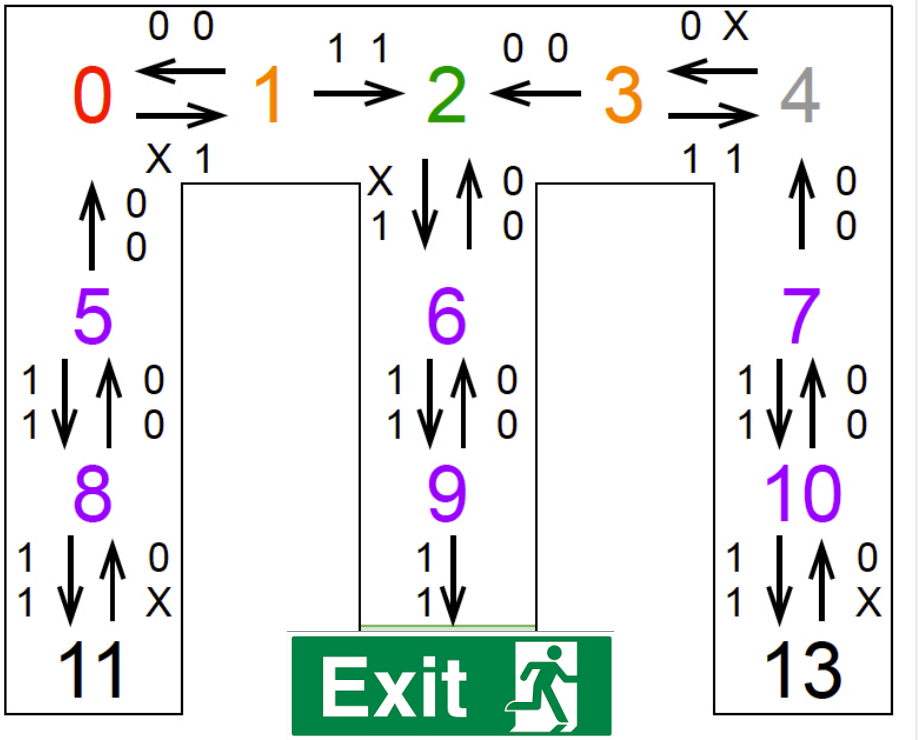

## Challenge

- Synthesise guards and updates in DPM control program with 16 holes
- Specification = conjunction of expected #lost reqs and energy consumption

## Results (16 parameters)

- Family size = 43,000,000 control programs of average size of 3,600 states
- Our approach: 9 hours; baseline: > 1 month

# Maze POMDP




- 22 parameters
- 9,400,000 possible strategies
- 200 states average MC size

- baseline: about two days
- our approach: 1 hour

Minimise the expected #steps to exit the maze

# Herman's Randomised Self-Stabilisation

Body Level One
Body Level Two

- Process $i$ performs:
  - if $x_i = x_{i-1}$, then $x_i := \begin{cases} 0 & \text{with probability } p \\ 1 & \text{with probability } 1-p \end{cases}$
  - if $x_i \neq x_{i-1}$ then $x_i := x_{i-1}$
- Process possesses **token** if $x_i$ equals $x_{i-1}$


Performance metric = expected convergence time

# Improving Herman's Randomised Self-Stabilisation

- Process $i$ performs:
  - if $x_i = x_{i-1}$, then $x_i := \begin{cases} 0 & \text{with probability } p \\ 1 & \text{with probability } 1-p \end{cases}$
  - if $x_i \neq x_{i-1}$ then $x_i := x_{i-1}$


Can we do better?

Use a single bit of memory and 25 different coin biases

- 7 parameters
- 3,100,000 possible strategies
- 1,100 states average MC size

- baseline: about 1,5 days
- our approach: 17 minutes

Initially use most fair coins, memory 0, and later highly unfair coins, memory 1

# Literature

## References


Milan Ceska, Christian Hensel, Sebastian Junges, Joost-Pieter Katoen:
**Counterexample-guided inductive synthesis for probabilistic systems.** Formal Aspects Comput. 33(4-5): 637-667 (2021)

Roman Andriushchenko, Milan Ceska, Sebastian Junges, Joost-Pieter Katoen:
**Inductive Synthesis for Probabilistic Programs Reaches New Horizons.** TACAS (1) 2021: 191-209

Roman Andriushchenko, Milan Ceska, Sebastian Junges, Joost-Pieter Katoen, Simon Stupinský:
**PAYNT: A Tool for Inductive Synthesis of Probabilistic Programs.** CAV (1) 2021: 856-869

Milan Ceska, Nils Jansen, Sebastian Junges, Joost-Pieter Katoen:
**Shepherding Hordes of Markov Chains.** TACAS (2) 2019: 172-190

Sebastian Junges:
**Parameter synthesis in Markov models.** RWTH Aachen University, Germany, 2020

## Conclusion Future directions

- Lots of applications for discrete parameters
- Abstraction refiinment and CEGIS are both major steps forward
- A lot remains to be done

- Integrating more/different counterexamples, see work by Baier et al.
- Dedicated support for richer properties
- More oracles
- Dedicated support for POMDP controllers
- Infinite parameter domains
- Better modelling support