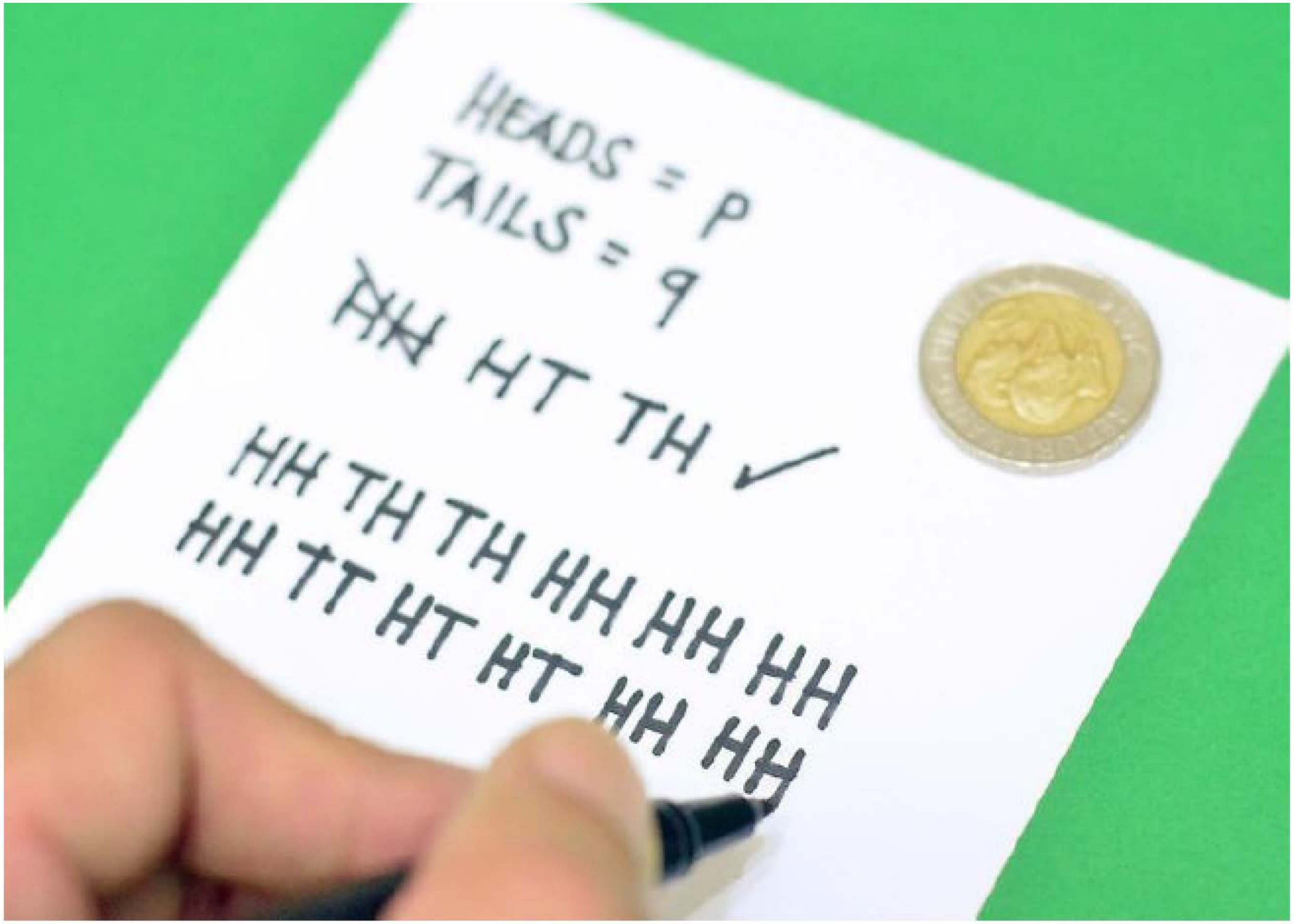
# On Synthesising Probabilistic Models and Programs


**Joost-Pieter Katoen**

joint with ***Milan Češka, Roman Andriushchenko, Sebastian Junges***

RWTH AACHEN UNIVERSITY — BRNO UNIVERSITY OF TECHNOLOGY — Berkeley UNIVERSITY OF CALIFORNIA

# How to Pick the Optimal Bias?



# Self-Stabilisation

A distributed algorithm is self-stabilising iff:

- Convergence:
  Starting from an arbitrary state, it always converges to a legitimate state
- Closure:
  And it remains in a legitimate set of states thereafter in absence of faults

A self-stabilising algorithm:

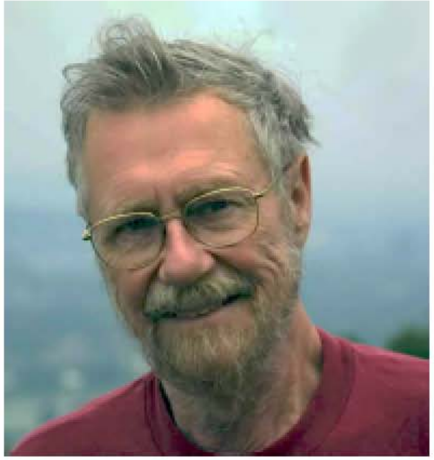
- Works correctly for every initialisation
- Recovers from the occurrence of transient faults

A key concept in fault-tolerant distributed computing

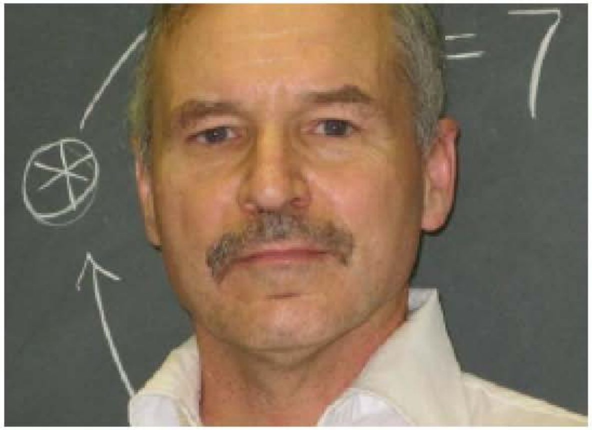
# Mission Impossible?

Dijkstra 1986: Self-stabilisation in anonymous networks is impossible

Herman's escape 1990: use randomisation



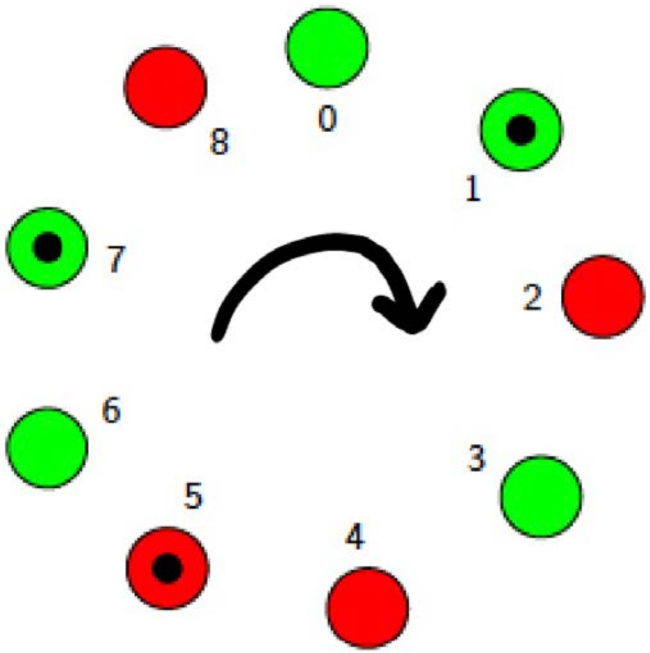
Edsger W. Dijkstra



Ted Herman

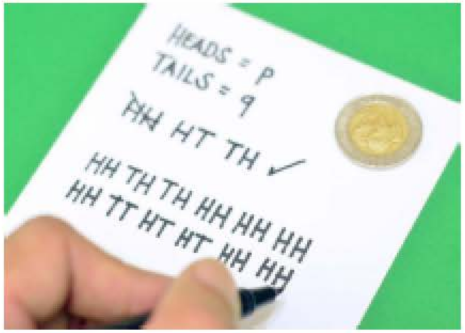
# Herman's Randomised Self-Stabilisation

- $N$+1 (odd) synchronous processes $0, \ldots, N$ form a directed ring
- Process $i$ has a Boolean variable $x_i \in \{0, 1\}$
- Processes have access to their neighbour's variables



# Herman's Randomised Self-Stabilisation

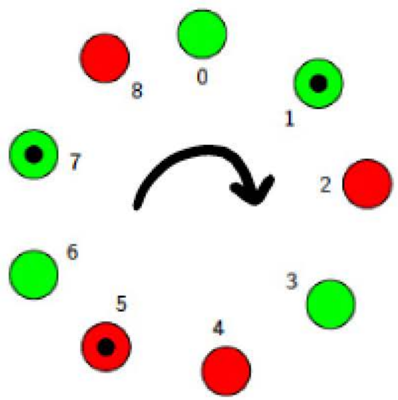
- Process $i$ performs:
  - if $x_i = x_{i-1}$, then $x_i := \begin{cases} 0 & \text{with probability } p \\ 1 & \text{with probability } 1-p \end{cases}$
  - if $x_i \neq x_{i-1}$ then $x_i := x_{i-1}$



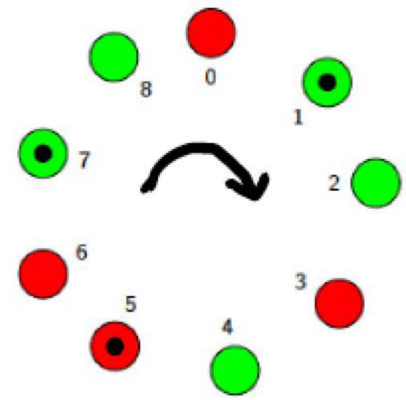

- Process possesses **token** if $x_i$ equals $x_{i-1}$

Performance metric = expected convergence time

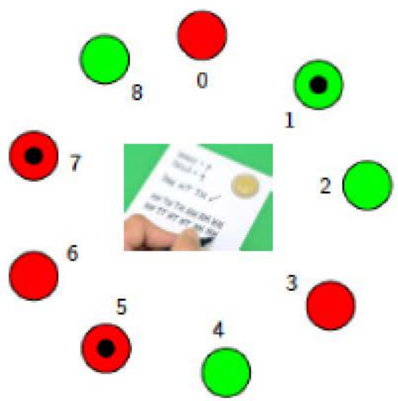
# A Round of Herman's Protocol



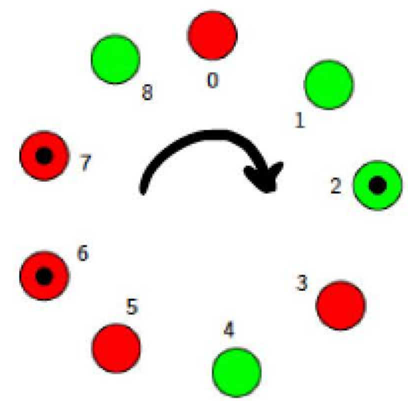

current configuration




toggle all processes w/o token

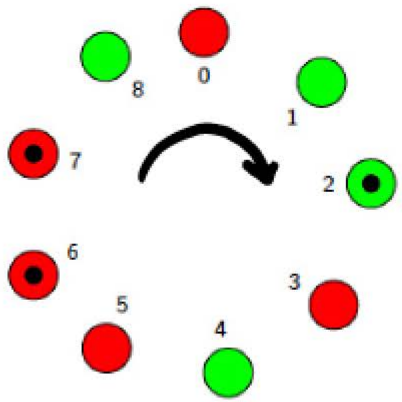



flip coins for every token

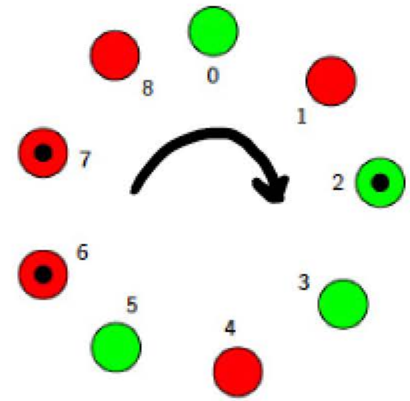



"assign" the tokens

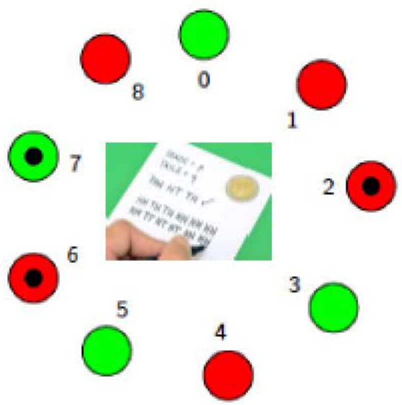
# Another Round



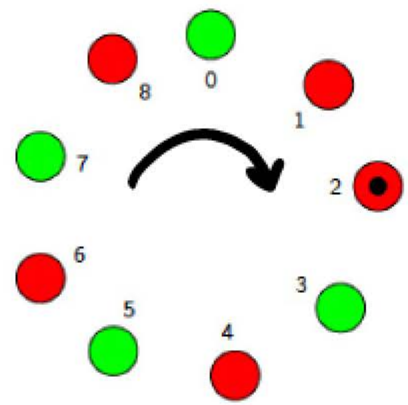

current configuration




toggle all processes w/o token




flip coins for every token



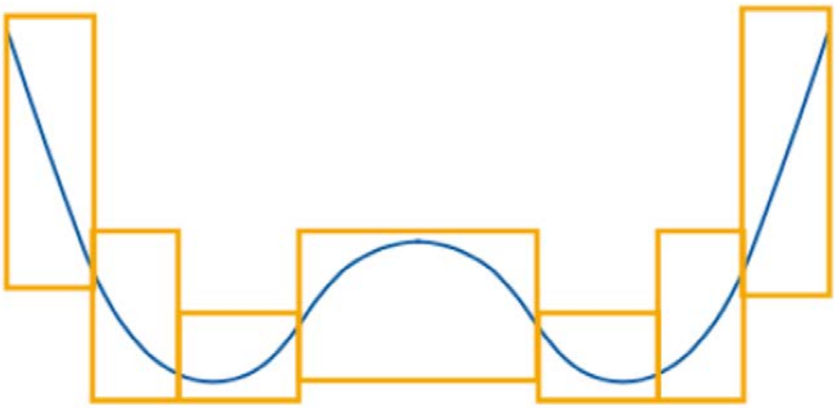

"assign" the tokens

# Parameter Synthesis Results

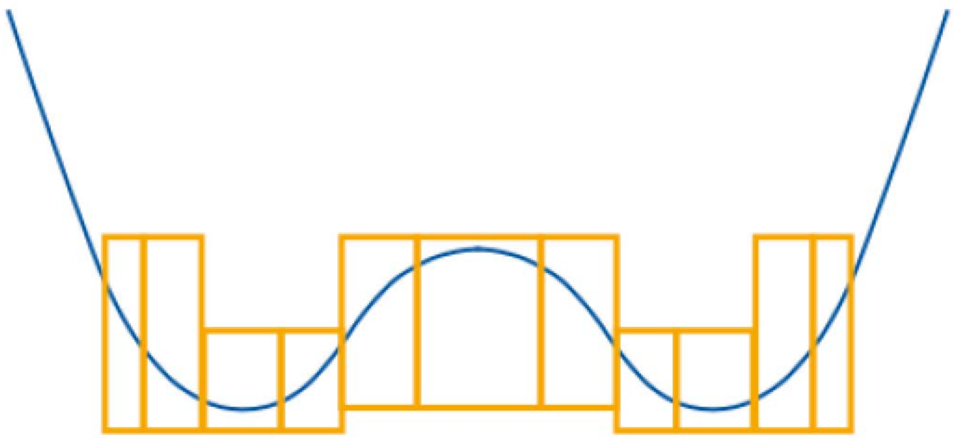
| $N$+1 | #states | #trans | $p$ | ECT | time(s) |
|---|---|---|---|---|---|
| 7 | 129 | 2,316 | [0.496, 0.504] | [4.493, 4.493] | 2.68 |
| 9 | 513 | 21K | [0.452, 0.465], [0.535, 0.548] | [7.914, 7.921] | 4.5 |
| 11 | 2,049 | 180K | [0.352, 0.382], [0.618, 0.648] | [12.097, 12.102] | 9.1 |
| 13 | 8,193 | 1.6M | [0.322, 0.344], [0.656, 0.678] | [16.942, 16.949] | 36.1 |
| 15 | 32,769 | 14.4M | [0.301, 0.319], [0.681, 0.699] | [22.445, 22.453] | 310 |
| 17 | 131,073 | 129M | [0.291, 0.304], [0.696, 0.709] | [28.603, 28.610] | 3480 |
| 19 | 524,289 | 1,162M | [0.279, 0.292], [0.708, 0.721] | [35.406, 35.416] | $>$ 24h |

Abstraction-refinement of parameter regions

# Iterative Parameter Synthesis



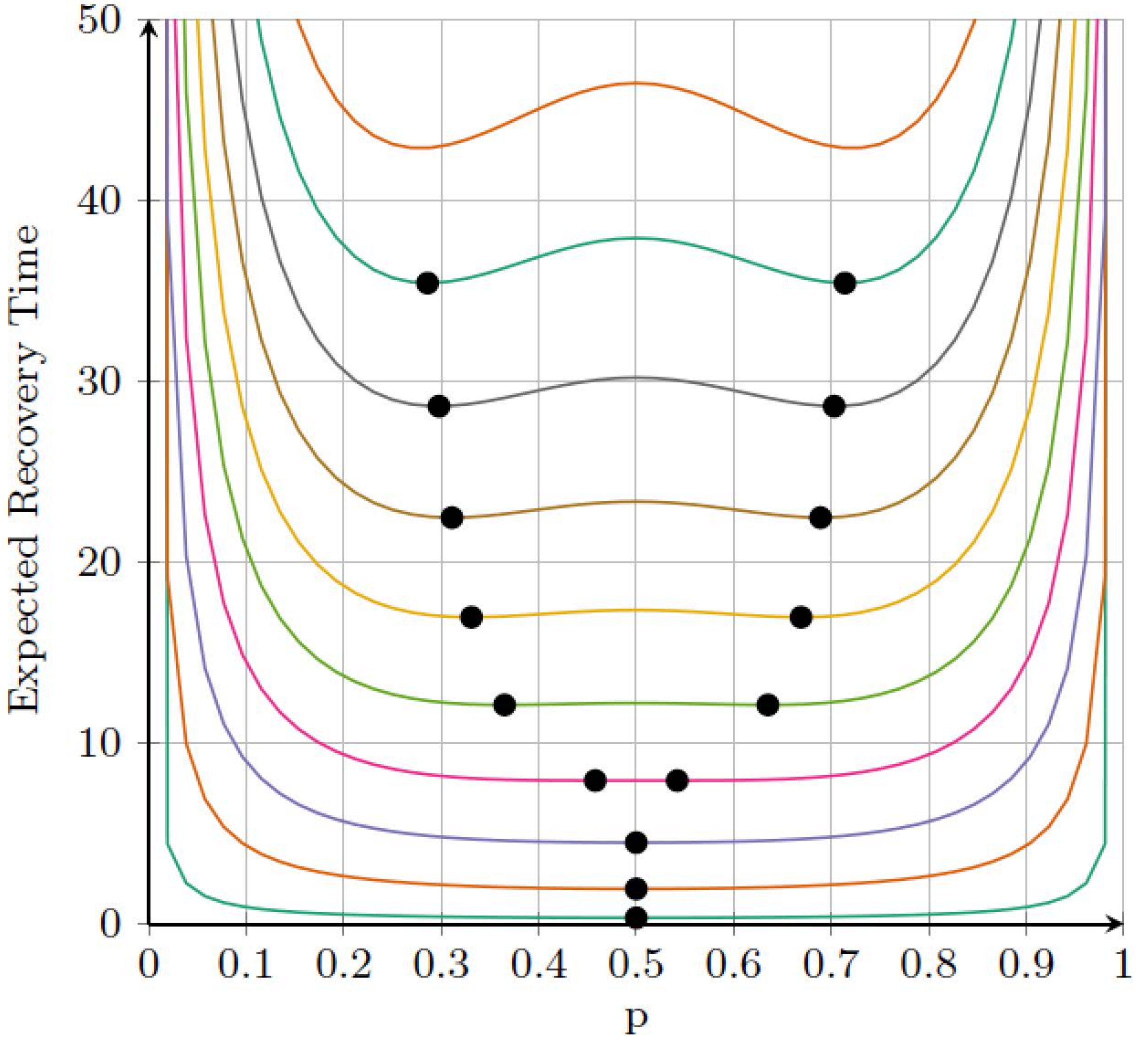
(a) Iteration 1



(b) Iteration 2

# Parameter Synthesis Results

[Aflaki *et al.*, SRDS 2017]

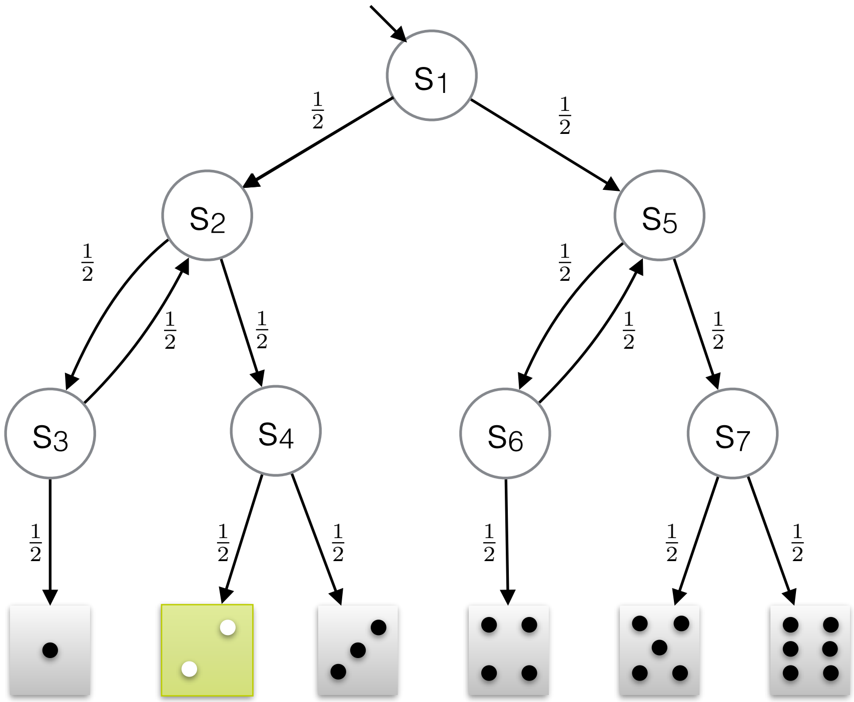


# Parameter Synthesis

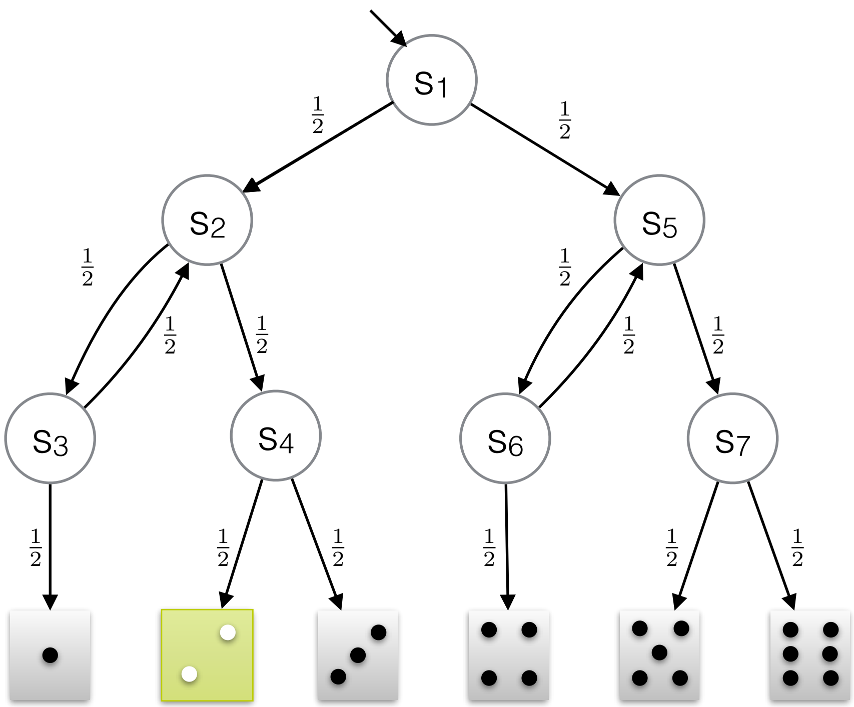
We can automatically synthesise
the parameter values
that minimise the expected convergence time

# Parameter Synthesis

# Parameter Synthesis



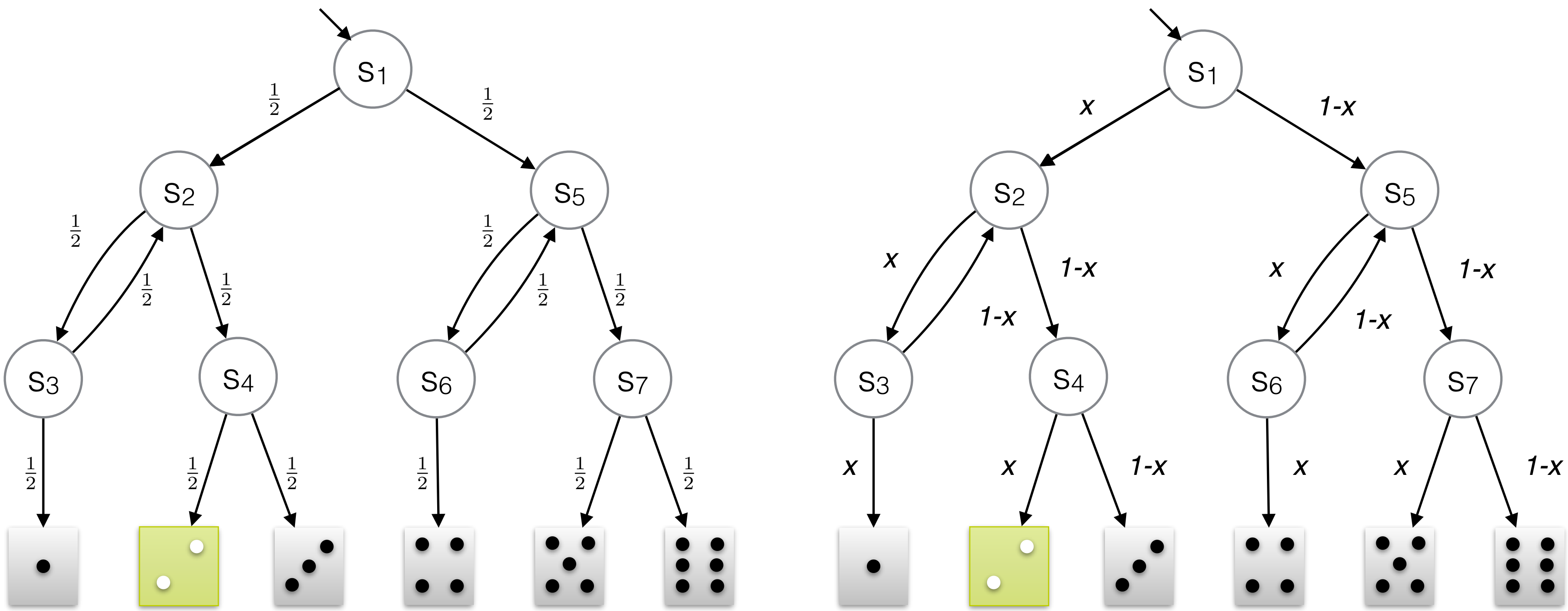
# Parameter Synthesis




Probability to reach 

Expected # steps to 

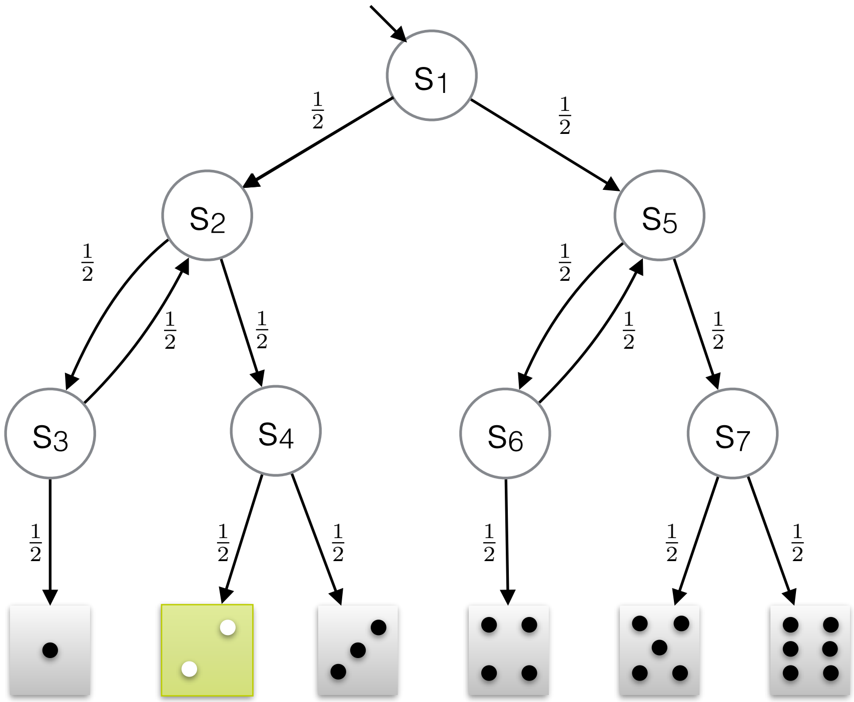
## Uncountable sets of finite Markov chains

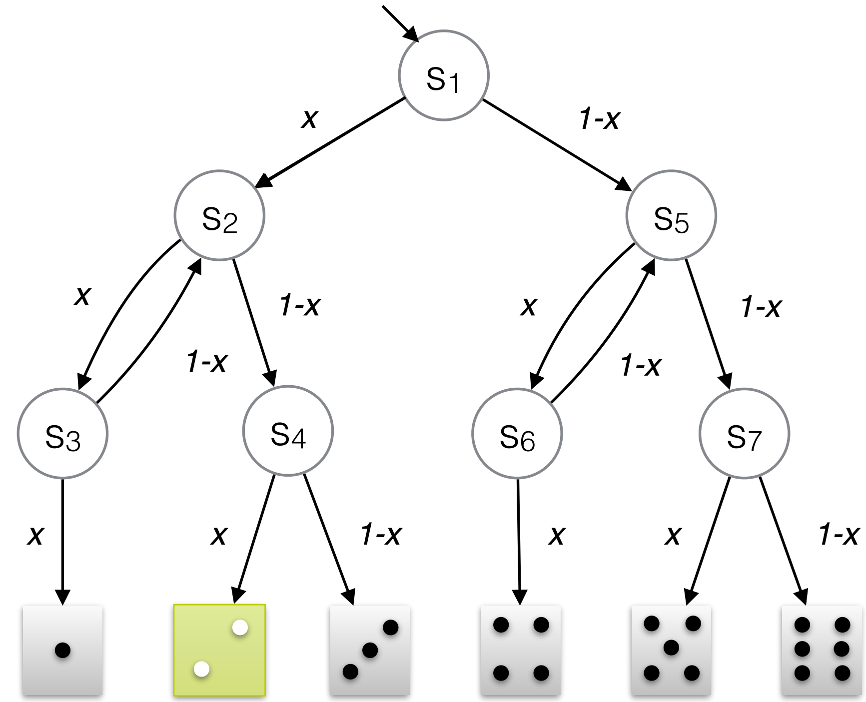



Probability to reach

Expected # steps to

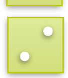
# Uncountable sets of finite Markov chains




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

Probability to reach 

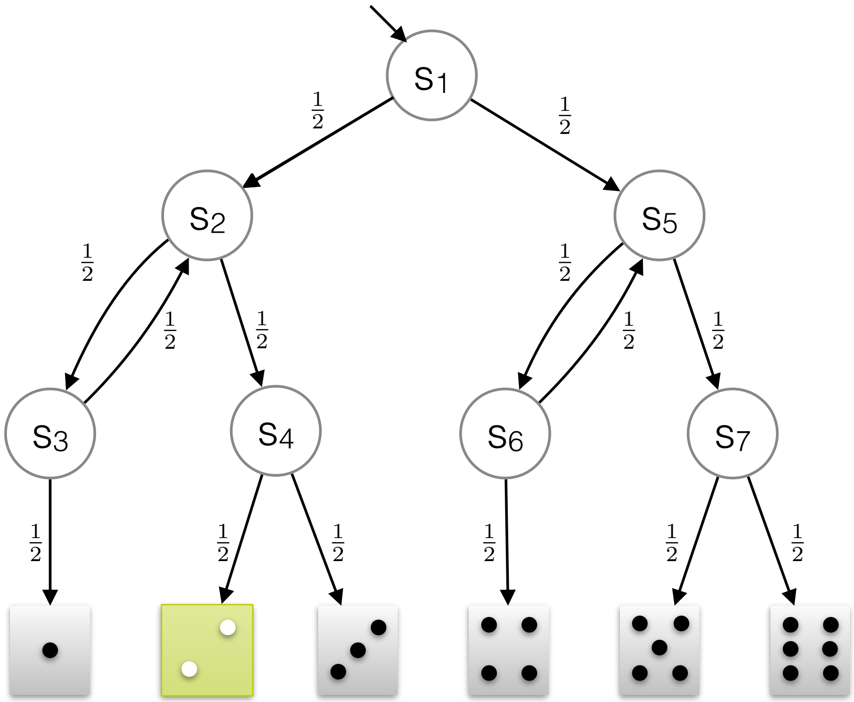
Expected # steps to 

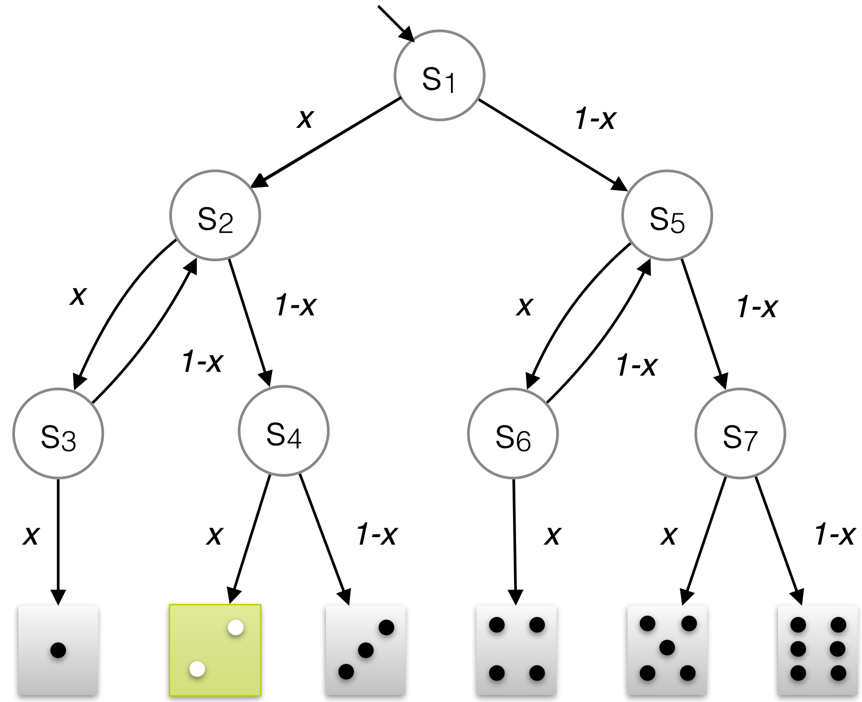
$(x^2 - x^3) / (x^2 - x + 1)$ in [9/60, 11/60]?

# Parameter Synthesis

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

## Uncountable sets of finite Markov chains




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]?

Probability to reach 

Expected # steps to 

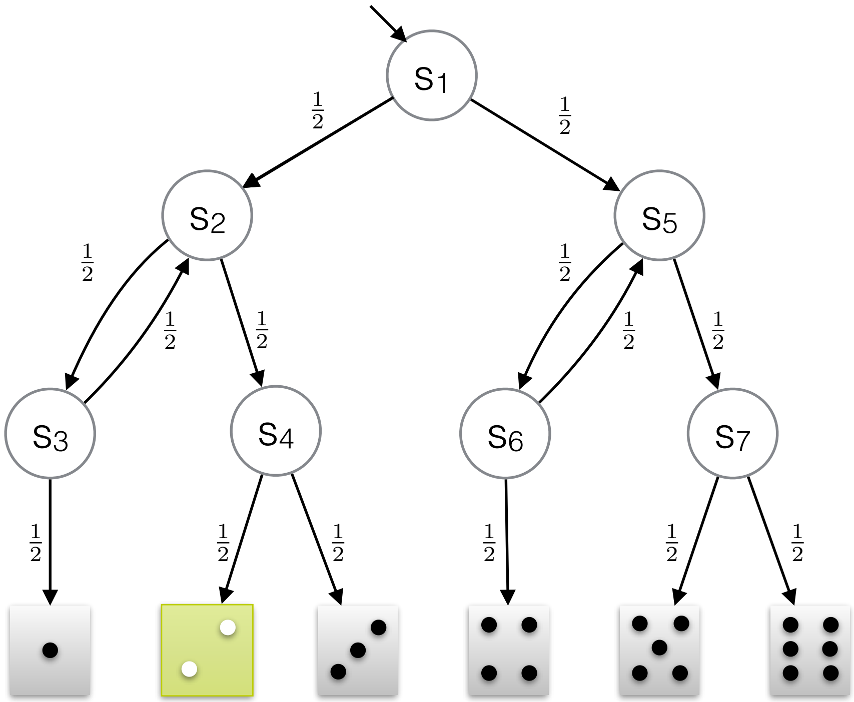
# Model Synthesis

What if we allow for topology changes?

Thus: focus on tweaking the control structure

# Model Synthesis

## finite Markov chains

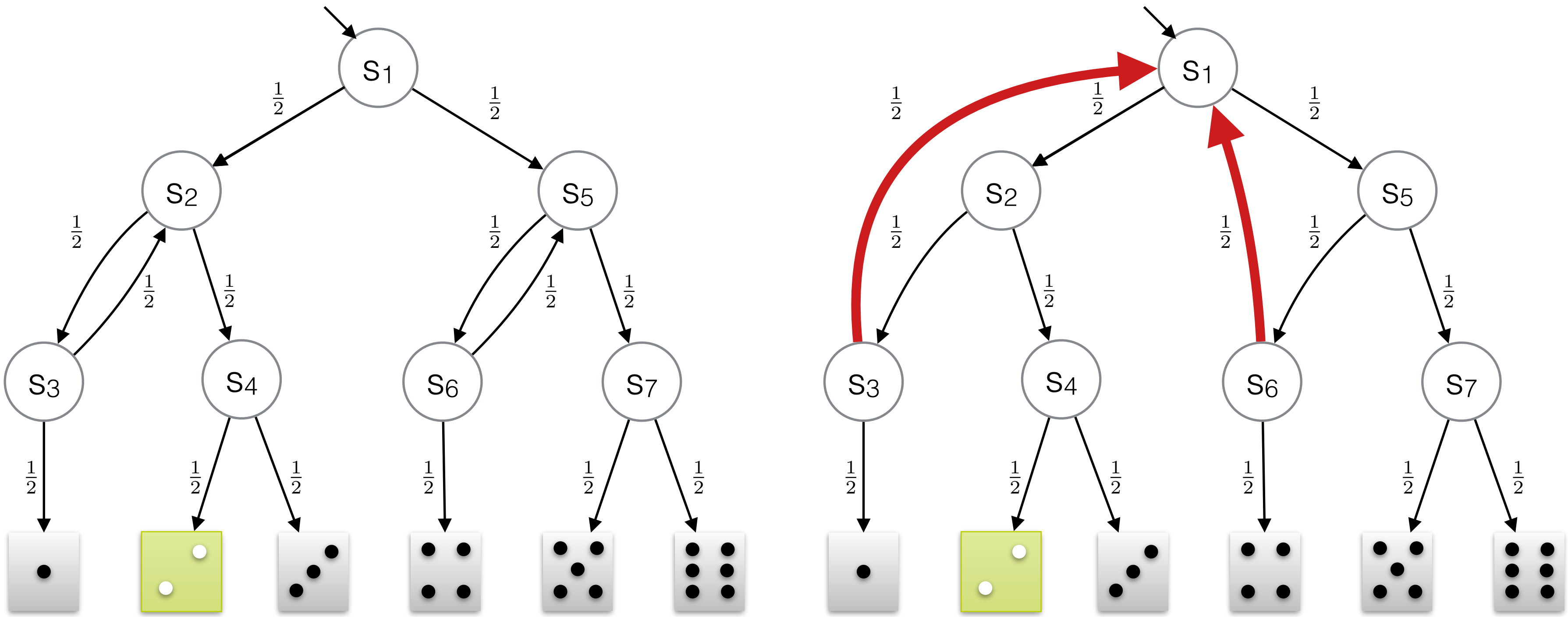



Probability to reach 

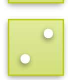
Expected # steps to 

# Model Synthesis

**finite Markov chains**

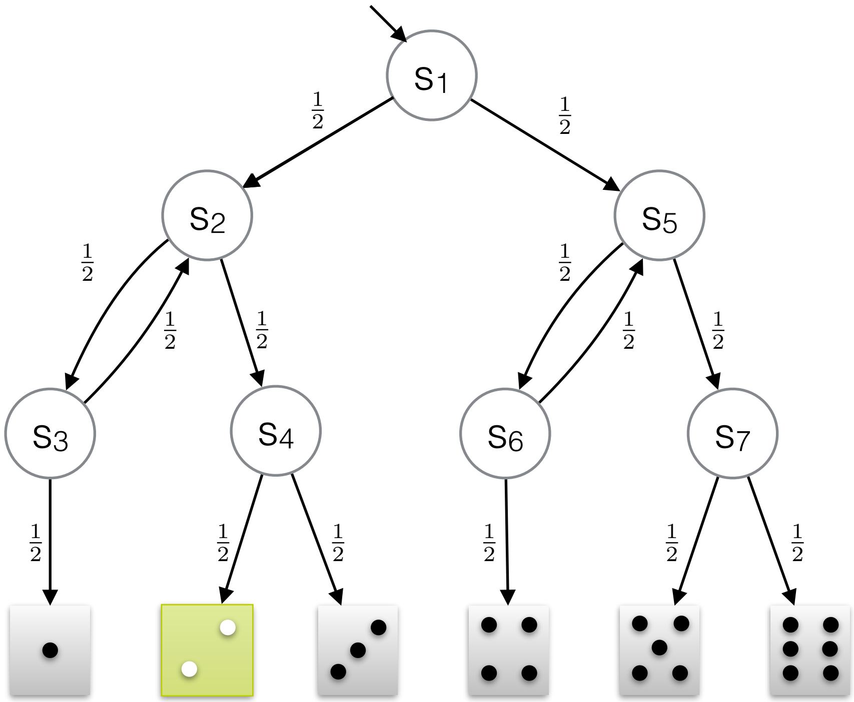


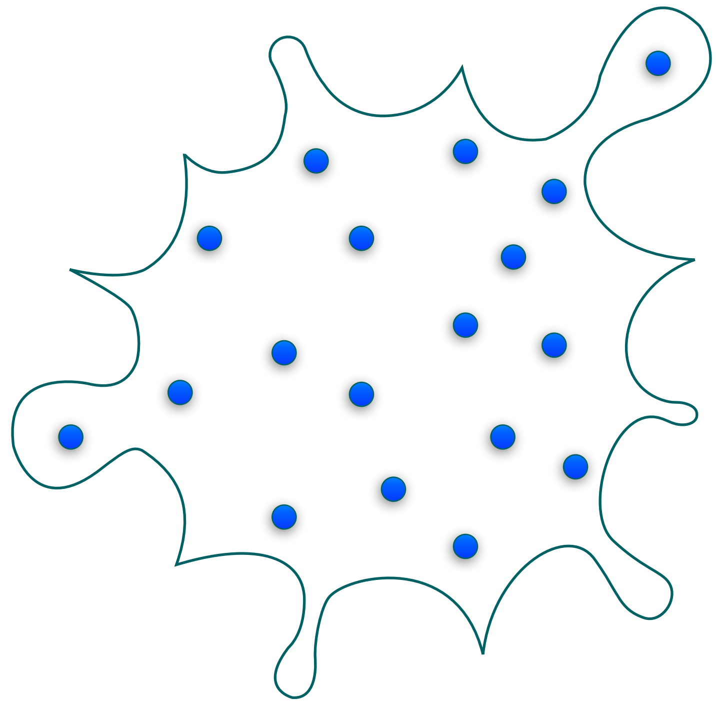

Probability to reach 

Expected # steps to 

# Model Synthesis

## Huge, finite sets of finite Markov chains



each ● = Markov chain

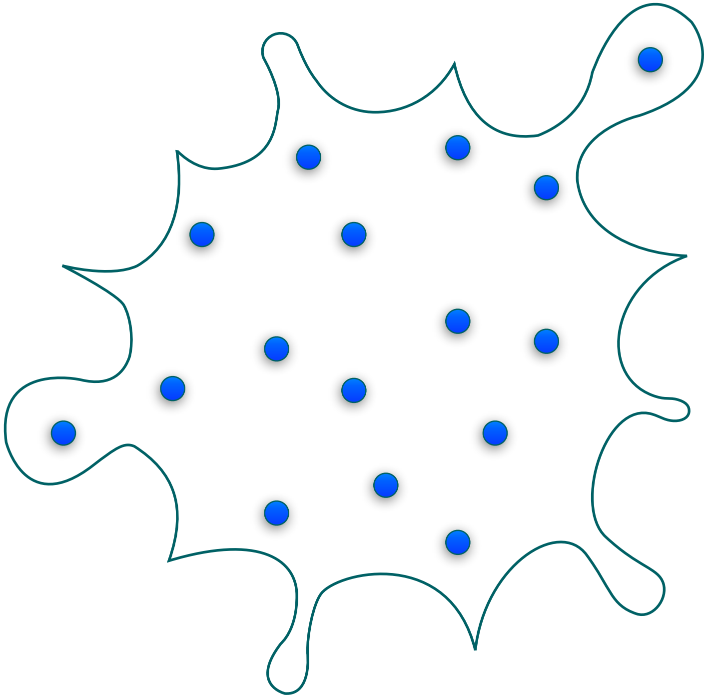
Probability to reach 

Expected # steps to 

# Synthesis Goals

**Inputs:**

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

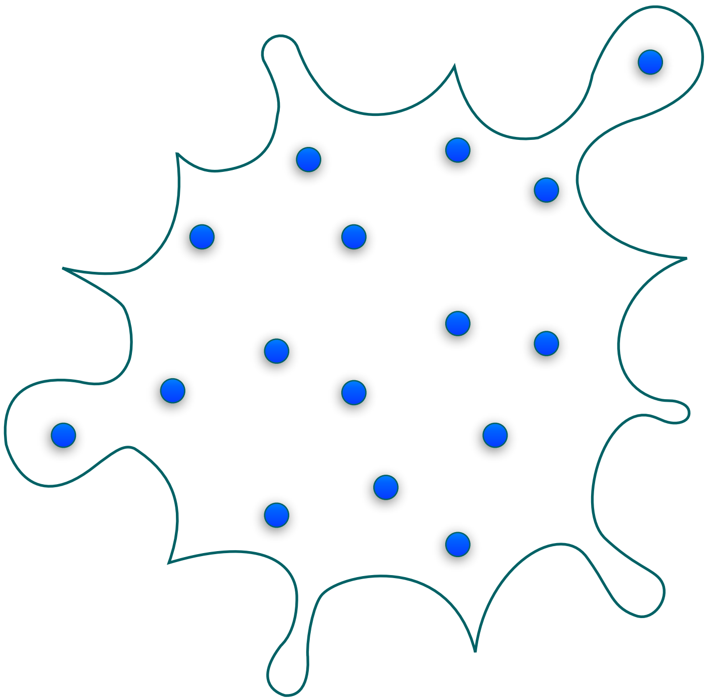


# Synthesis Goals

**Inputs:**

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

**Synthesis goals:**



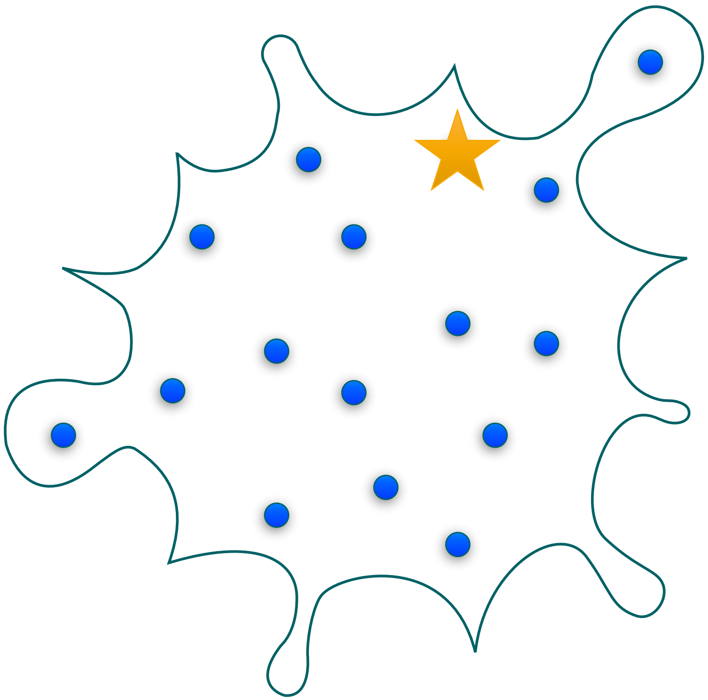
# Synthesis Goals

**Inputs:**

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

**Synthesis goals:**

1. Find a realisation (an MC) satisfying ***f***.



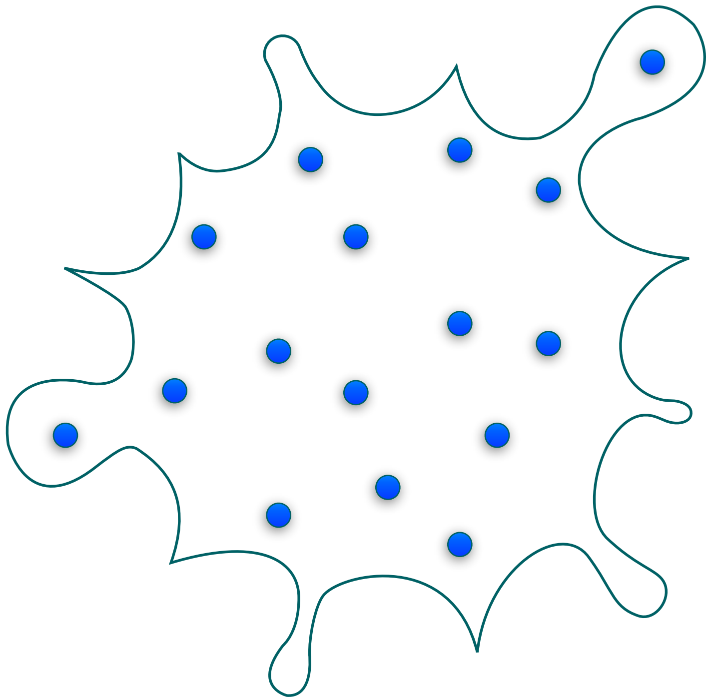
# Synthesis Goals

**Inputs:**

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

**Synthesis goals:**

1. Find a realisation (an MC) satisfying ***f***.



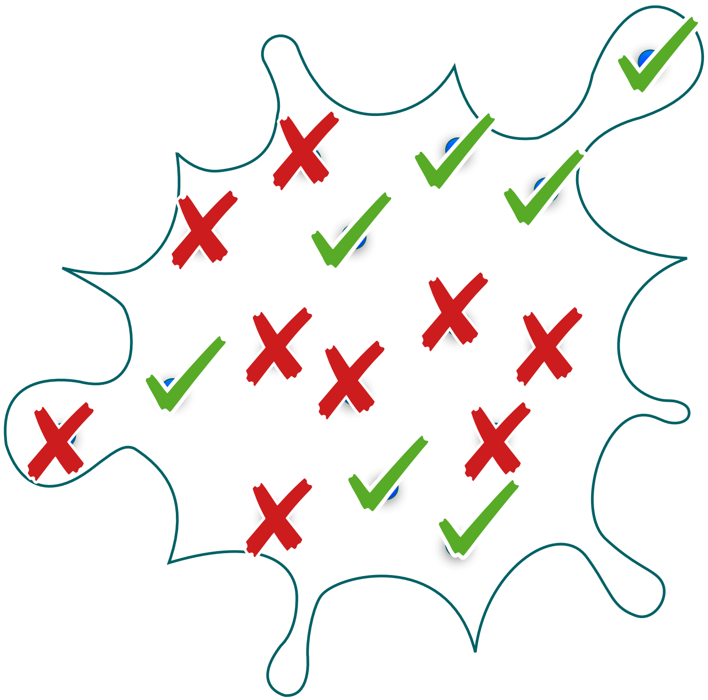
# Synthesis Goals

**Inputs:**

a Markov chain 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***.



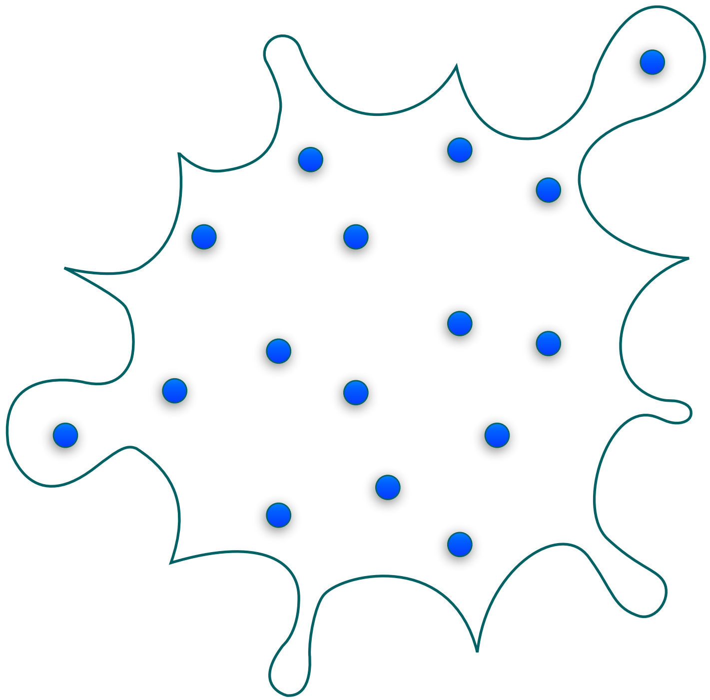
## Synthesis Goals

**Inputs:**

a Markov chain 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***.



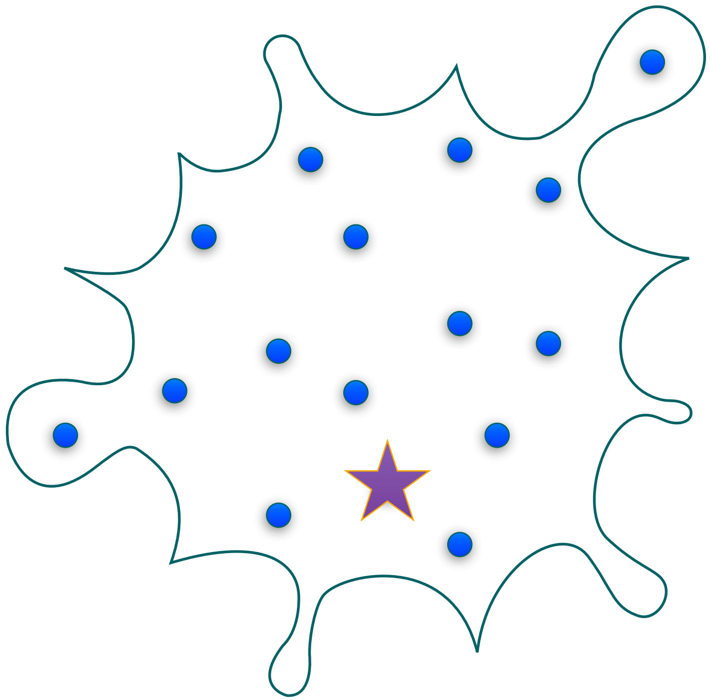
# Synthesis Goals

**Inputs:**

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

**Synthesis goals:**

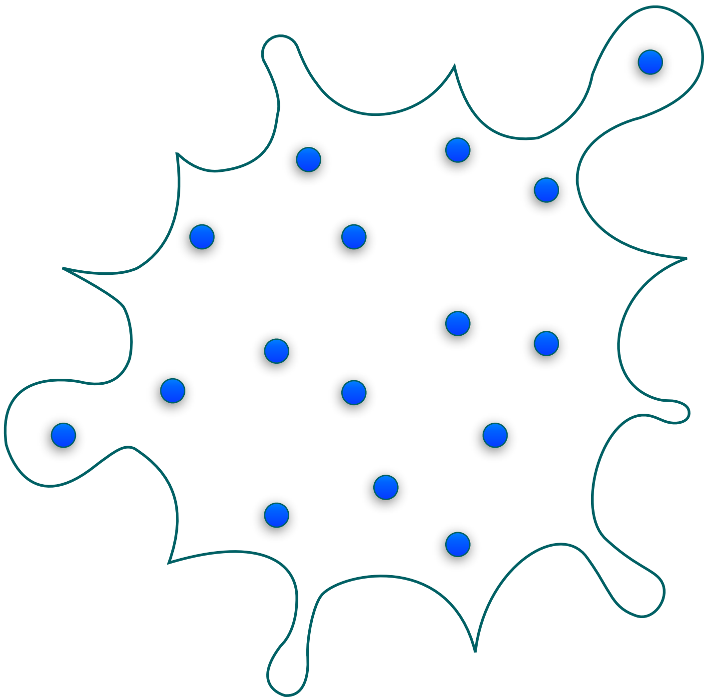
1. Find a realisation (an MC) satisfying ***f***.
2. Find all realisations satisfying ***f***.
3. Find the realisation with the maximal probability to reach ***G***.



# Synthesis Goals

**Inputs:**

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



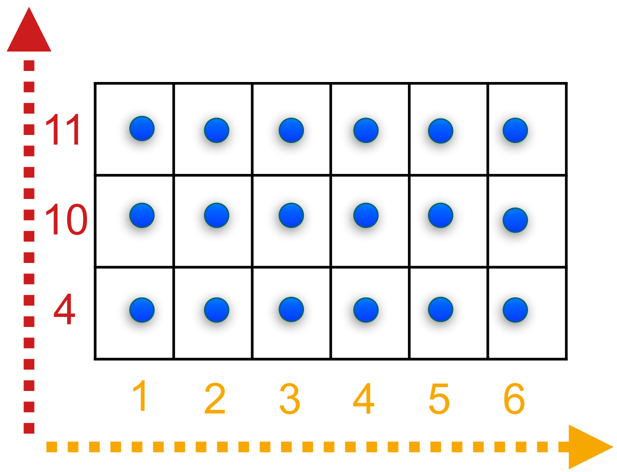
**Synthesis goals:**

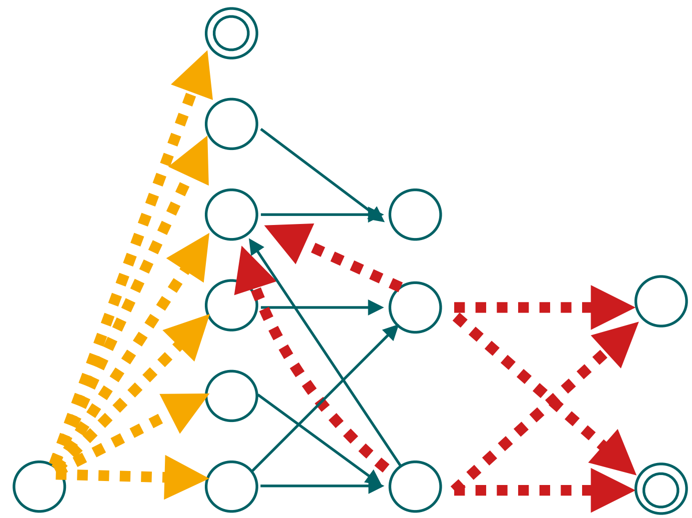
1. Find a realisation (an MC) satisfying ***f***.
2. Find all realisations satisfying ***f***.
3. Find the realisation with the maximal probability to reach ***G***.

**Cost-based variants:**

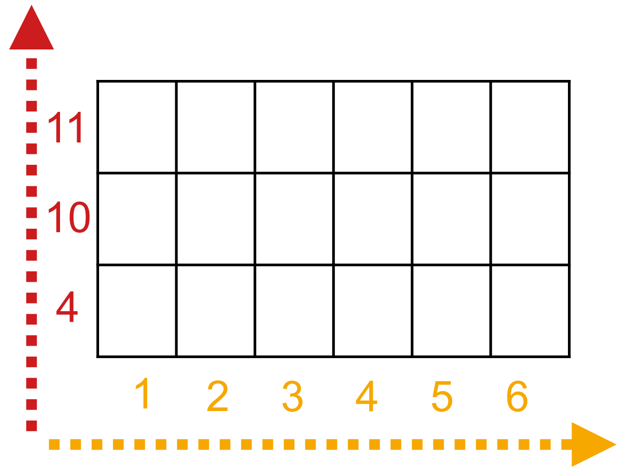
4. Find the cheapest realisation satisfying ***f***.
5. Find all within-budget realisations satisfying ***f***.

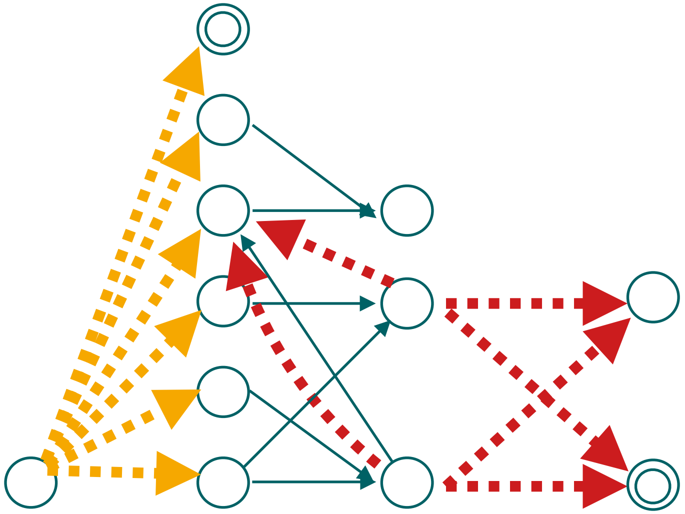
# Families



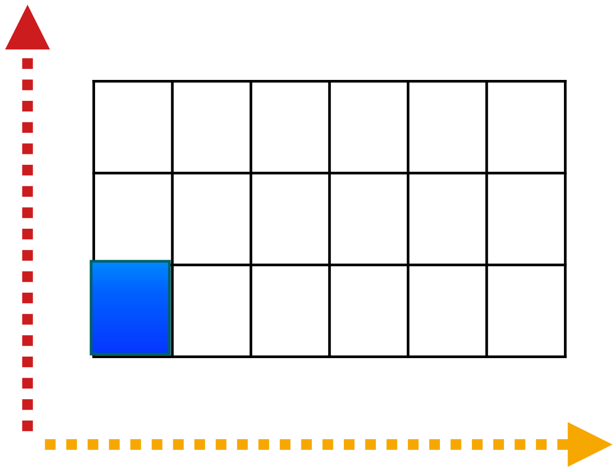



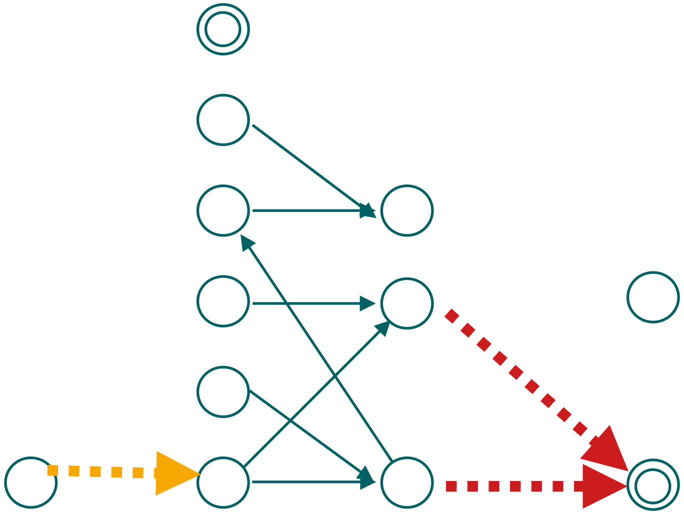
# Families



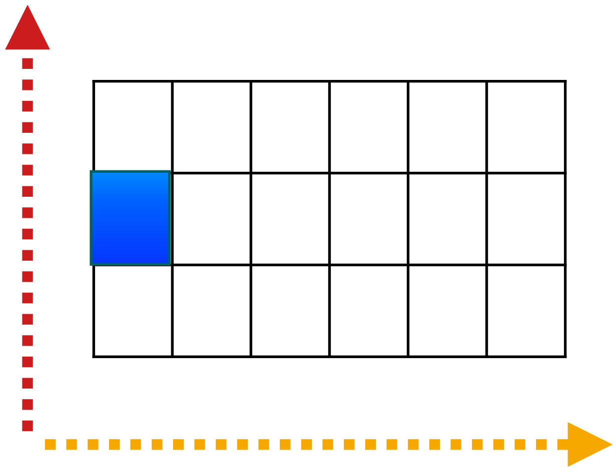



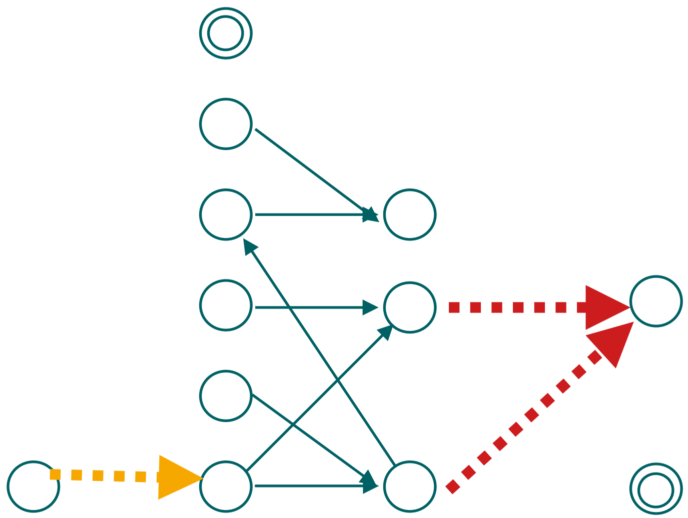
# Families



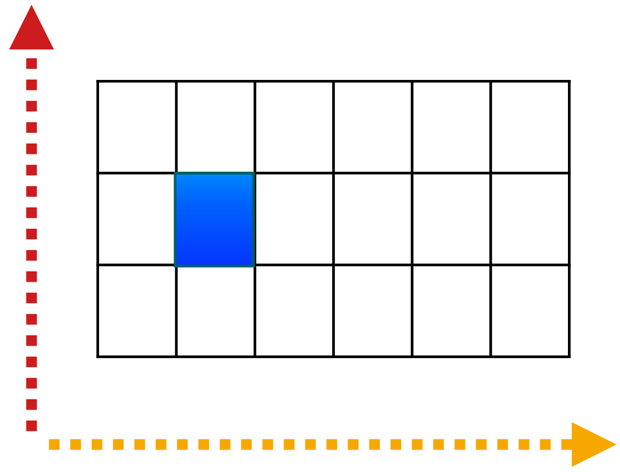


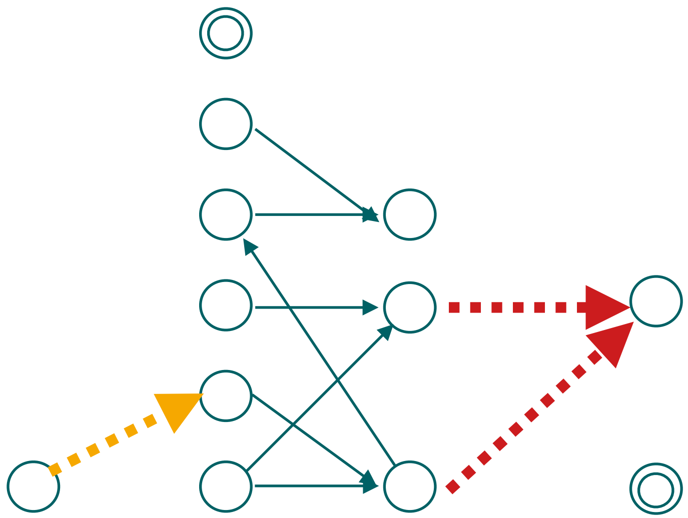
# Families



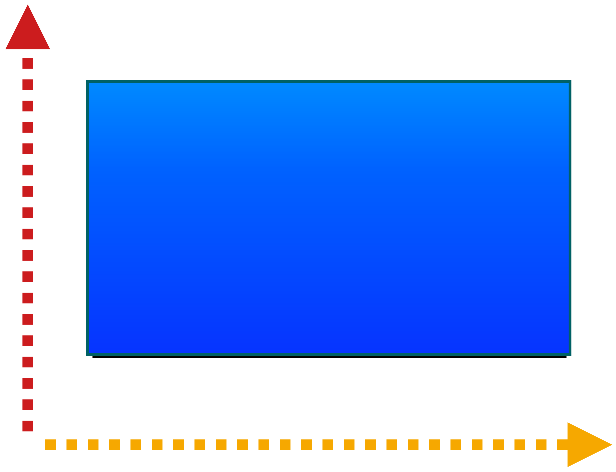


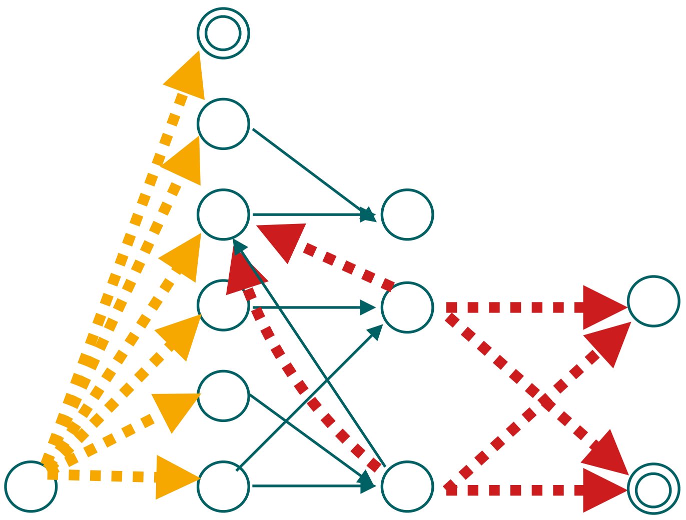
# Families



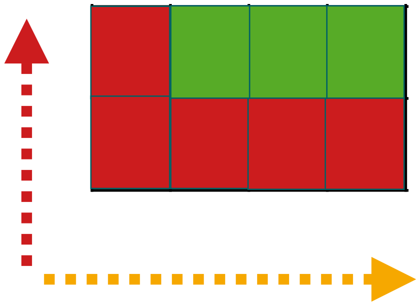


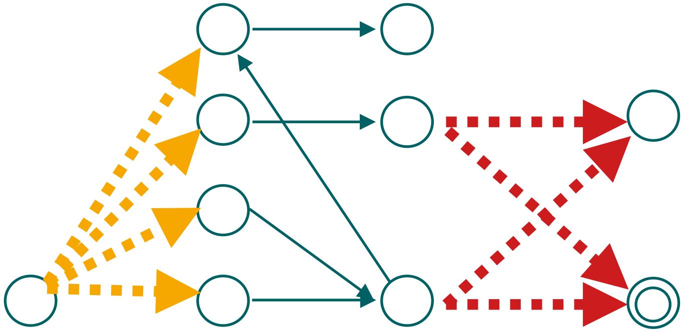
# Families





# Objective: Partition Parameter Space

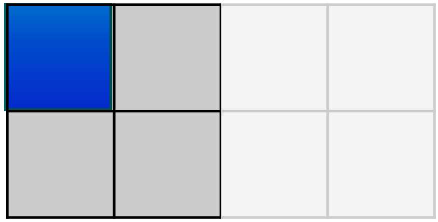




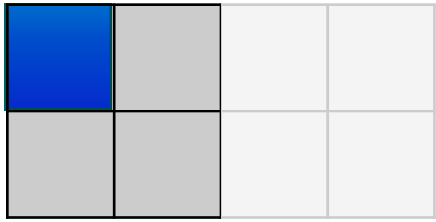
# The Baseline: Enumeration

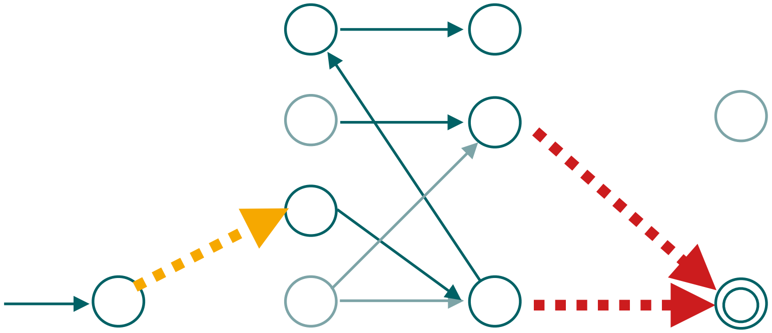
Fix the parameters, build the model, check

# The Baseline: Enumerate



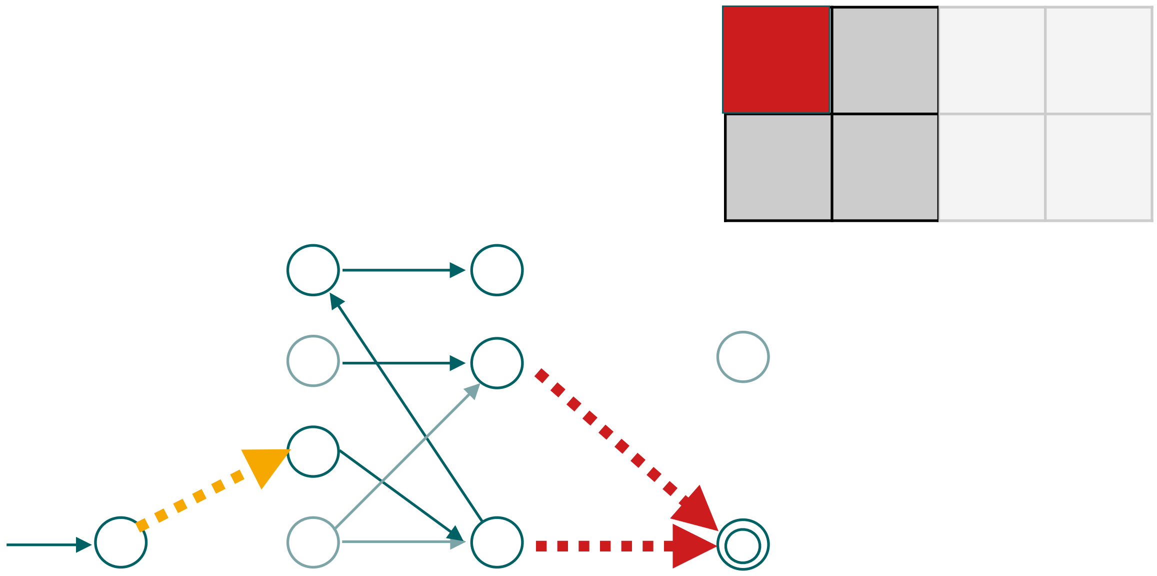
# The Baseline: Enumerate



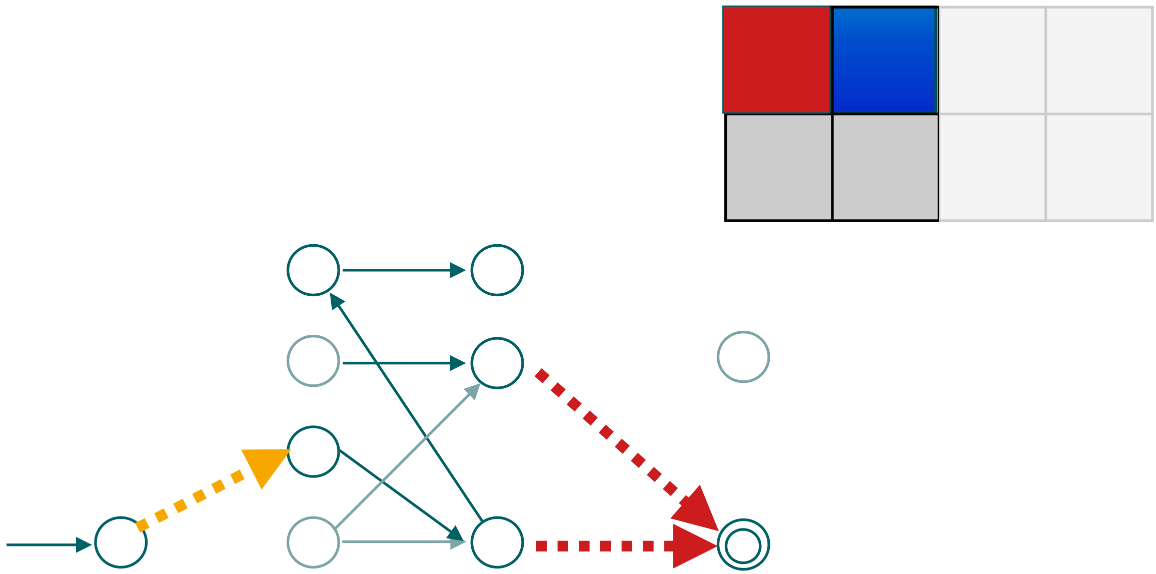


Joost-Pieter Katoen

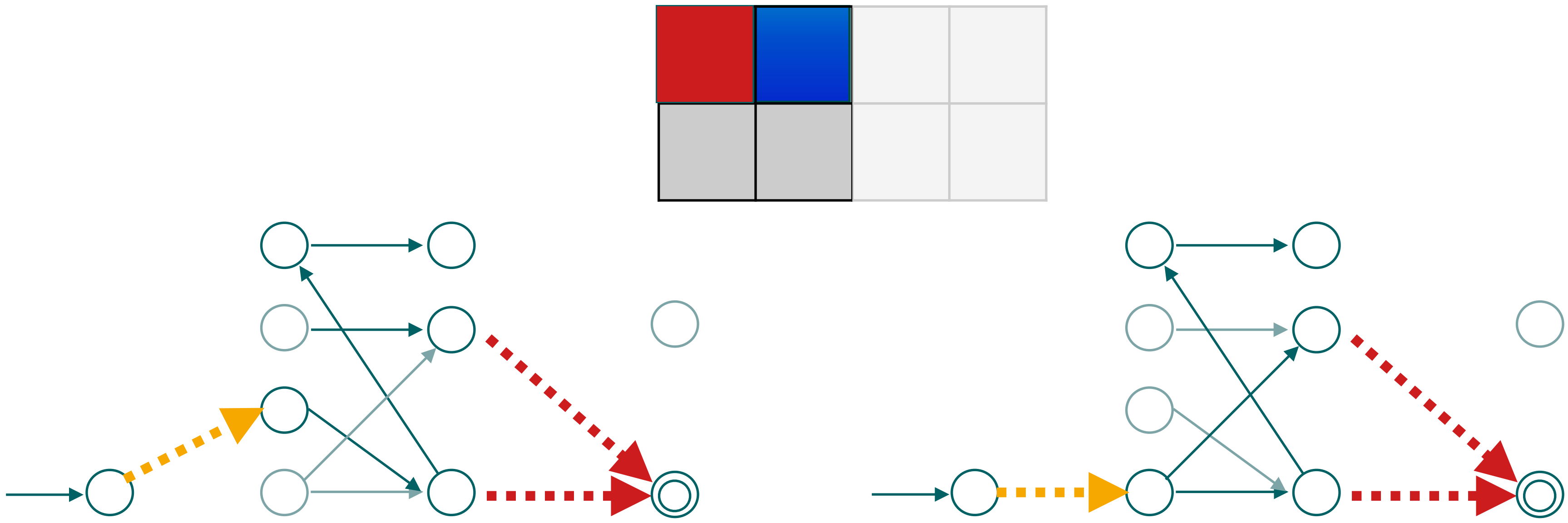
# The Baseline: Enumerate



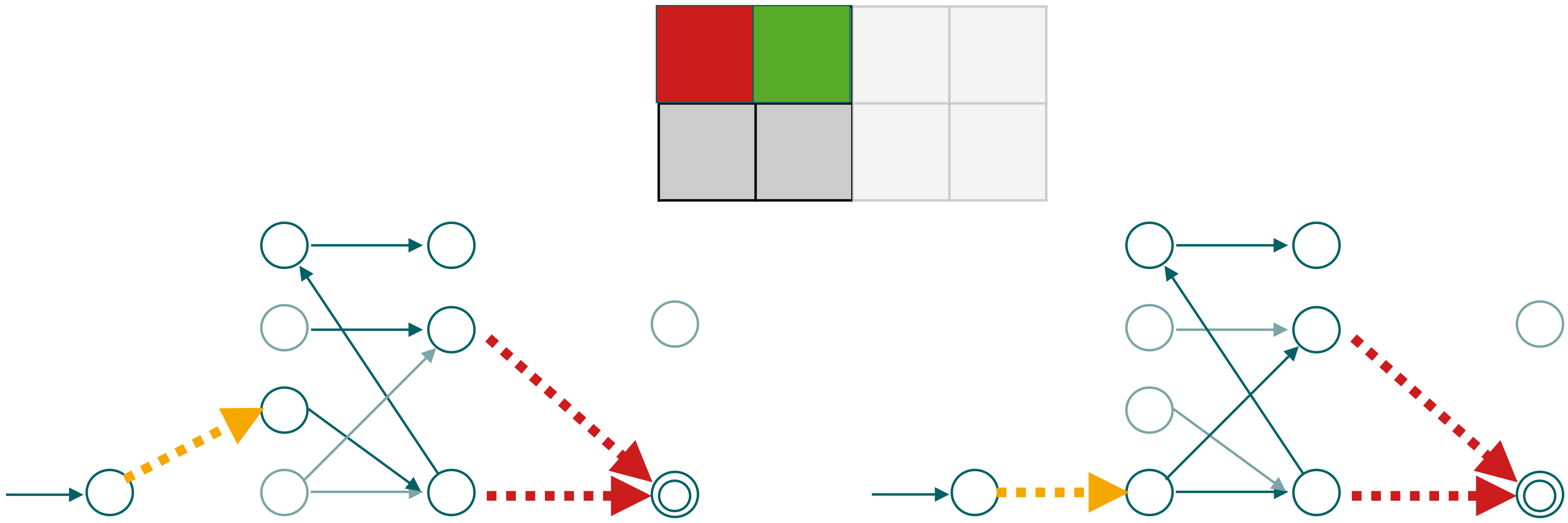
# The Baseline: Enumerate



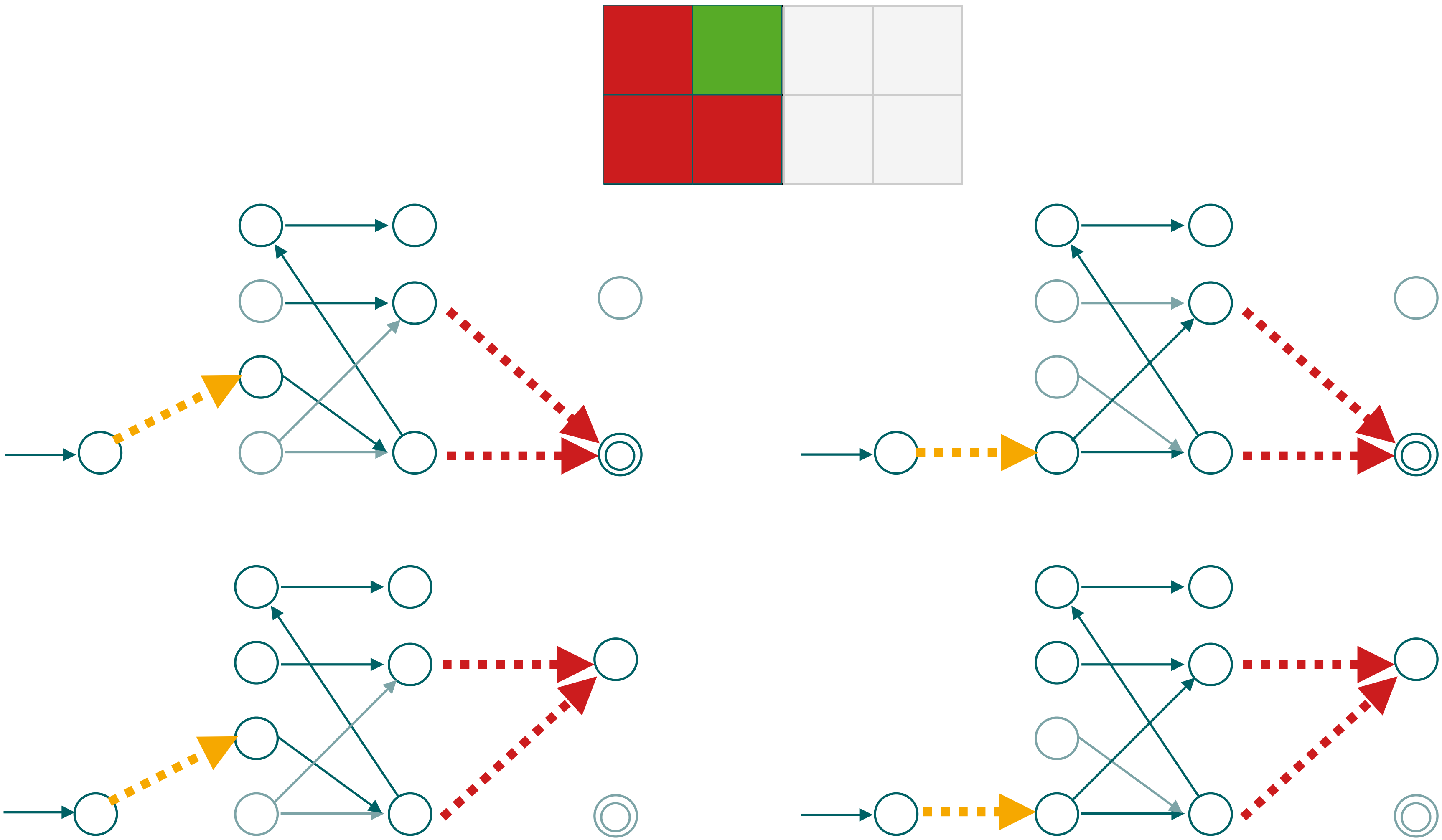
# The Baseline: Enumerate



# The Baseline: Enumerate

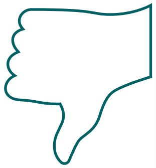


# The Baseline: Enumerate



# The Baseline: Enumerate

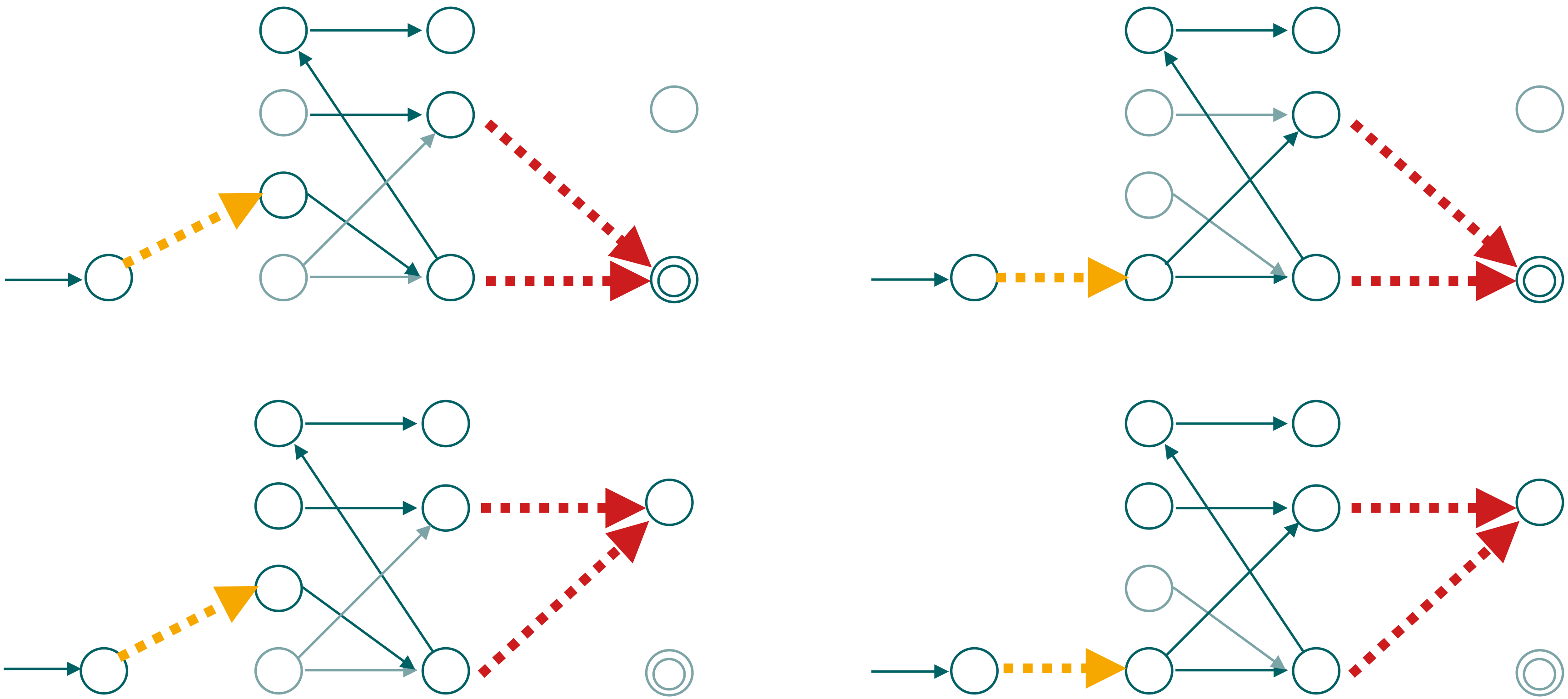
Simple



Does not use any of the structure from the family

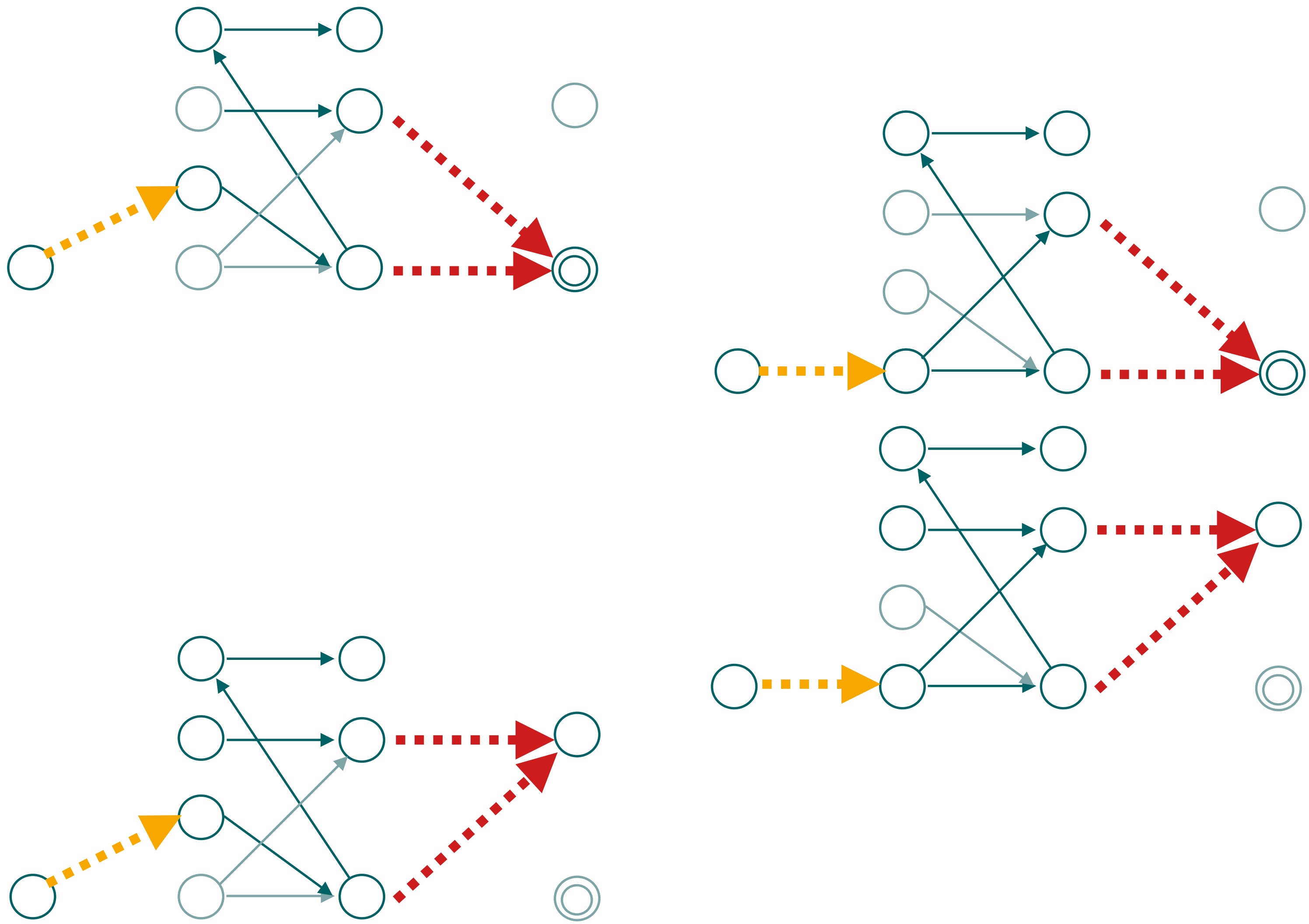
# The “All-in-One” Approach

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



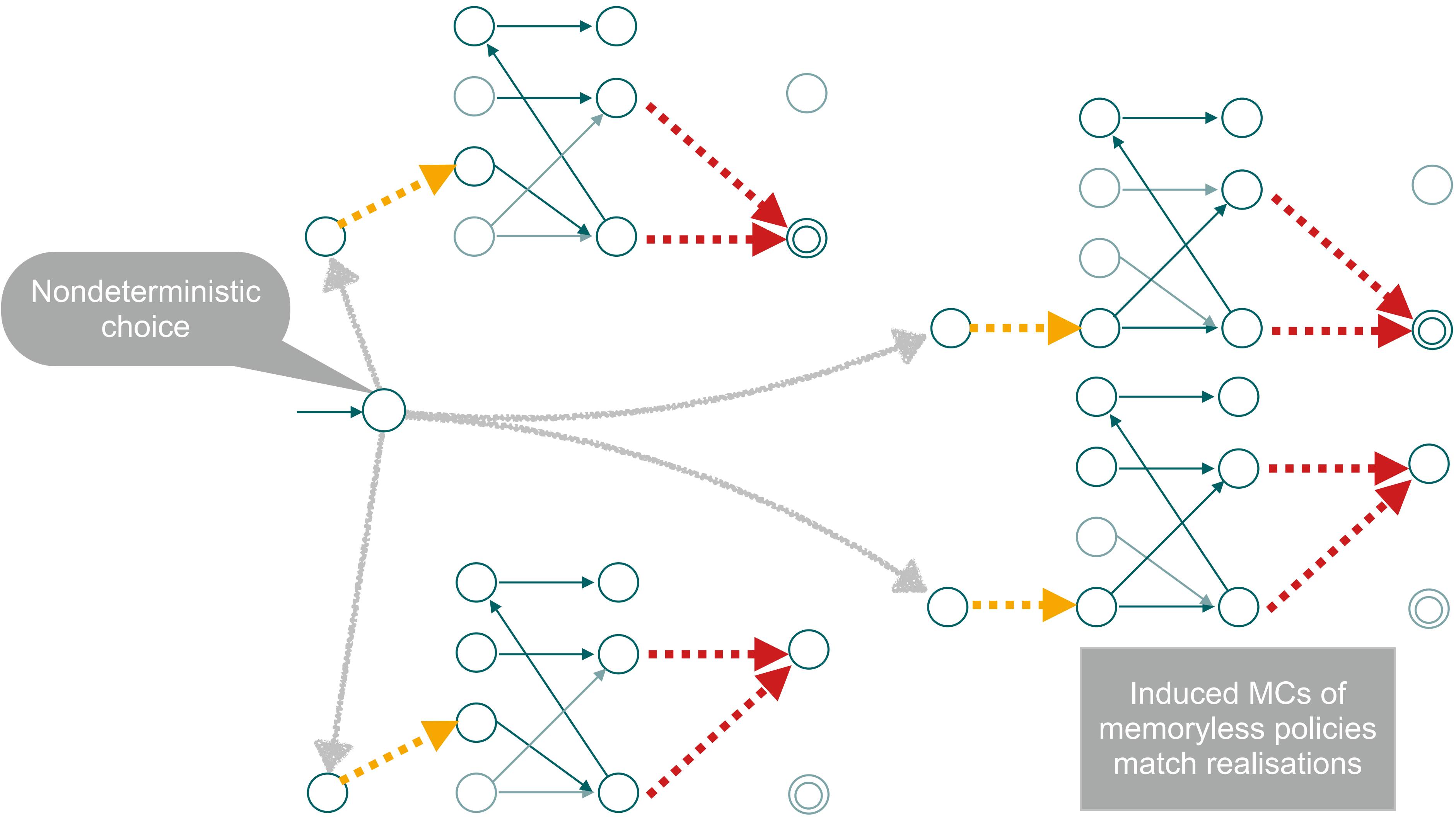
# The "All-in-One" Approach

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



# The “All-in-One” Approach

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



# The "All-in-One" Approach

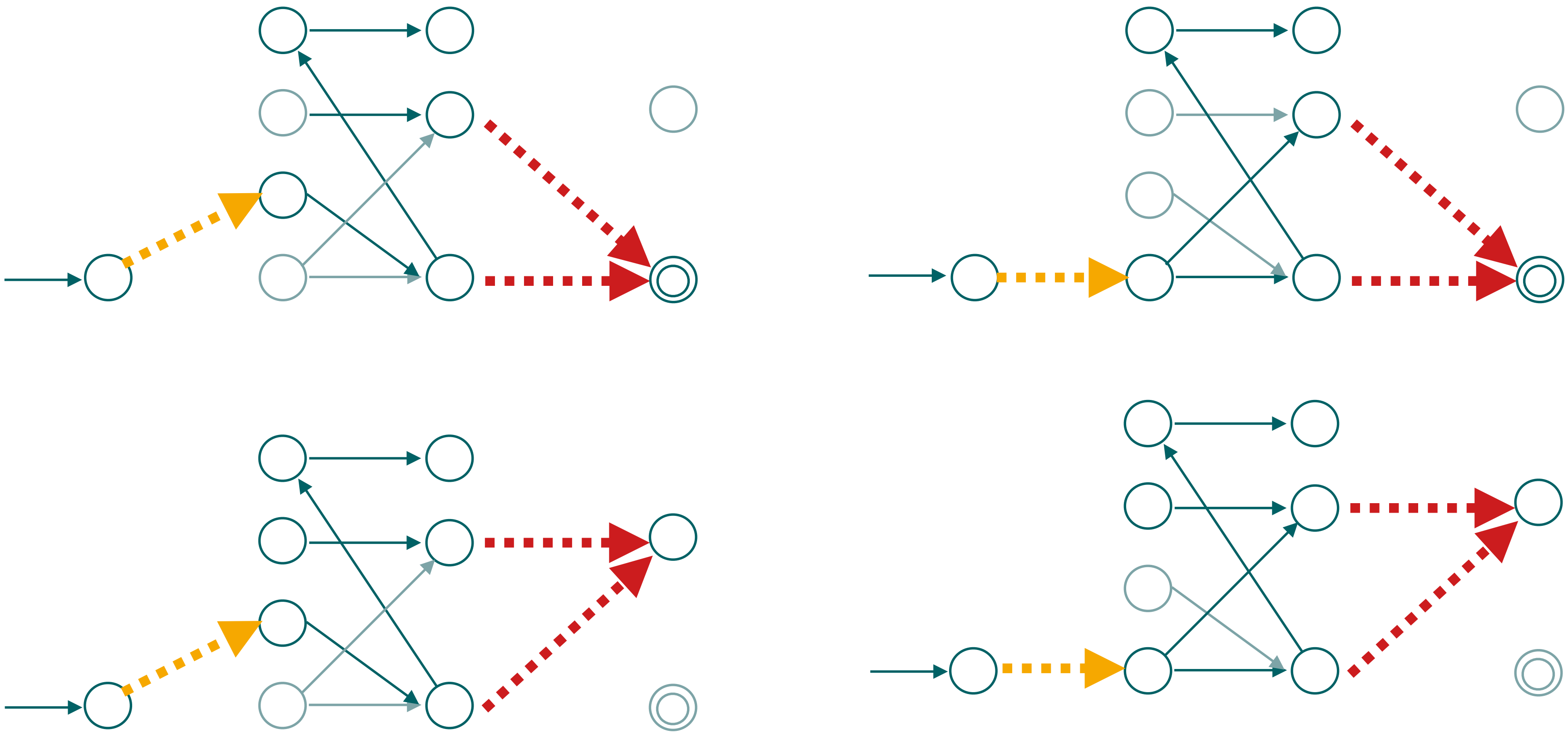
[Chrszon *et al*, Form Asp Comp 2018]

Build the union (linear blow-up), apply model checking and extract result

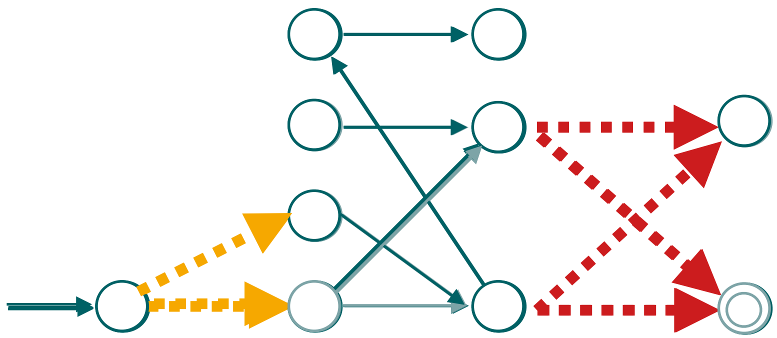
Exploit regularity
with symbolic methods

Probabilistic model checking
slow, already on moderate
family sizes

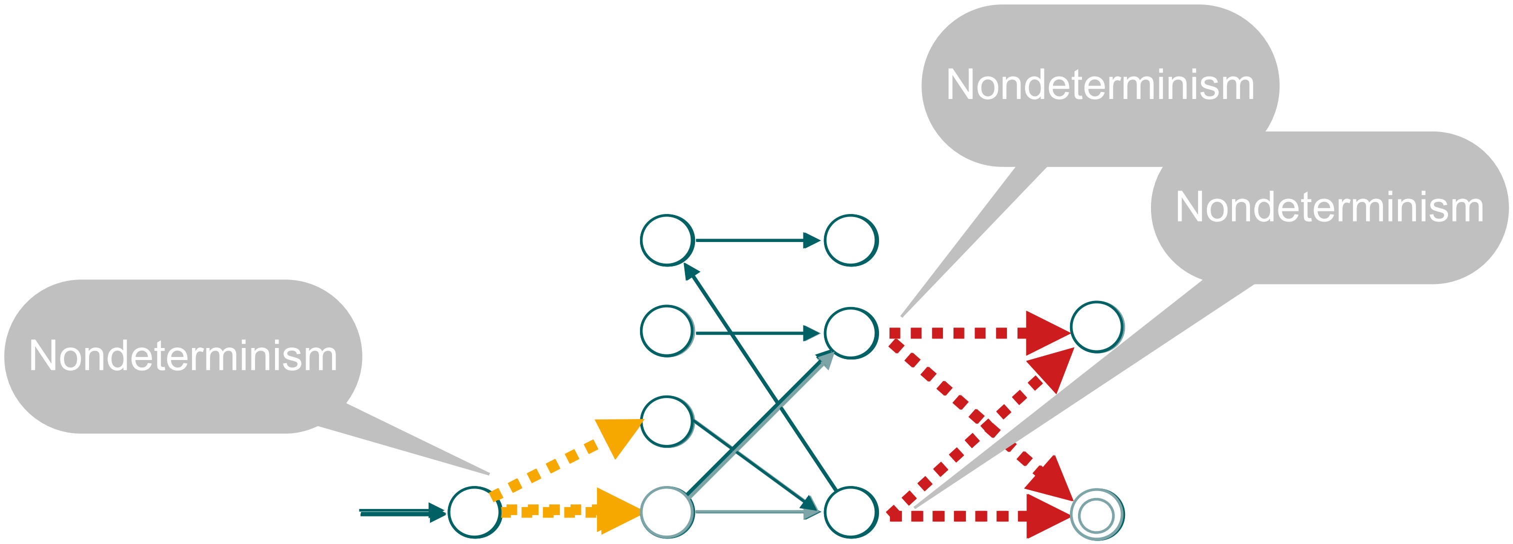
# Use Abstraction



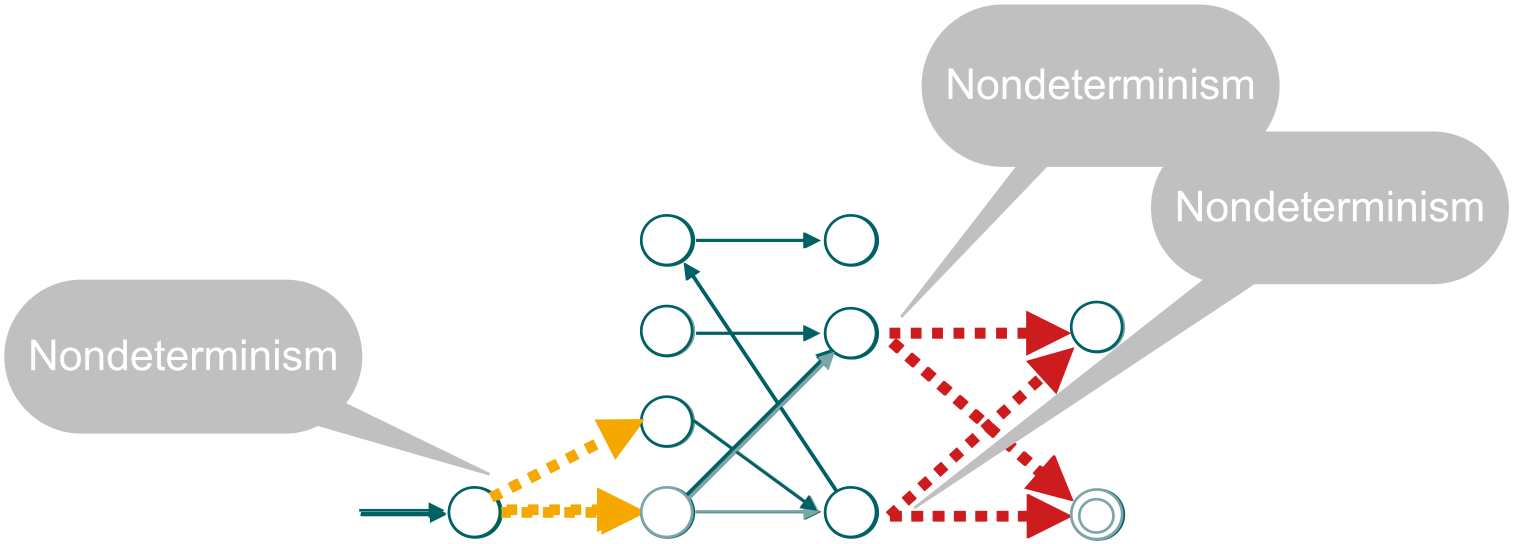
# “Forget” The Realisation We Are In



# “Forget” The Realisation We Are In



# “Forget” The Realisation We Are In



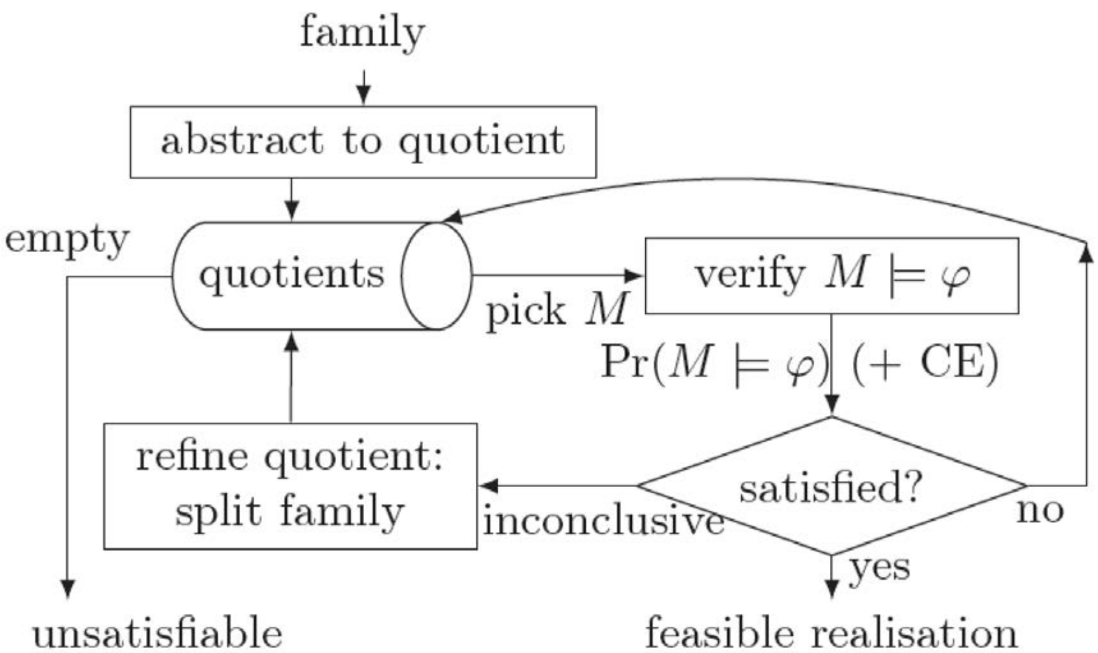

This yields a quotient MDP

# Abstraction Refinement: CEGAR

[Hermanns *et al*, CAV 2008] [Cordy *et al*, FSE 2014]

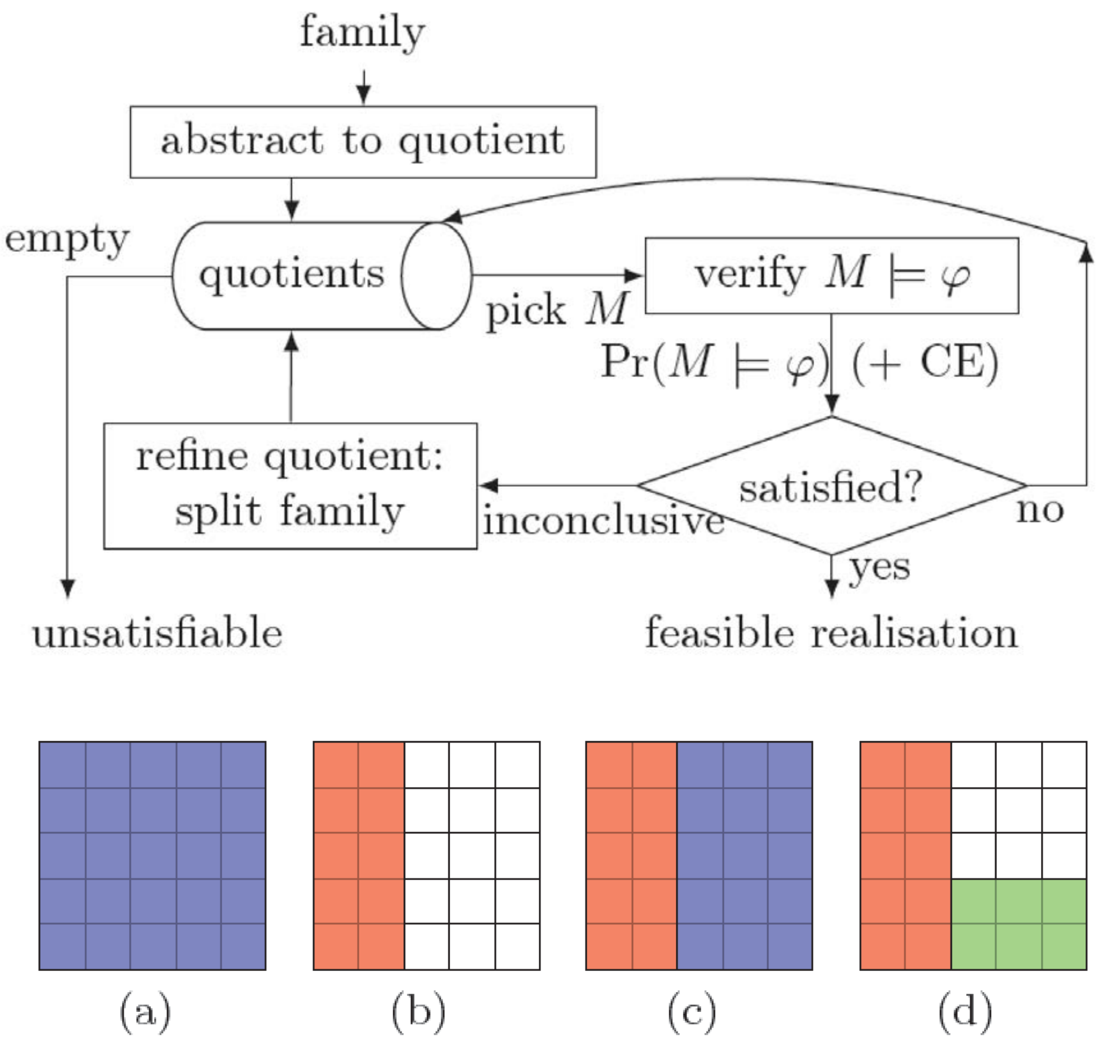
# Abstraction Refinement: CEGAR

[Hermanns *et al*, CAV 2008] [Cordy *et al*, FSE 2014]



# Abstraction Refinement: CEGAR

[Hermanns *et al*, CAV 2008] [Cordy *et al*, FSE 2014]



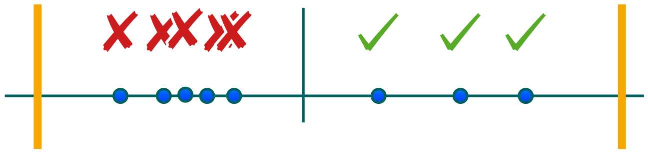
# Abstraction-Refinement (by splitting)

Full Situation

Algorithm's Perspective

# Abstraction-Refinement (by splitting)

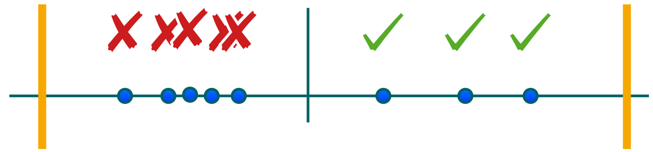
Full Situation



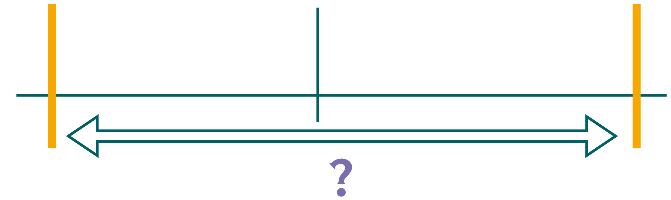
Algorithm's Perspective

# Abstraction-Refinement (by splitting)

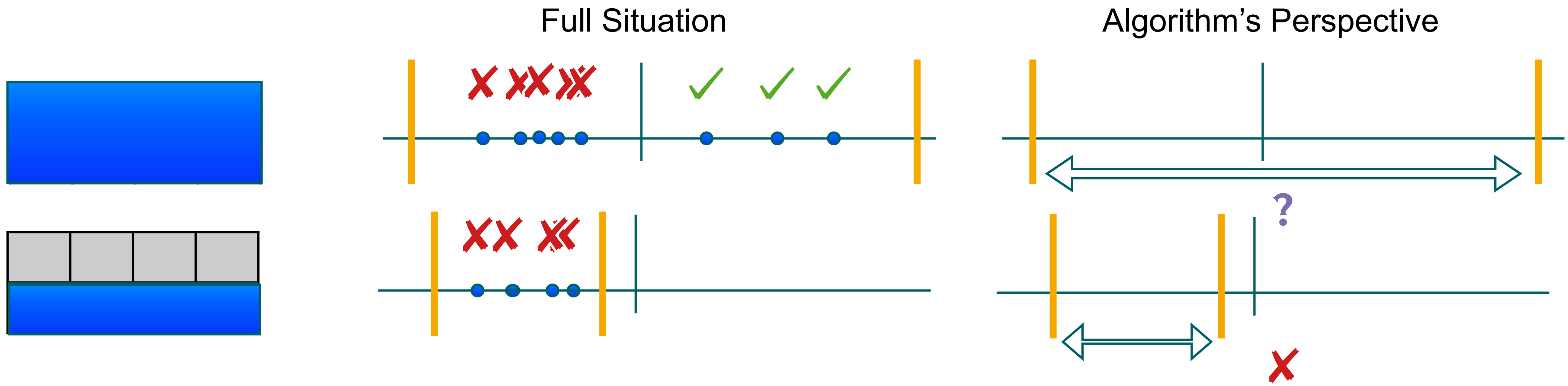
Full Situation



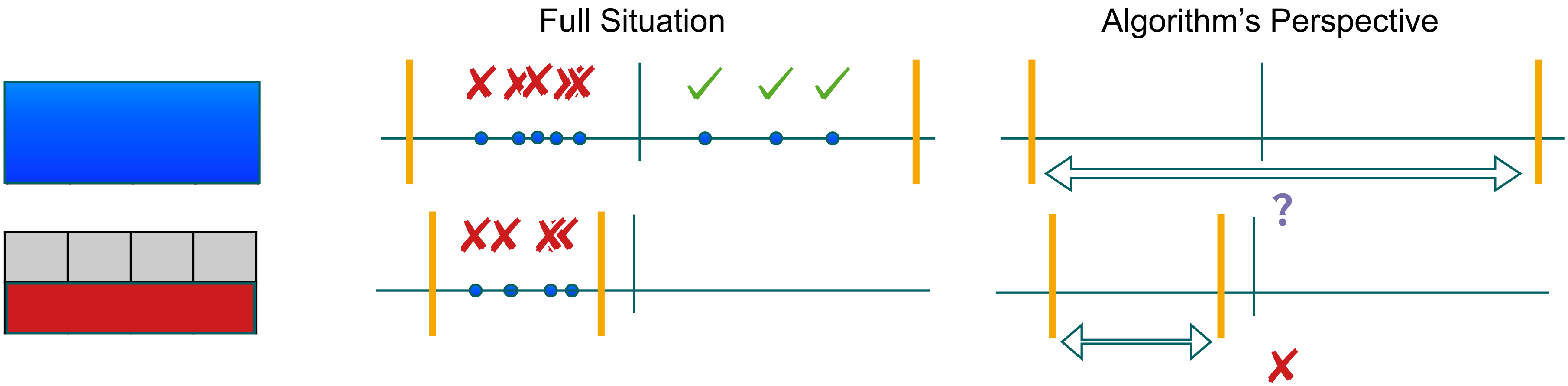
Algorithm’s Perspective



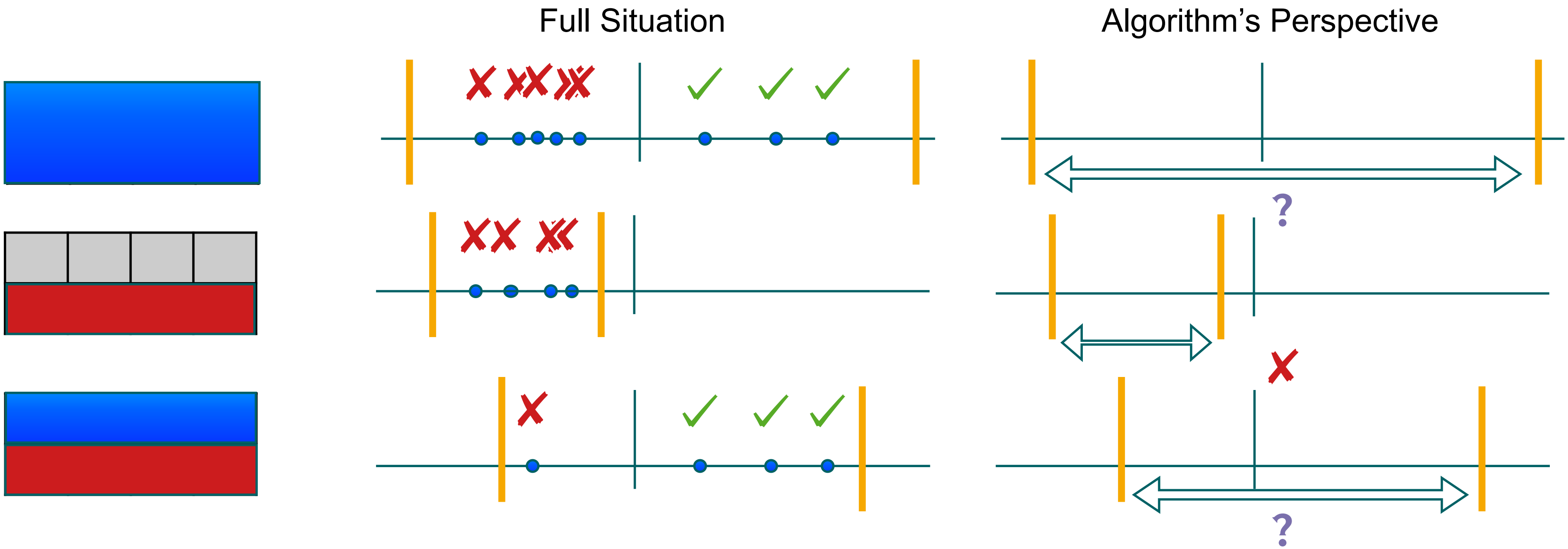
# Abstraction-Refinement (by splitting)



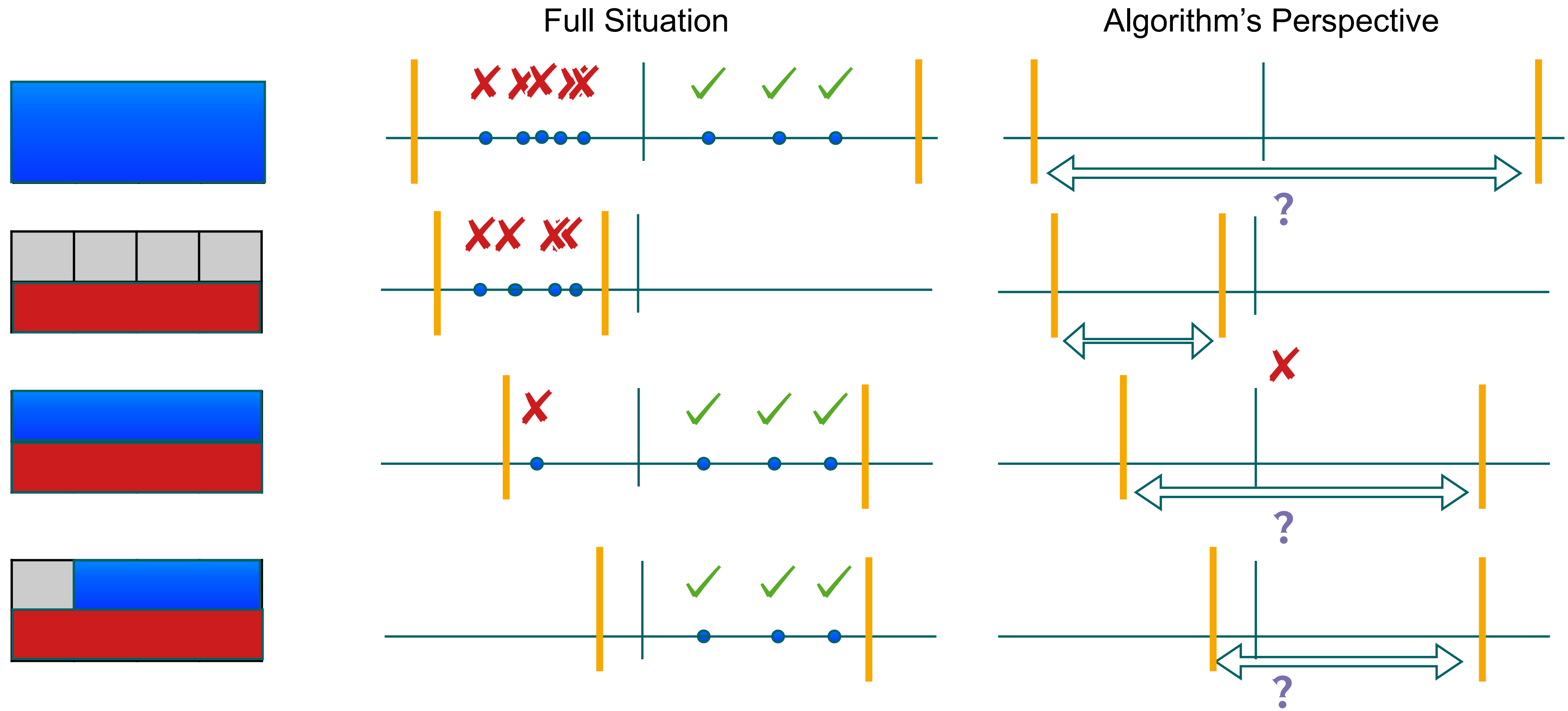
# Abstraction-Refinement (by splitting)



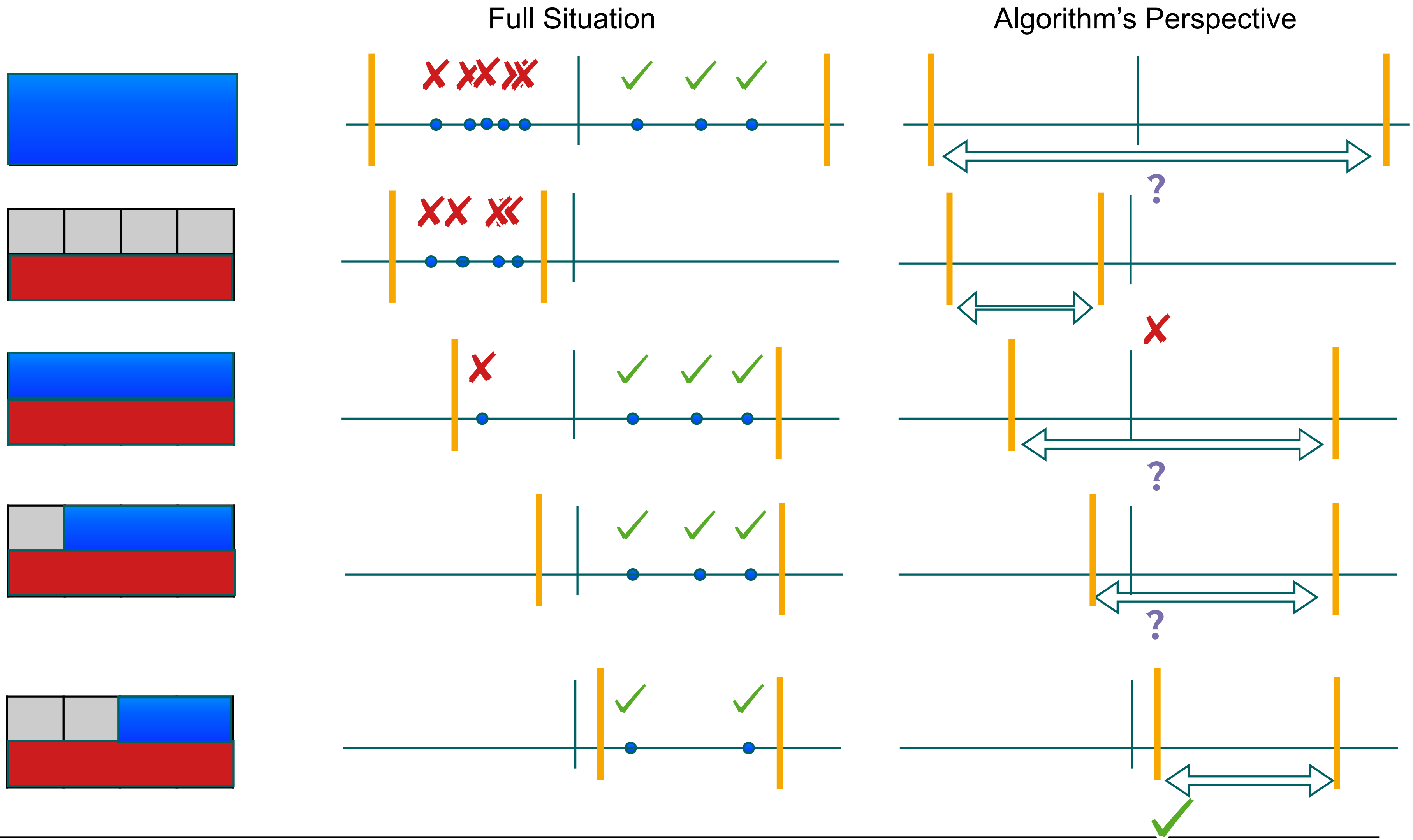
# Abstraction-Refinement (by splitting)



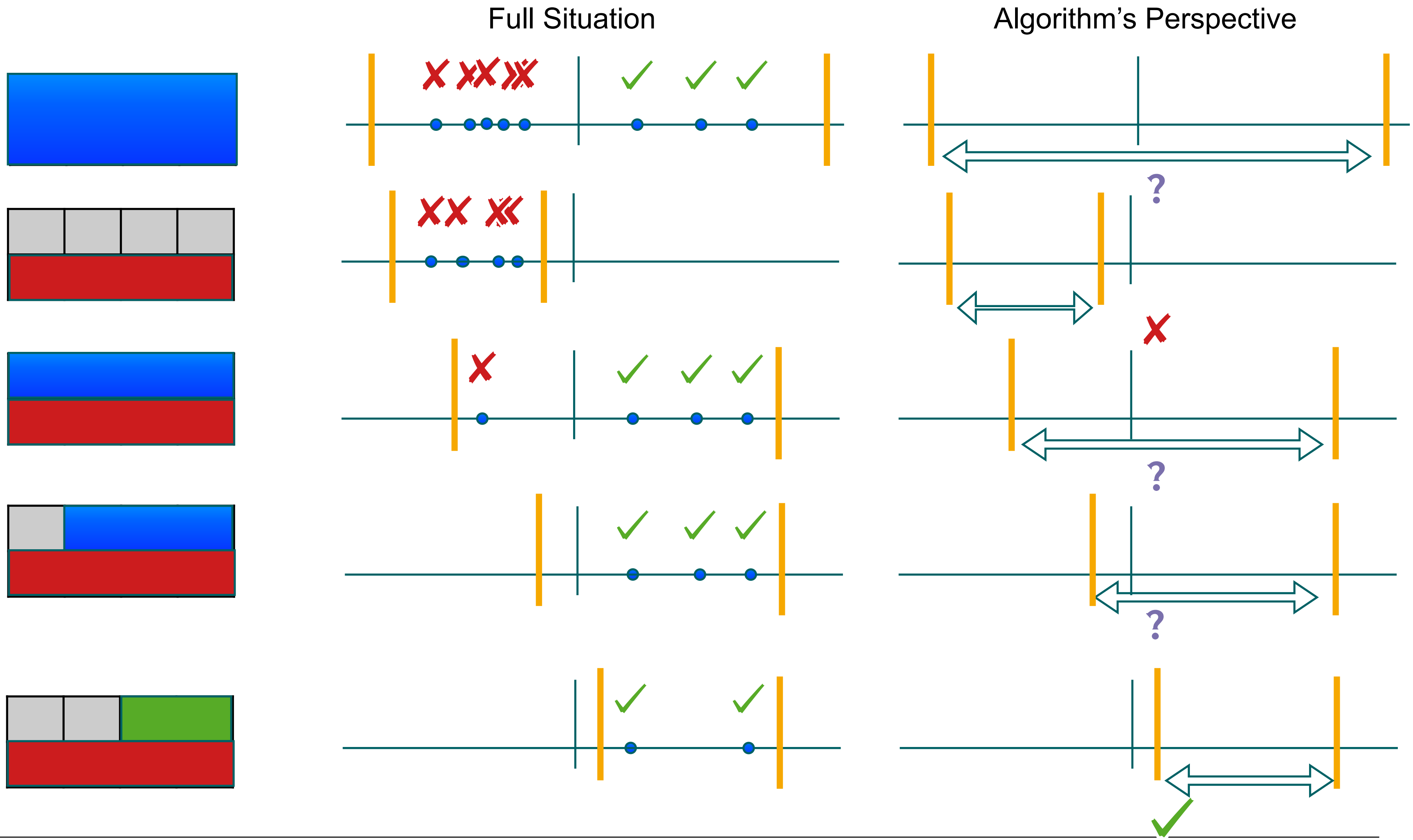
# Abstraction-Refinement (by splitting)



# Abstraction-Refinement (by splitting)



# Abstraction-Refinement (by splitting)



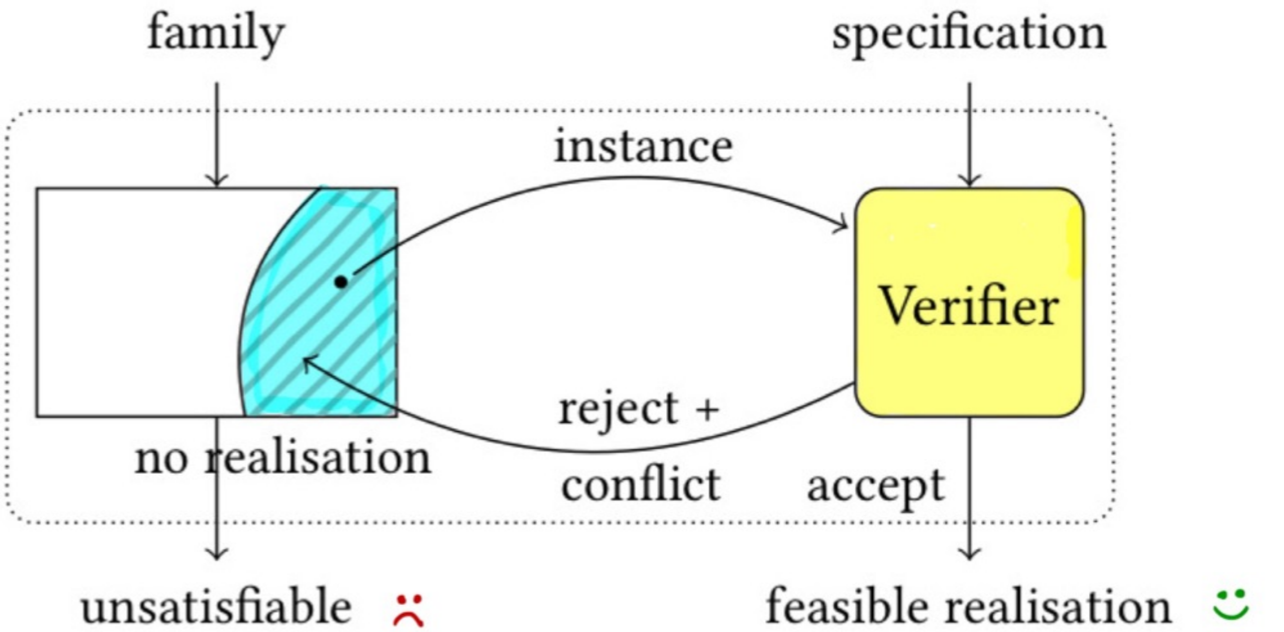
# CEGAR

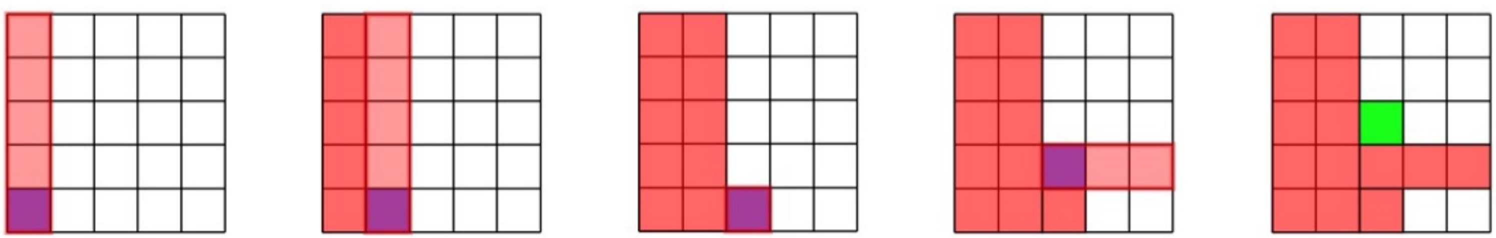
Huge potential,
limited runtime penalty

Quotient may be much larger
than any family member

# CEGIS

[Gulwani *et al*, 2017] [Solar-Lezama *et al*, PLDI 2005]



# Counterexamples

[Hensel *et al*, ATVA 2014] [Wimmer *et al*, LMCS 2015]

Counterexample := minimal command set of a PRISM program *P* violating *f*.

# Counterexamples

[Hensel *et al*, ATVA 2014] [Wimmer *et al*, LMCS 2015]

Counterexample := minimal command set of a PRISM program *P* violating *f*.

Finding a minimal command set is hard: MAXSAT or greedy approaches

# Counterexamples

[Hensel *et al*, ATVA 2014] [Wimmer *et al*, LMCS 2015]

Counterexample := minimal command set of a PRISM program *P* violating *f*.

Finding a minimal command set is hard: MAXSAT or greedy approaches

| model | instance | states | transitions | probabilities | | edges | |
|---|---|---|---|---|---|---|---|
| | | | | $\lambda$ | $\Pr^{+}(\Diamond T)$ | $\lvert E^{*}\rvert$ | $\lvert E\rvert$ |
| coin | (2, 2) | 272 | 492 | 0.30 | 0.56 | 8 | 14 |
| | (4, 4) | 43,136 | 144,352 | 0.30 | 0.54 | 17 | 28 |
| | (6, 2) | 1,258,240 | 6,236,736 | 0.30 | 0.59 | ≥ 16 | 42 |
| csma | (2, 4) | 7,958 | 10,594 | 0.50 | > 0.99 | 36 | 38 |
| | (2, 6) | 66,718 | 93,072 | 0.50 | > 0.99 | 36 | 42 |
| | (4, 2) | 761,962 | 1,327,068 | 0.40 | 0.78 | ≥ 43 | 72 |
| firewire | (3) | 4,093 | 5,585 | 0.50 | 1 | 24 | 64 |
| | (12) | 22,852 | 40,904 | 0.50 | 1 | 24 | 64 |
| | (36) | 212,268 | 481,792 | 0.50 | 1 | 24 | 64 |
| wlan | (2, 2) | 28,598 | 57,332 | 0.10 | 0.18 | 33 | 70 |
| | (4, 4) | 345,120 | 762,422 | 4e−4 | 7.9e−4 | 39 | 76 |
| | (6, 6) | 5,007,670 | 11,475,920 | 1e−7 | 2.2e−7 | 43 | 80 |

# Counterexamples

[Hensel *et al*, ATVA 2014] [Wimmer *et al*, LMCS 2015]

Counterexample := minimal command set of a PRISM program *P* violating *f*.

Finding a minimal command set is hard: MAXSAT or greedy approaches

| model | instance | states | transitions | probabilities | | edges | |
|---|---|---|---|---|---|---|---|
| | | | | $\lambda$ | $\Pr^+(\lozenge T)$ | $\lvert E^*\rvert$ | $\lvert E\rvert$ |
| coin | (2, 2) | 272 | 492 | 0.30 | 0.56 | 8 | 14 |
| | (4, 4) | 43,136 | 144,352 | 0.30 | 0.54 | 17 | 28 |
| | (6, 2) | 1,258,240 | 6,236,736 | 0.30 | 0.59 | ≥ 16 | 42 |
| csma | (2, 4) | 7,958 | 10,594 | 0.50 | > 0.99 | 36 | 38 |
| | (2, 6) | 66,718 | 93,072 | 0.50 | > 0.99 | 36 | 42 |
| | (4, 2) | 761,962 | 1,327,068 | 0.40 | 0.78 | ≥ 43 | 72 |
| firewire | (3) | 4,093 | 5,585 | 0.50 | 1 | 24 | 64 |
| | (12) | 22,852 | 40,904 | 0.50 | 1 | 24 | 64 |
| | (36) | 212,268 | 481,792 | 0.50 | 1 | 24 | 64 |
| wlan | (2, 2) | 28,598 | 57,332 | 0.10 | 0.18 | 33 | 70 |
| | (4, 4) | 345,120 | 762,422 | 4e–4 | 7.9e–4 | 39 | 76 |
| | (6, 6) | 5,007,670 | 11,475,920 | 1e–7 | 2.2e–7 | 43 | 80 |

# Counterexamples

[Hensel *et al*, ATVA 2014] [Wimmer *et al*, LMCS 2015]

Counterexample := minimal command set of a PRISM program *P* violating *f*.

Finding a minimal command set is hard: MAXSAT or greedy approaches

| model | instance | states | transitions | probabilities | | edges | |
|---|---|---|---|---|---|---|---|
| | | | | $\lambda$ | $\Pr^{+}(\lozenge T)$ | $\lvert E^{*}\rvert$ | $\lvert E\rvert$ |
| coin | (2, 2) | 272 | 492 | 0.30 | 0.56 | 8 | 14 |
| | (4, 4) | 43,136 | 144,352 | 0.30 | 0.54 | 17 | 28 |
| | (6, 2) | 1,258,240 | 6,236,736 | 0.30 | 0.59 | ≥ 16 | 42 |
| csma | (2, 4) | 7,958 | 10,594 | 0.50 | > 0.99 | 36 | 38 |
| | (2, 6) | 66,718 | 93,072 | 0.50 | > 0.99 | 36 | 42 |
| | (4, 2) | 761,962 | 1,327,068 | 0.40 | 0.78 | ≥ 43 | 72 |
| firewire | (3) | 4,093 | 5,585 | 0.50 | 1 | 24 | 64 |
| | (12) | 22,852 | 40,904 | 0.50 | 1 | 24 | 64 |
| | (36) | 212,268 | 481,792 | 0.50 | 1 | 24 | 64 |
| wlan | (2, 2) | 28,598 | 57,332 | 0.10 | 0.18 | 33 | 70 |
| | (4, 4) | 345,120 | 762,422 | 4e−4 | 7.9e−4 | 39 | 76 |
| | (6, 6) | 5,007,670 | 11,475,920 | 1e−7 | 2.2e−7 | 43 | 80 |

# Counterexamples

[Hensel *et al*, ATVA 2014] [Wimmer *et al*, LMCS 2015]

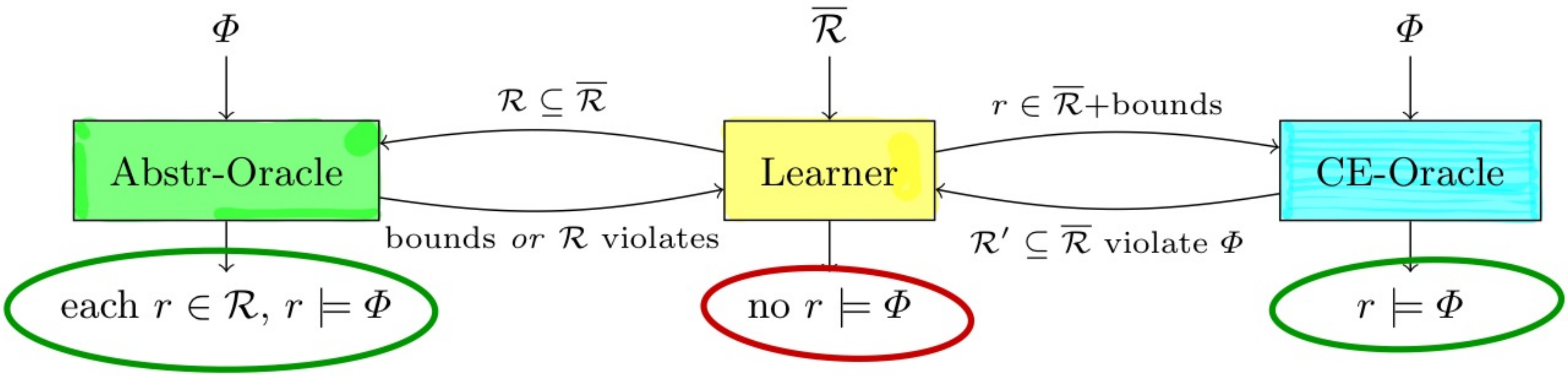
Counterexample := minimal command set of a PRISM program *P* violating *f*.

Finding a minimal command set is hard: MAXSAT or greedy approaches

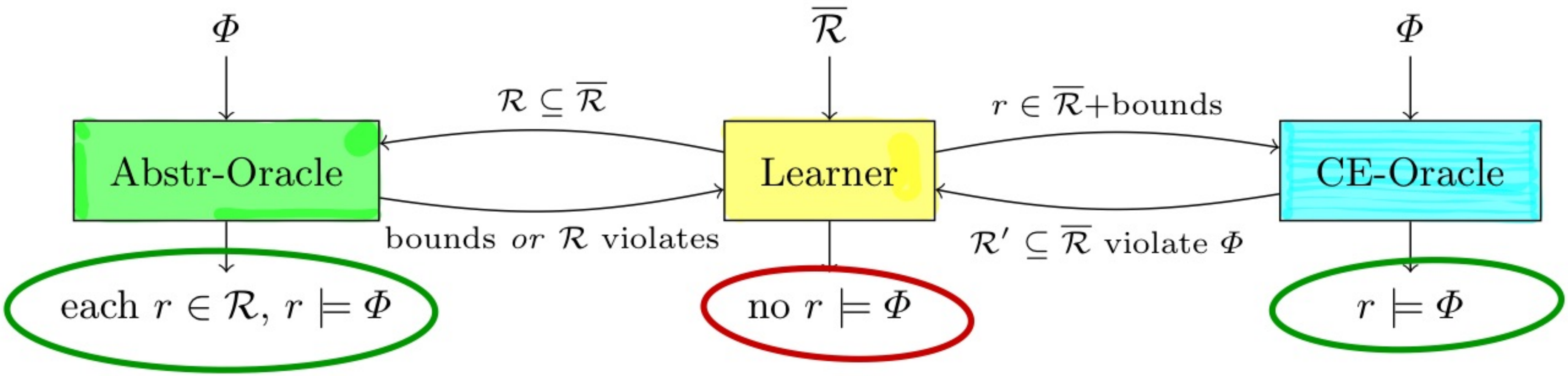
| model | instance | states | transitions | probabilities $\lambda$ | probabilities $\Pr^+(\lozenge T)$ | edges $\lvert E^*\rvert$ | edges $\lvert E\rvert$ |
|---|---|---|---|---|---|---|---|
| coin | (2, 2) | 272 | 492 | 0.30 | 0.56 | 8 | 14 |
| | (4, 4) | 43,136 | 144,352 | 0.30 | 0.54 | 17 | 28 |
| | (6, 2) | 1,258,240 | 6,236,736 | 0.30 | 0.59 | ≥ 16 | 42 |
| csma | (2, 4) | 7,958 | 10,594 | 0.50 | > 0.99 | 36 | 38 |
| | (2, 6) | 66,718 | 93,072 | 0.50 | > 0.99 | 36 | 42 |
| | (4, 2) | 761,962 | 1,327,068 | 0.40 | 0.78 | ≥ 43 | 72 |
| firewire | (3) | 4,093 | 5,585 | 0.50 | 1 | 24 | 64 |
| | (12) | 22,852 | 40,904 | 0.50 | 1 | 24 | 64 |
| | (36) | 212,268 | 481,792 | 0.50 | 1 | 24 | 64 |
| wlan | (2, 2) | 28,598 | 57,332 | 0.10 | 0.18 | 33 | 70 |
| | (4, 4) | 345,120 | 762,422 | 4e−4 | 7.9e−4 | 39 | 76 |
| | (6, 6) | 5,007,670 | 11,475,920 | 1e−7 | 2.2e−7 | 43 | 80 |

Property: If a sub-MC of MC *D* refutes the safety property *f*, then *D* refutes *f* too.

# The Best of Both Worlds



## The Best of Both Worlds




Implemented in Python, using Z3 and the probabilistic model checker Storm

# Experimental Evaluation

- Implementation on top of Python API of Storm
- Takes PRISM or JANI file with open integer constants

# Experimental Evaluation

- Implementation on top of Python API of Storm
- Takes PRISM or JANI file with open integer constants

stormchecker.org

**CAV 2017**

# Experimental Evaluation

- Implementation on top of Python API of Storm
- Takes PRISM or JANI file with open integer constants

stormchecker.org

**CAV 2017**

**Why Storm?**

# Experimental Evaluation

- Implementation on top of Python API of Storm
- Takes PRISM or JANI file with open integer constants

stormchecker.org

**CAV 2017**

**Why Storm?**

The 2019 Comparison of Tools
for the Analysis of Quantitative
Formal Models
(QComp 2019 Competition Report)

**TACAS 2019**
**ISOLA 2020**

# Experimental Evaluation

- Implementation on top of Python API of Storm
- Takes PRISM or JANI file with open integer constants

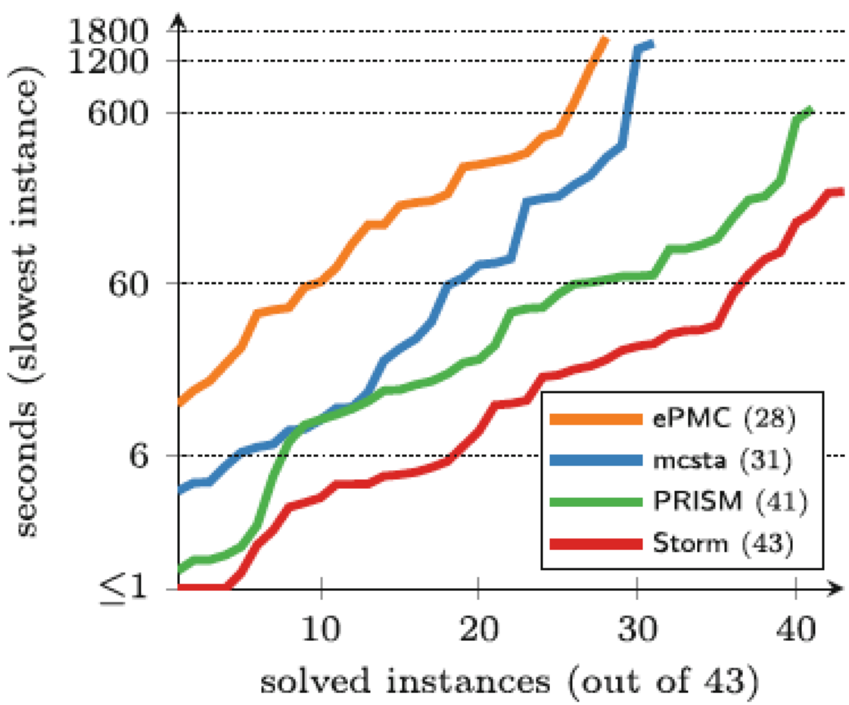
stormchecker.org

**CAV 2017**

**Why Storm?**

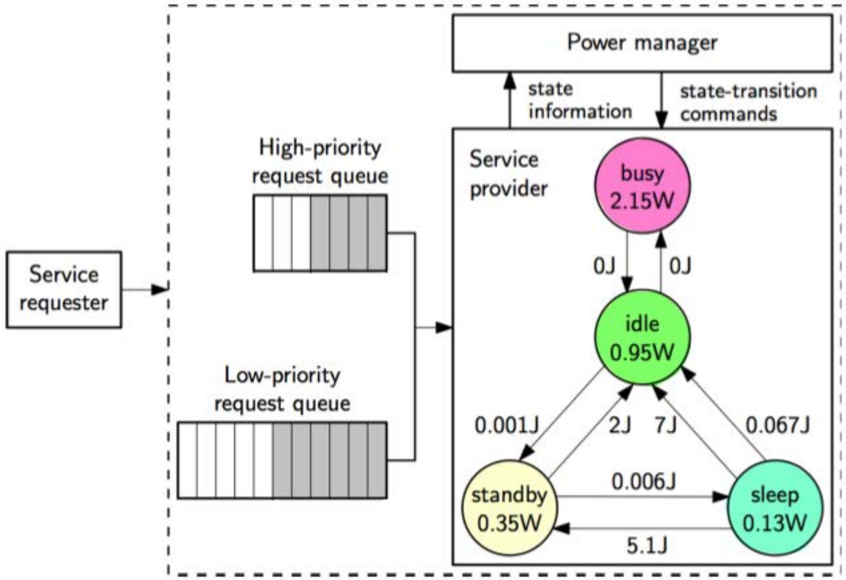
The 2019 Comparison of Tools for the Analysis of Quantitative Formal Models
(QComp 2019 Competition Report)

**TACAS 2019**
**ISOLA 2020**



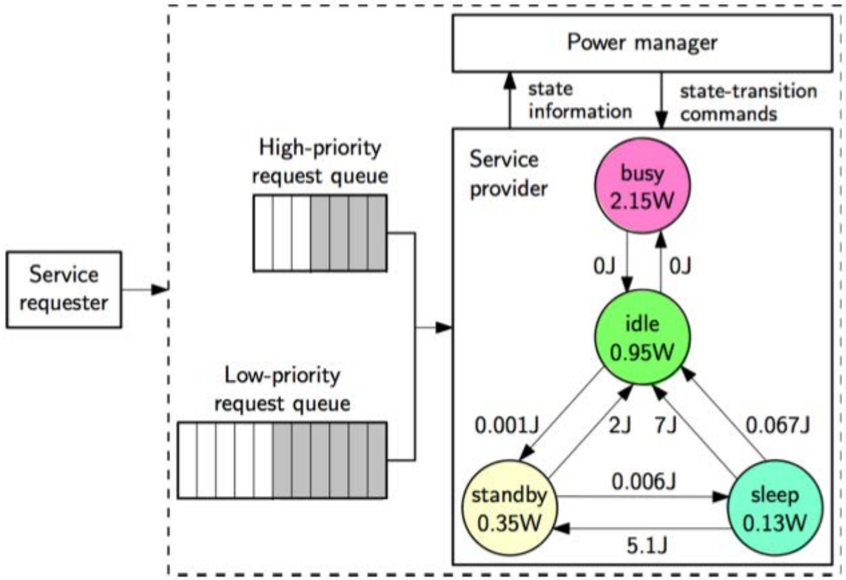
# Sketching Probabilistic Programs

[Benini *et al*, IEEE CAD 1999]



# Sketching Probabilistic Programs

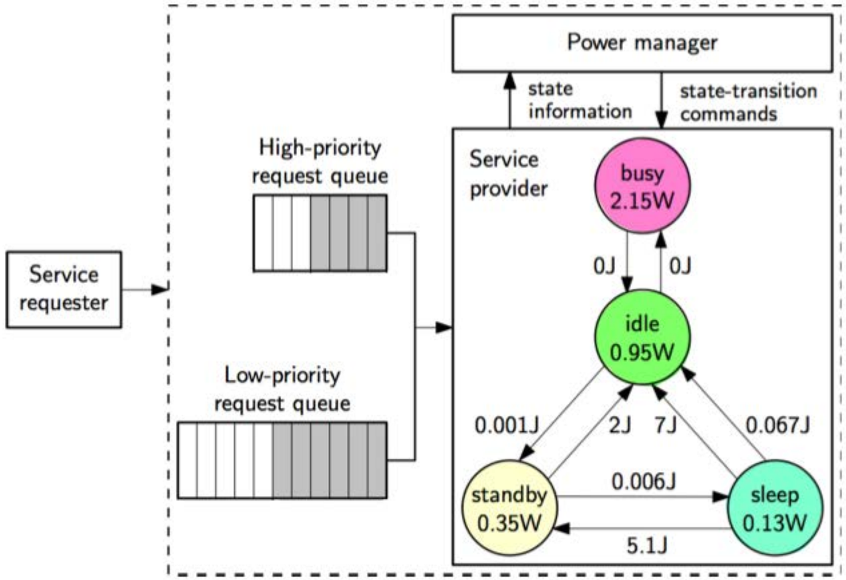
[Benini *et al*, IEEE CAD 1999]




```
dtmc

const int H;
const int K;

module example
   s : [0..11] init 0;
   [] s=0 -> 1:   (s'=H);
   [] s=1 -> 0.5:(s'=7) + 0.5:(s'=8);
   //...
   [] s=7 -> 0.8:(s'=K) + 0.2:(s'=2);
   [] s=8 -> 1:   (s'=K);
   //...
endmodule
```

# Sketching Probabilistic Programs

[Benini *et al*, IEEE CAD 1999]



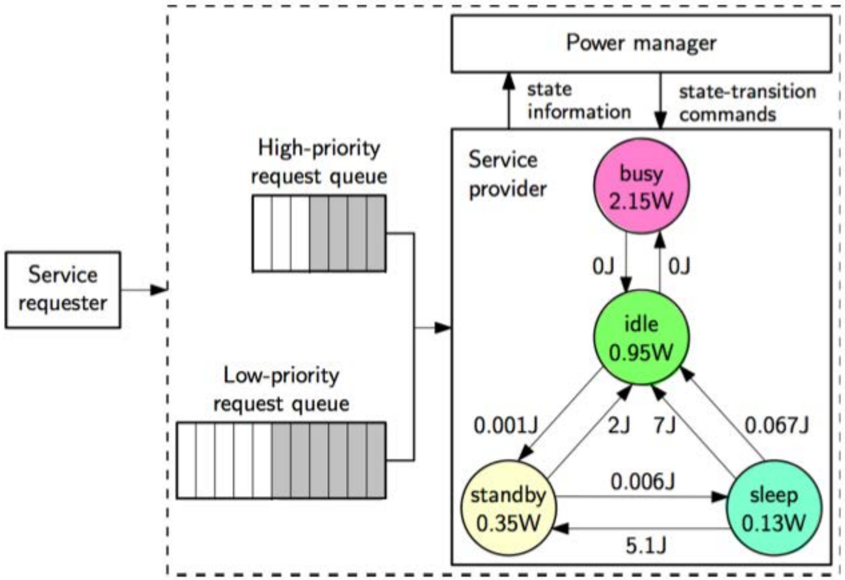

```
dtmc

const int H;
const int K;

module example
   s : [0..11] init 0;
   [] s=0 -> 1:   (s'=H);
   [] s=1 -> 0.5:(s'=7) + 0.5:(s'=8);
   //...
   [] s=7 -> 0.8:(s'=K) + 0.2:(s'=2);
   [] s=8 -> 1:   (s'=K);
   //...
endmodule
```

### Challenge

- Synthesise guards and updates in DPM control program with 9 holes
- Specification = conjunction of expected #lost reqs and energy consumption

# Sketching Probabilistic Programs

[Benini *et al*, IEEE CAD 1999]




```
dtmc

const int H;
const int K;

module example
   s : [0..11] init 0;
   [] s=0 -> 1:   (s'=H);
   [] s=1 -> 0.5:(s'=7) + 0.5:(s'=8);
   //...
   [] s=7 -> 0.8:(s'=K) + 0.2:(s'=2);
   [] s=8 -> 1:   (s'=K);
   //...
endmodule
```

**Challenge**

- Synthesise guards and updates in DPM control program with 9 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

[Mealeau *et al*, UAI 1999]

# POMDP Strategy Synthesis is Hard

# POMDP Strategy Synthesis is Hard

Determining an optimal strategy for reachability is undecidable

# POMDP Strategy Synthesis is Hard

Determining an optimal strategy for reachability is undecidable

Determining an optimal *positional* strategy for reachability
is ETR-complete

# POMDP Strategy Synthesis is Hard

Determining an optimal strategy for reachability is undecidable

Determining an optimal *positional* strategy for reachability is ETR-complete

This is as hard as finding the real roots of a polynomial

# POMDP Strategy Synthesis is Hard

Determining an optimal strategy for reachability is undecidable

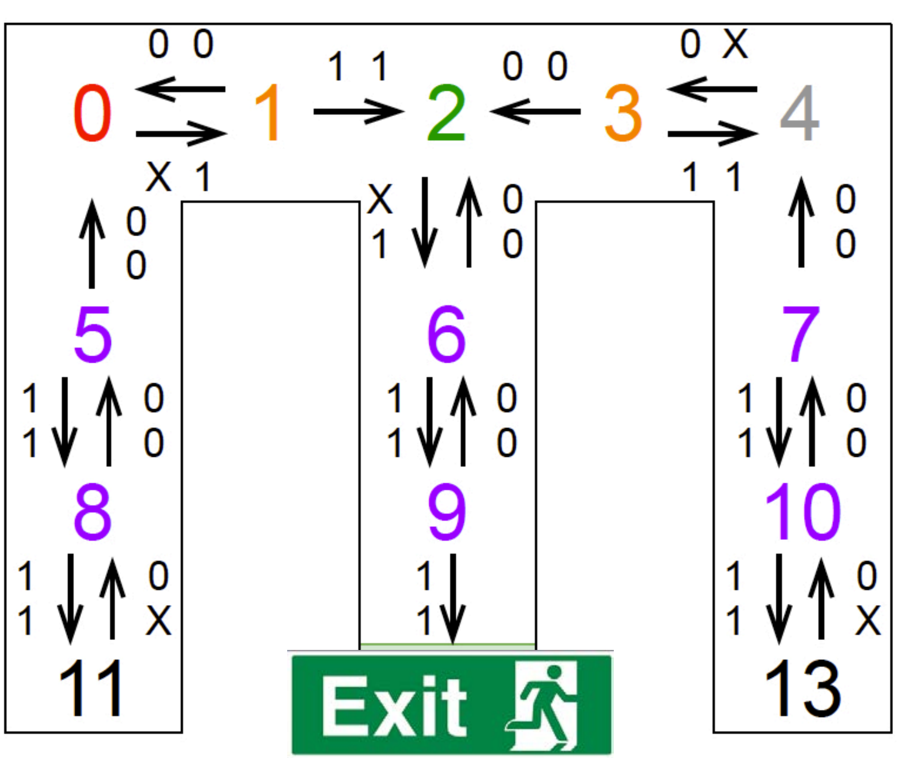
Determining an optimal *positional* strategy for reachability is ETR-complete

This is as hard as finding the real roots of a polynomial

Practice: determine randomised finite-state controllers with bounded memory

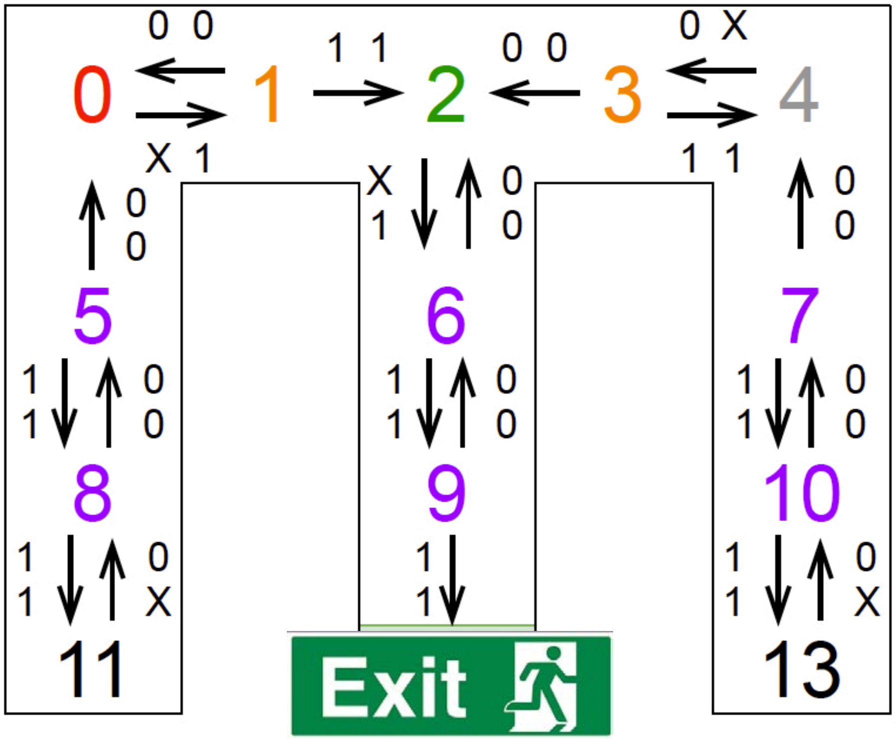
# Maze POMDP

## Maze POMDP




Minimise the expected #steps to exit the maze

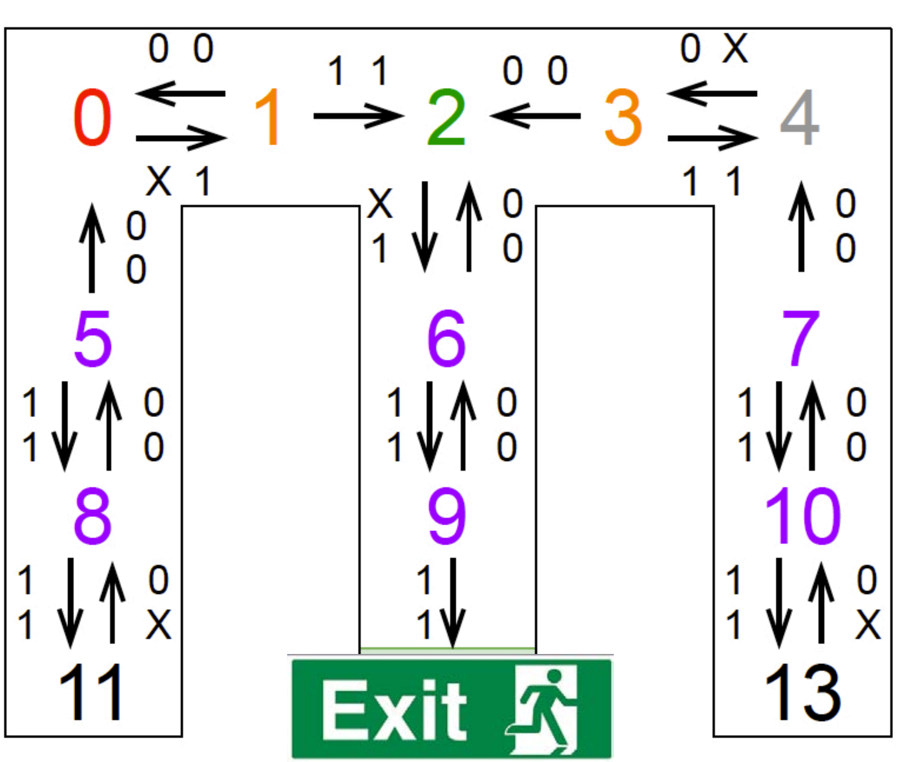
# Maze POMDP




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

Minimise the expected #steps to exit the maze

# 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

# Improving Herman's Randomised Self-Stabilisation

# 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 $x_i := x_{i-1}$

# 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

# 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

# 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

# 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

# On Automated Synthesis of Probabilistic Models

## On Automated Synthesis of Probabilistic Models

- **Applications**:
  - ✓ Program sketching
  - ✓ Controller synthesis in partially observable systems
  - ✓ Software product lines
  - ✓ Randomised distributed computing

- **Approaches:** CEGAR, CEGIS and their combination

- **Further work:**
  - ✓ Other refinement strategies
  - ✓ Other models: MDPs, POMDPs, ……
  - ✓ Infinite families, infinite-state realisations, ……

- **Further details:**

  LNCS 10500 (Festschrift Scott Smolka), TACAS 2019+21, FM 2019

# PAYNT (Probabilistic progrAm sYNThesizer)