02246 - Model Checking

$$M \models \Phi ?$$

Lecture 8 - Model Checking PCTL

Technical University of Denmark

System

refine/implement

Requirements

abstract model

$M \models \Phi$?

some logic here

Finite-state transition System

some smart algorithm here

yes :)

may be :|

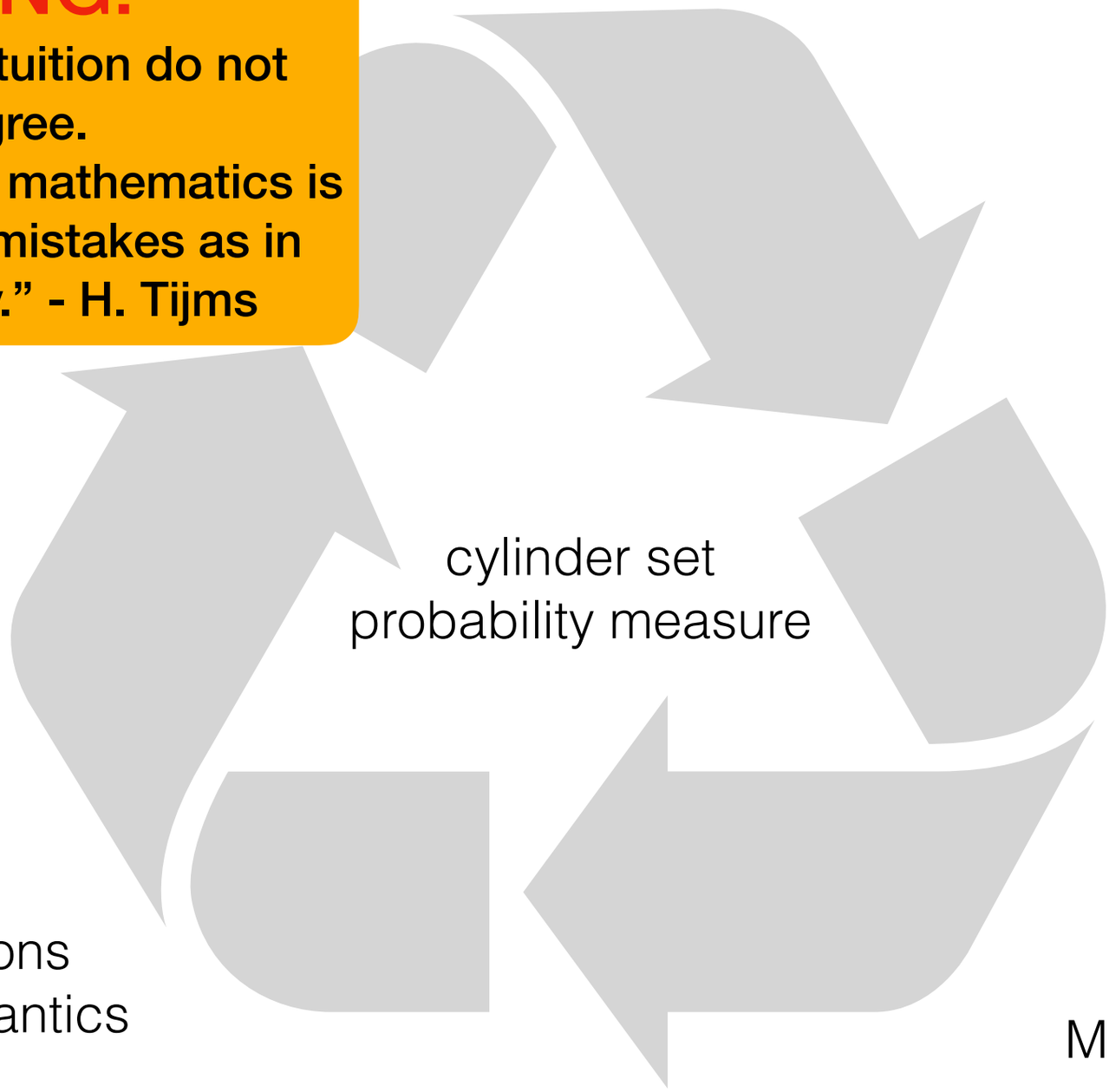
no :(

How to check probabilistic properties?

Intuitions / analytical expression based on intuitions

WARNING!

“Probability and intuition do not always agree.
In no other branch of mathematics is it so easy to make mistakes as in probability theory.” - H. Tijms



cylinder set
probability measure

analytical expressions
based on formal semantics

MC algorithms

Key points of this lecture

Model checking PCTL is done bottom-up (as for CTL)

Model checking PCTL requires to compute probabilities of bounded and unbounded reachability.

Recursive algorithms for bounded reachability (similar to transient distribution).

Algorithms for unbounded reachability based on solving systems of equations (similar to steady states).

Examples

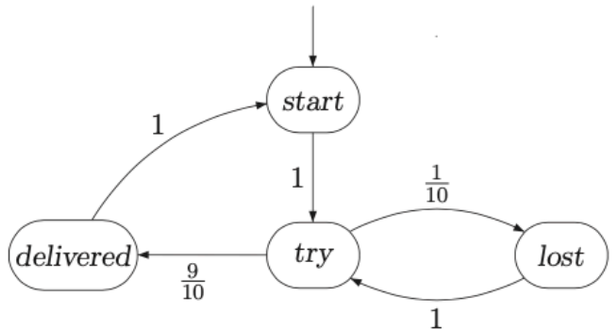


Figure 10.1: Markov chain for a simple communication protocol.

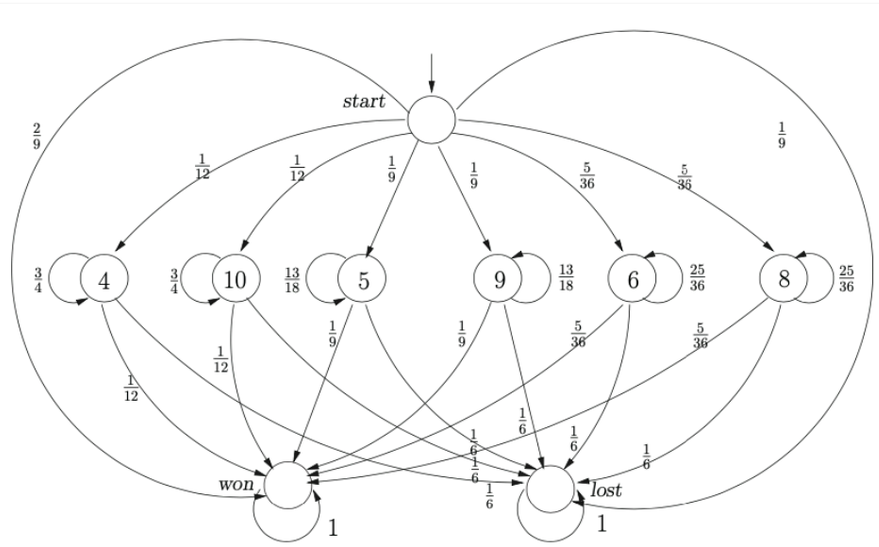
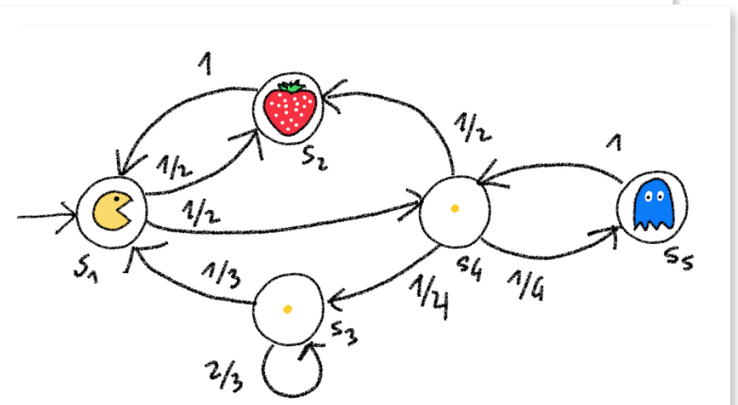


Figure 10.3: Markov chain for the behavior of the craps game.



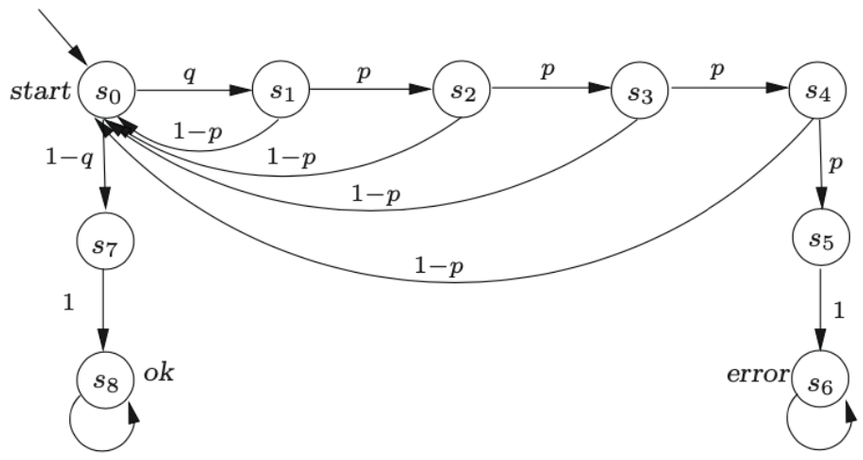
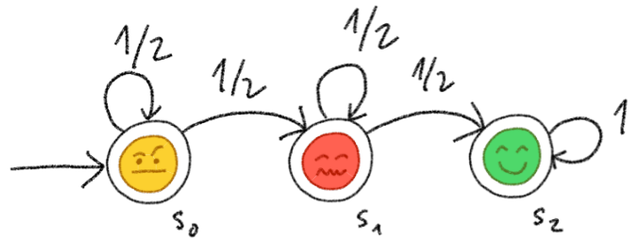


Figure 10.4: Markov chain of the IPv4 zeroconf protocol (for n=4 probes).



Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\circ B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

PCTL Syntax

The syntax for PCTL state formulas is

$$\phi ::= true \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \mathbb{P}_J(\psi)$$

where p is an atomic proposition and J is some interval in [0,1]

$\mathbb{P}_J(\psi)$ reads “the probability to satisfy Ψ is in the interval J ”

The syntax for PCTL path formulas is

$$\psi ::= \bigcirc\phi \mid \phi_1 \mathsf{U} \phi_2 \mid \phi_1 \mathsf{U}^{\leq n} \phi_2$$

where n is some natural number

CTL - formal semantics

We define the formal semantics of PCTL as 2 relations.

A relation between states and state formulas

$s \models true$ (holds always)

$s \models p$ iff iff $p \in L(s)$

$s \models \neg\phi$ iff $s \not\models \phi$

$s \models \phi_1 \wedge \phi_2$ iff $s \models \phi_1$ and $s \models \phi_2$

$s \models \mathbb{P}_J(\psi)$ iff $Pr(\{\pi \in Paths(s) \mid \pi \models \psi\}) \in J$

and a relation between paths and path formulas (next slide)

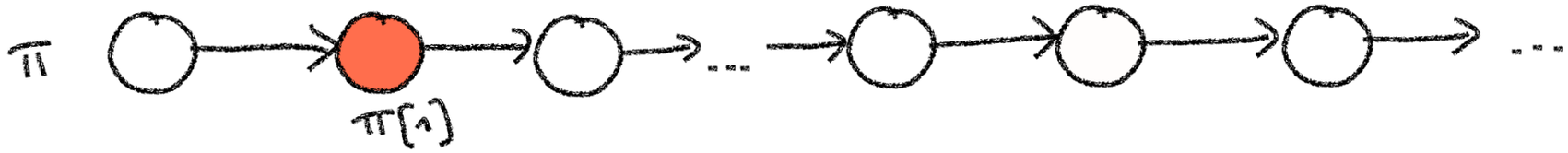
$\pi \models \psi$ iff …

$Pr_s(\psi) = Pr(\{\pi \in Paths(s) \mid \pi \models \psi\})$

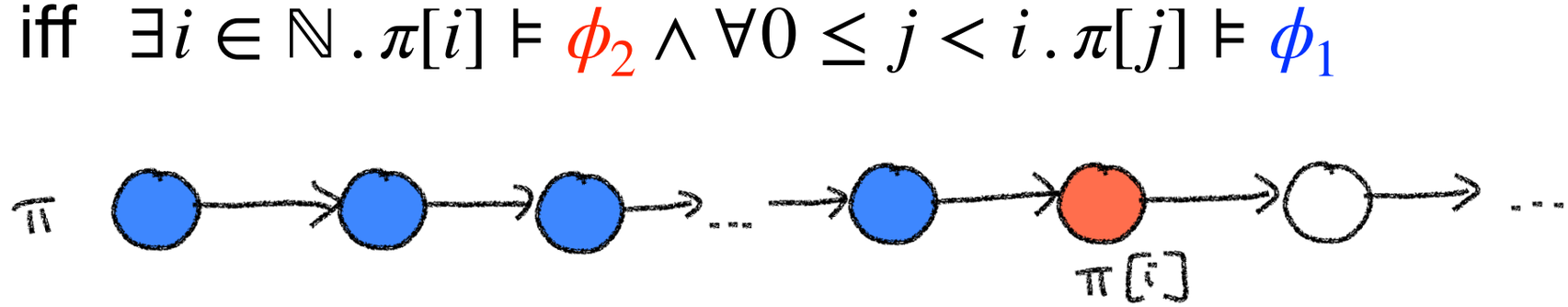
Semantics of path formulas

And here is the formal semantics of path formulas

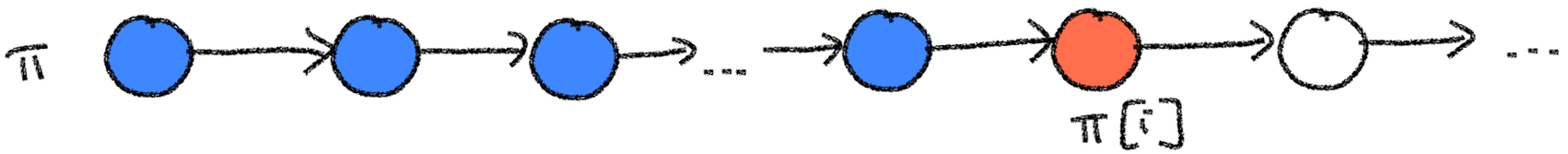
$\pi \vDash \bigcirc \phi$ iff $\pi[1] \vDash \phi$



$\pi \vDash \phi_1 \mathsf{U} \phi_2$ iff $\exists i \in \mathbb{N} \,.\, \pi[i] \vDash \phi_2 \wedge \forall 0 \leq j < i \,.\, \pi[j] \vDash \phi_1$



$\pi \vDash \phi_1 \mathsf{U}^{\leq n} \phi_2$ iff $\exists i \leq n \,.\, \pi[i] \vDash \phi_2 \wedge \forall 0 \leq j < i \,.\, \pi[j] \vDash \phi_1$



where *i* and *j* are natural numbers

Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\circ B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

Satisfaction Sets

We define the satisfaction set of a PCTL state formula as the set of states that satisfy the formula

$$sat(\phi) = \{s \mid s \vDash \phi\}$$

Semantics over a TS

We say that a transition system satisfies a formula if all its initial states satisfy the formula:

$$T \models \phi \text{ iff } \forall s \in I \,.\, s \models \phi$$

or, equivalently

$$T \models \phi \text{ iff } I \subseteq sat(\phi)$$

We use the latter in our algorithm

```
modelCheck(TS,phi) = {
   return I ⊆ sat(phi);
}
```

Main algorithm

Now that we have our main algorithm

```
modelCheck(TS,phi) = {
   return I ⊆ sat(phi);
}
```

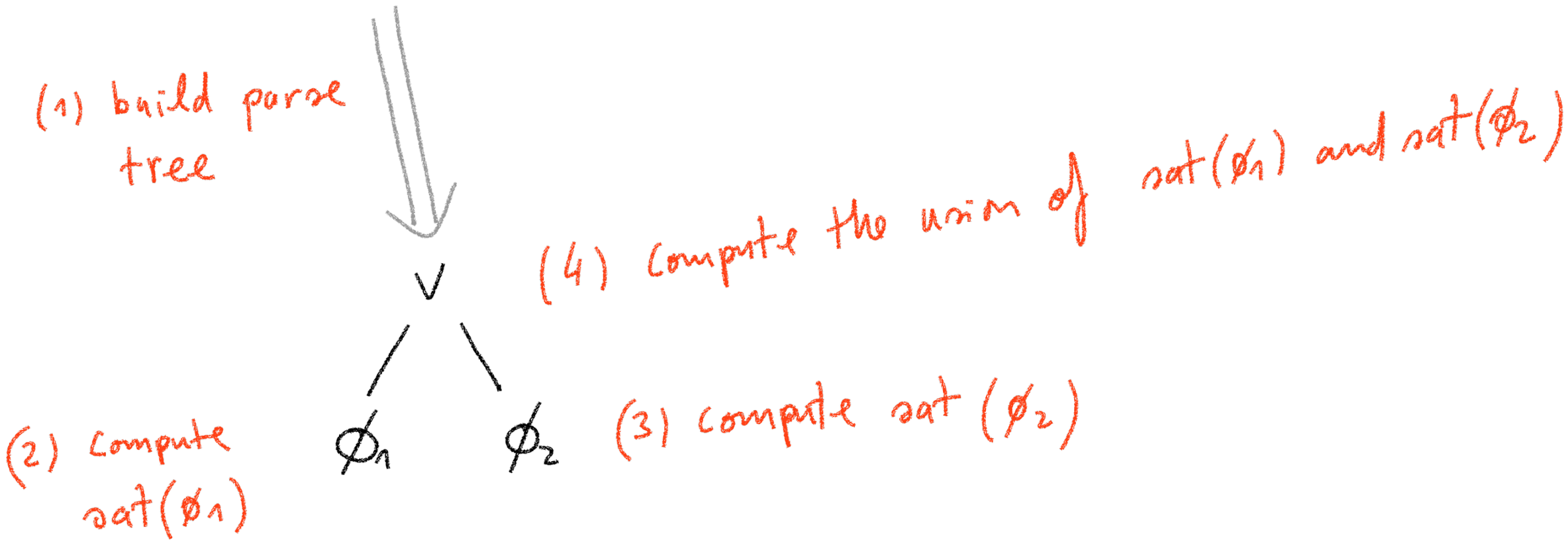
All we need to do is to implement function **`sat(…)`**

```
sat(true) = …
sat(p) =…
sat(not phi) = …
sat(phi1 and phi2) = …
sat(Pr_J(psi) = …
```

Recursive/bottom-up computation

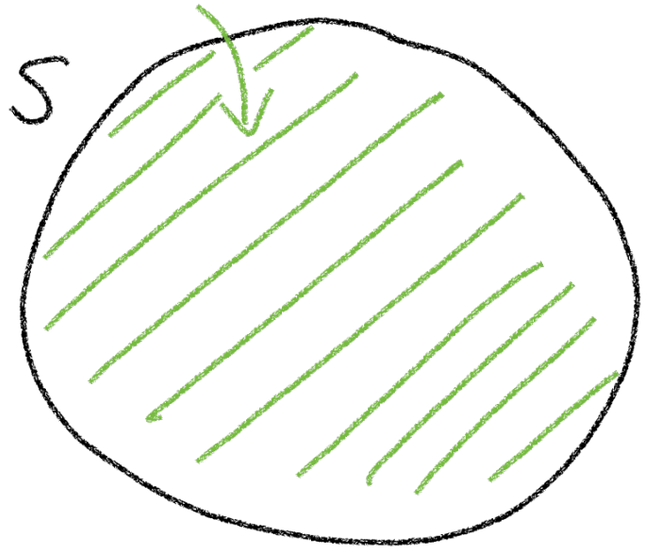
You can think of the recursive computation of satisfaction sets as as bottom-up computation on the parse tree of the formula

$$sat(\phi_1 \vee \phi_2) = sat(\phi_1) \cup sat(\phi_2)$$

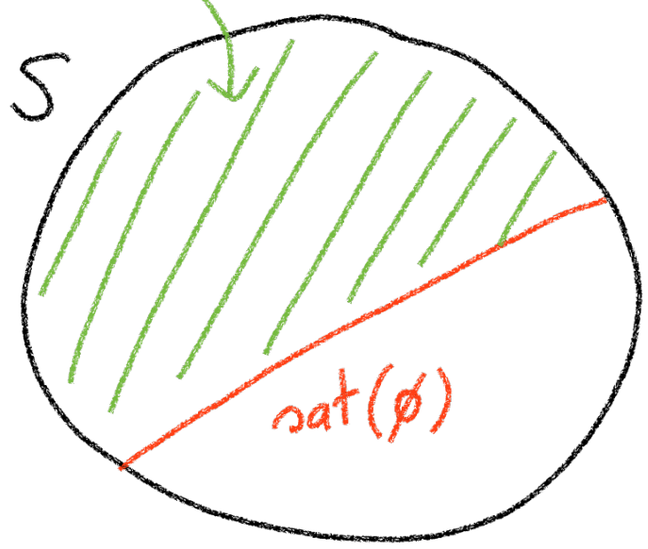


Satisfaction sets characterisation of propositional fragment

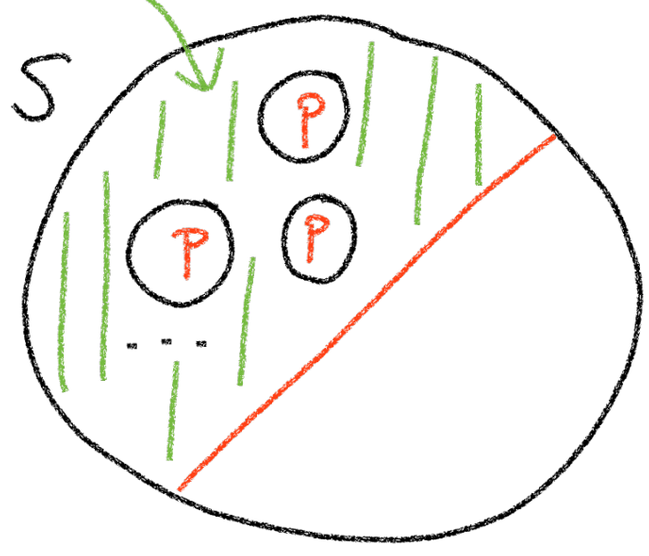
$$sat(true) = S$$



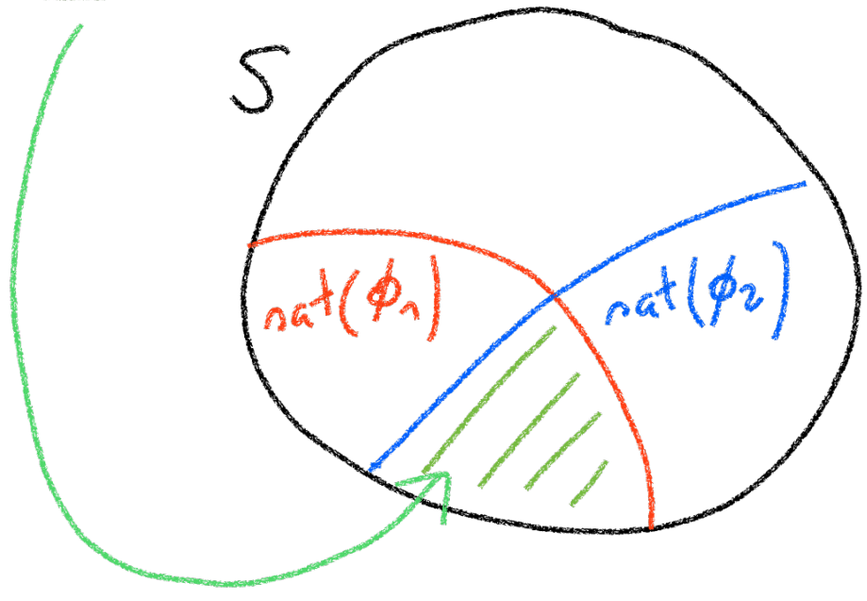
$$sat(\neg\phi) = S \setminus sat(\phi)$$



$$sat(p) = \{s \in S \mid p \in L(s)\}$$



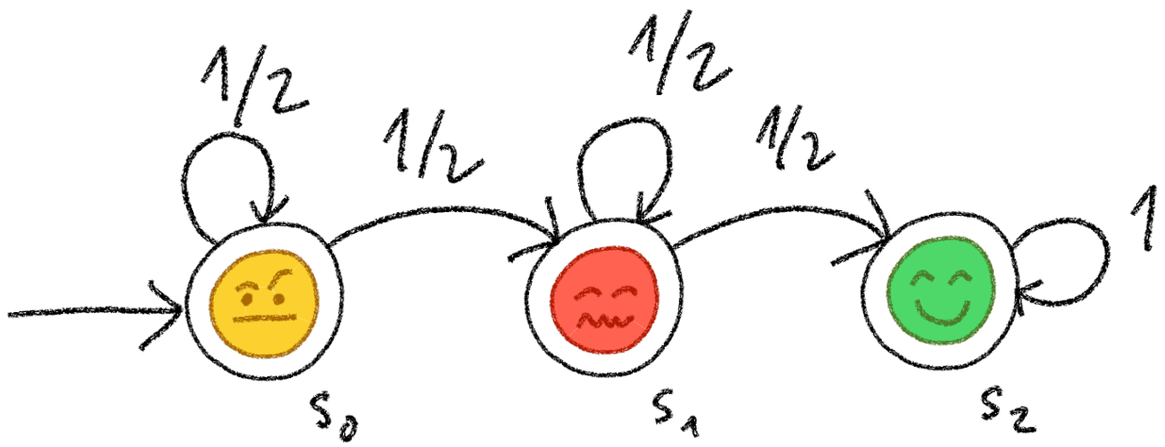
$$sat(\phi_1 \wedge \phi_2) = sat(\phi_1) \cap sat(\phi_2)$$



Model Checking PCTL

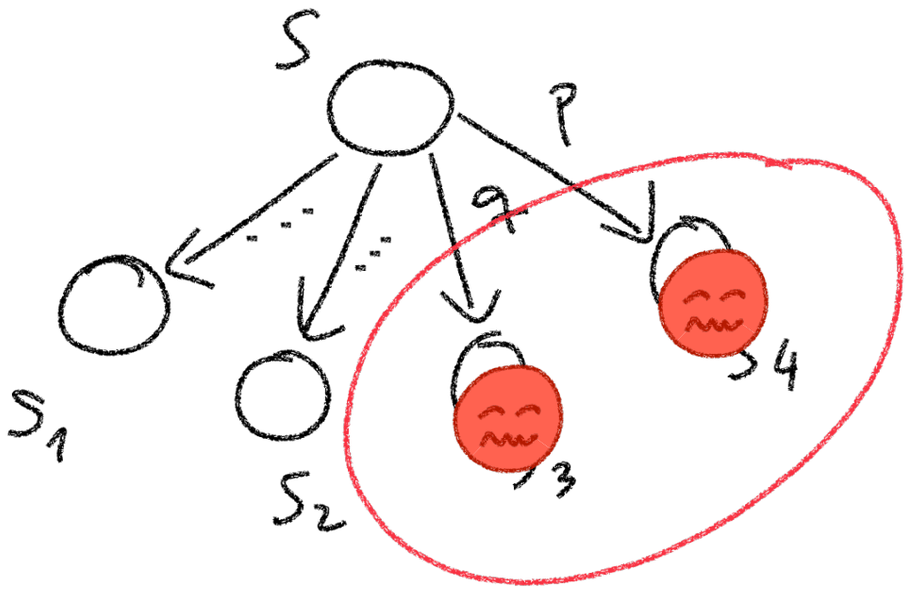
- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\circ B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

Next-Step probability in our example



What is the probability of getting infected tomorrow?

$$Pr_{s_0}(\bigcirc) = \ldots \frac{1}{2}$$



Computing $Pr_s(\bigcirc B)$

Recall the semantic definition:

$$Pr_s(\bigcirc B) = \sum_{s' \in B} P(s, s')$$

We can trivially perform such computation algorithmically

Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\bigcirc B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

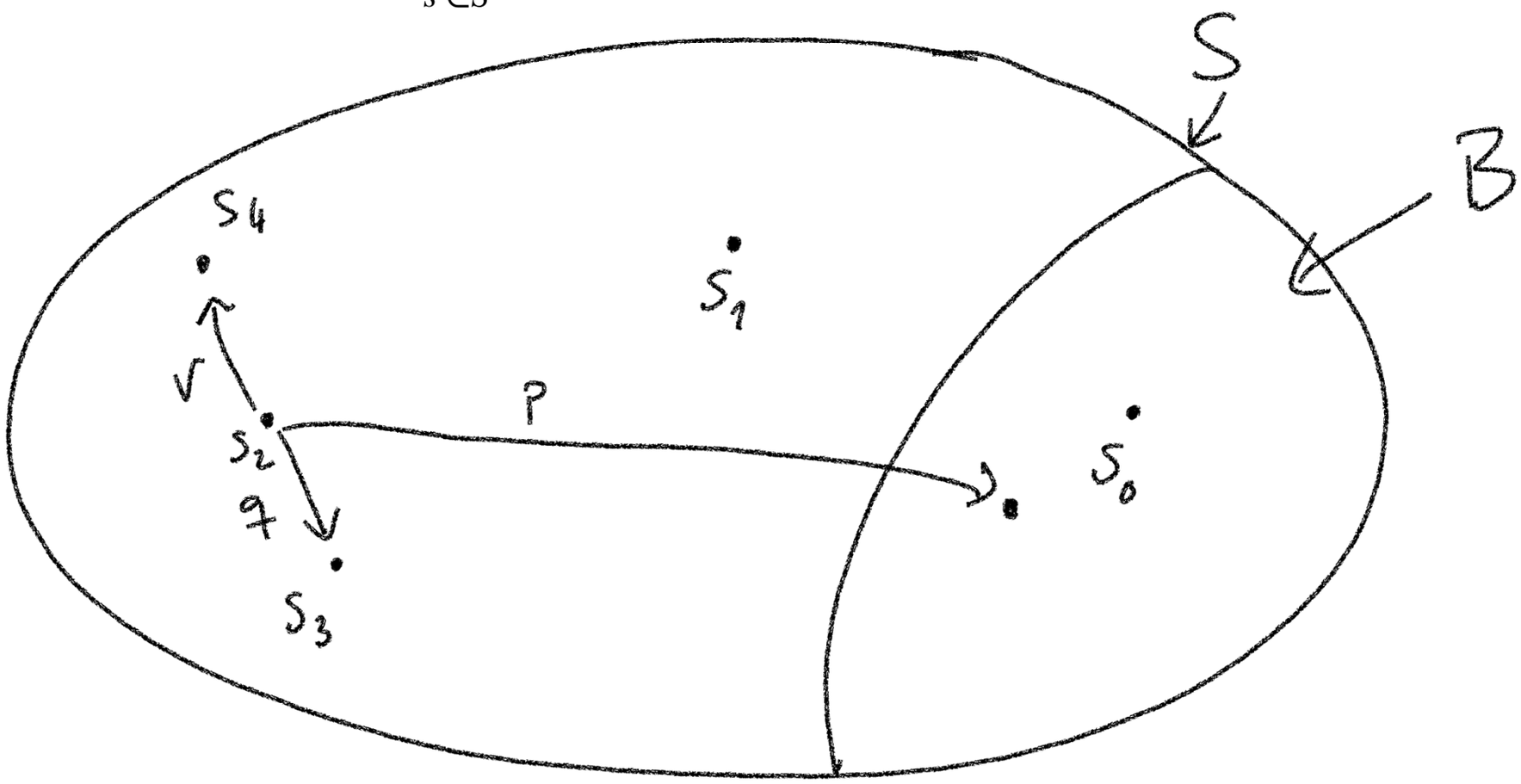
Computing $Pr_s(\Diamond^{\leq n} B)$

We can use recursion!

$$Pr_s(\Diamond^{\leq n} B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(\Diamond^{\leq 0} B) = 0 \qquad \text{if } s \notin B$$

$$Pr_s(\Diamond^{\leq n+1} B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(\Diamond^{\leq n} B) \qquad \text{otherwise}$$



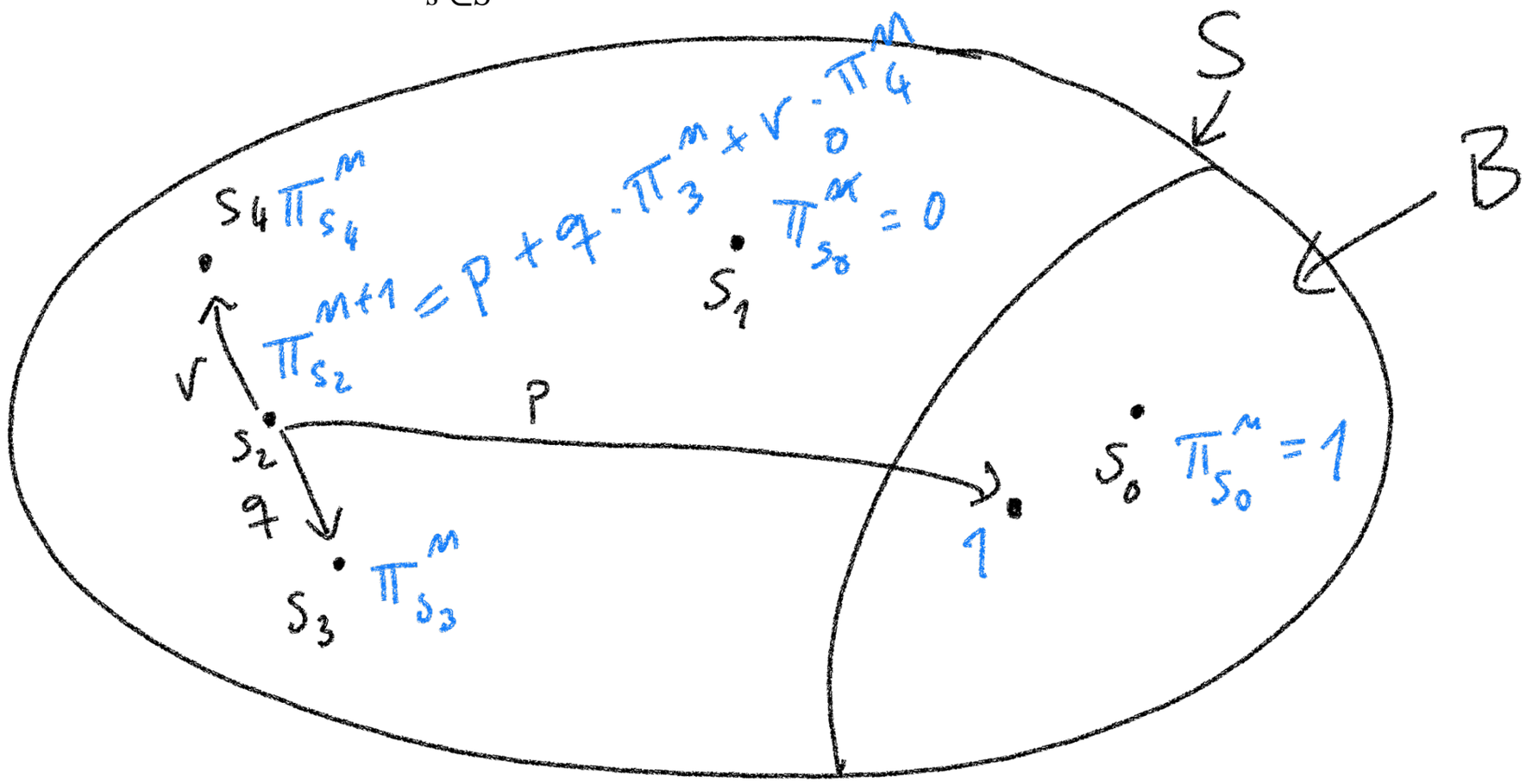
Computing $Pr_s(\Diamond^{\leq n} B)$

We can use recursion!

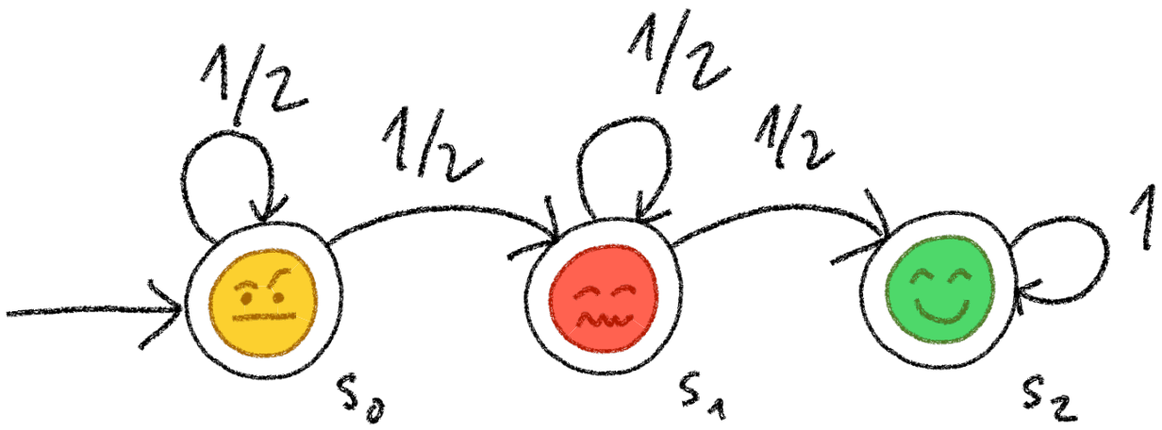
$$Pr_s(\Diamond^{\leq n} B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(\Diamond^{\leq 0} B) = 0 \qquad \text{if } s \notin B$$

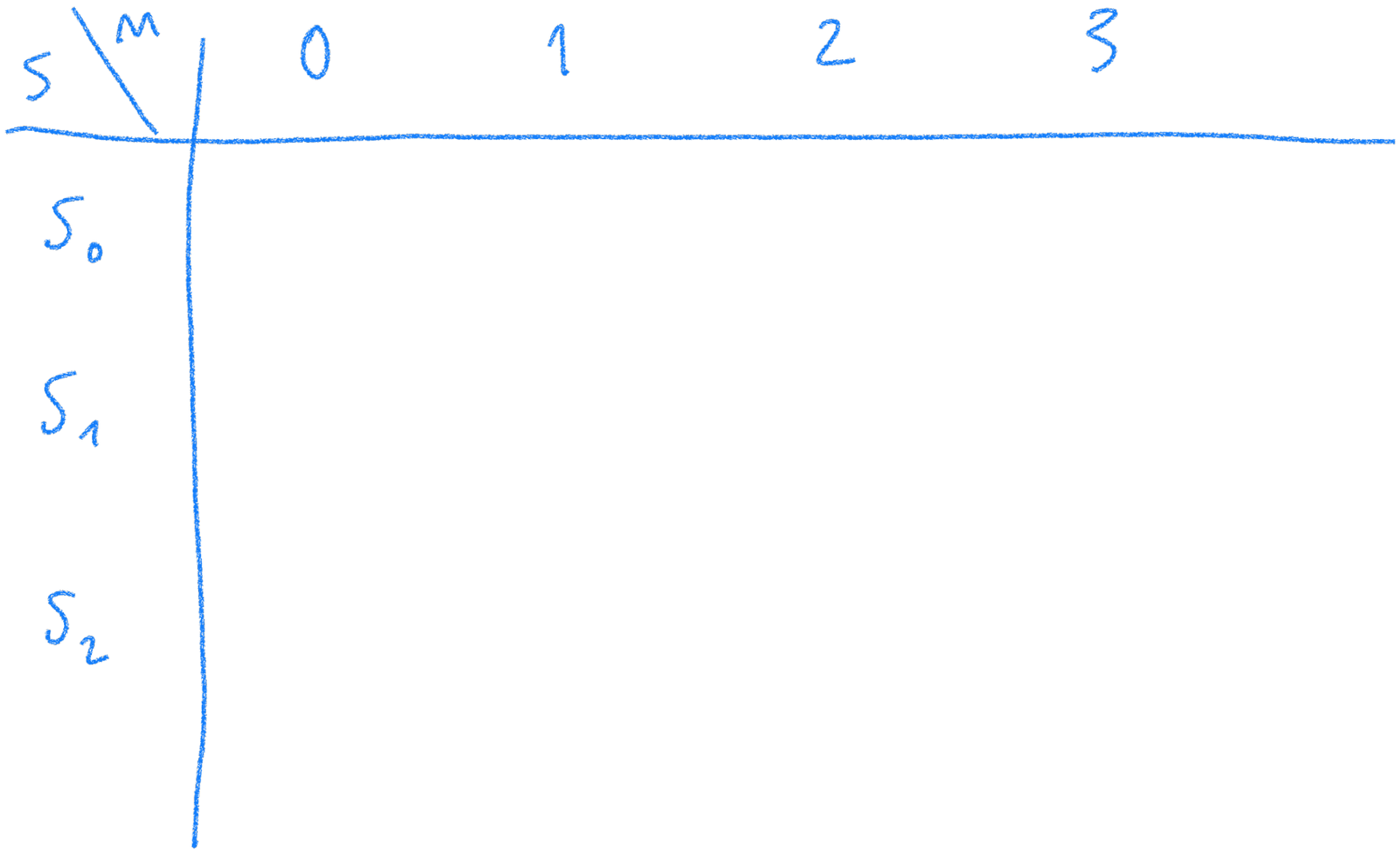
$$Pr_s(\Diamond^{\leq n+1} B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(\Diamond^{\leq n} B) \qquad \text{otherwise}$$



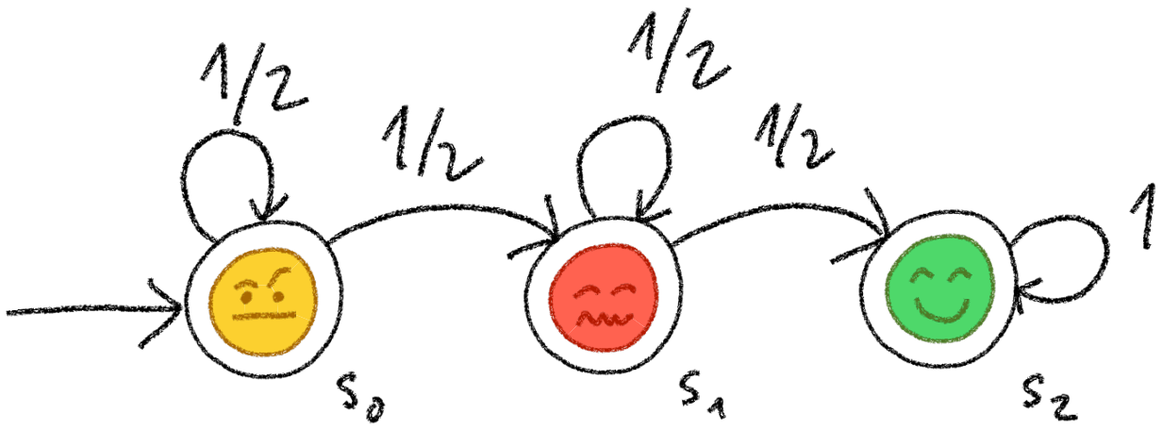
Let's try with an example



$Pr_s(\Diamond^{\leq n}$  $) = \ldots$



Let's try with an example



$$Pr_s(\Diamond^{\leq n} \; \text{☺} \;) = \dots$$

s \ n	0	1	2	3
s_0	0	~~$1/2 \cdot 0 + 1/2 \cdot 0$~~ $= 0$	$1/2 \cdot 0 + 1/2 \cdot 1/2 = 1/4$	...
s_1	0	$1/2 \cdot 0 + 1/2 \cdot 1 = 1/2$	$1/2 \cdot 1/2 + 1/2 \cdot 1 = 3/4$	...
s_2	1	1	1	1 .....

Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\bigcirc B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

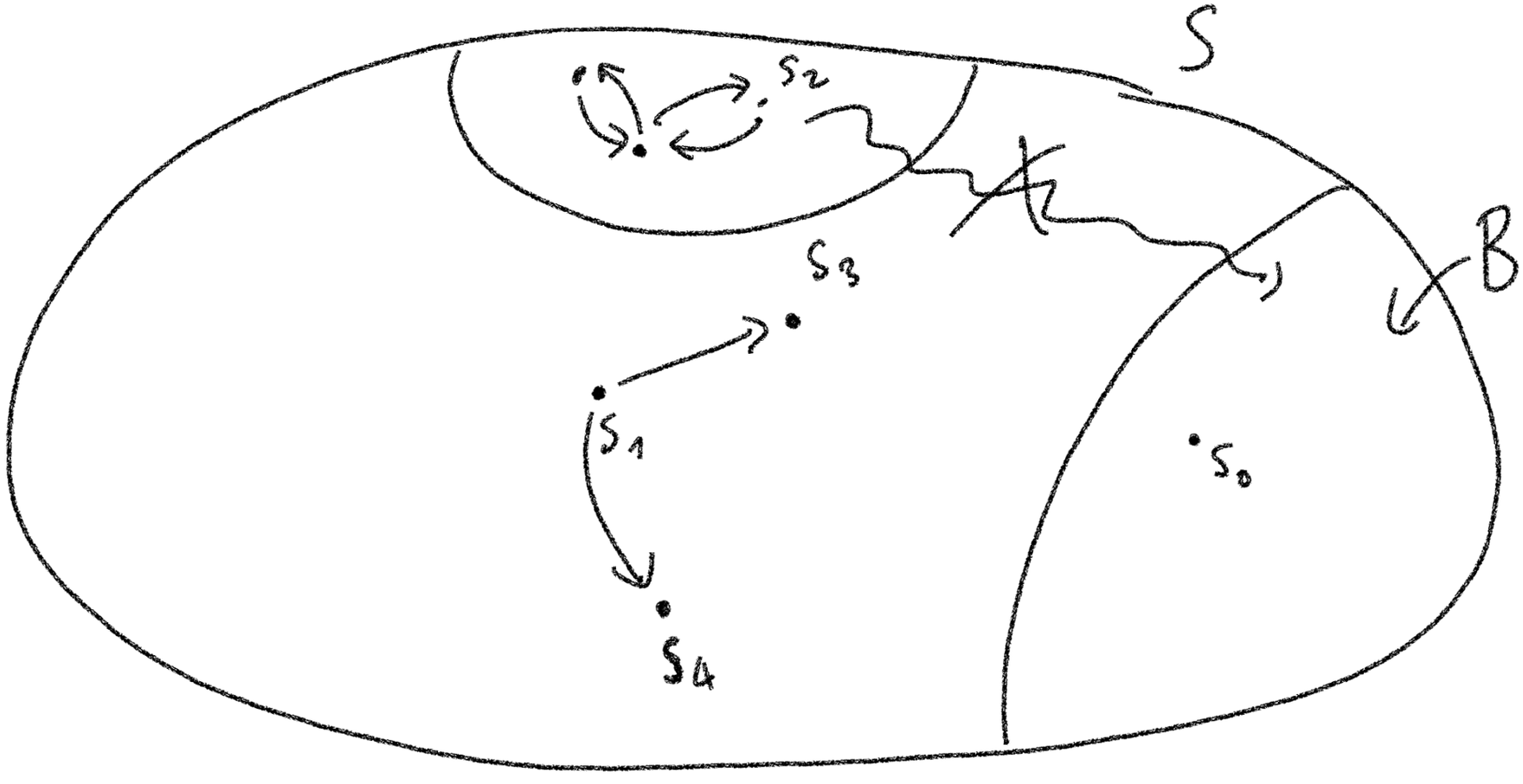
Computing $Pr_s(\Diamond B)$

It is as simple as solving the following system of equations

$$Pr_s(\Diamond B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(\Diamond B) = 0 \qquad \text{if } s \models \neg \exists \Diamond B$$

$$Pr_s(\Diamond B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(\Diamond B) \qquad \text{otherwise}$$



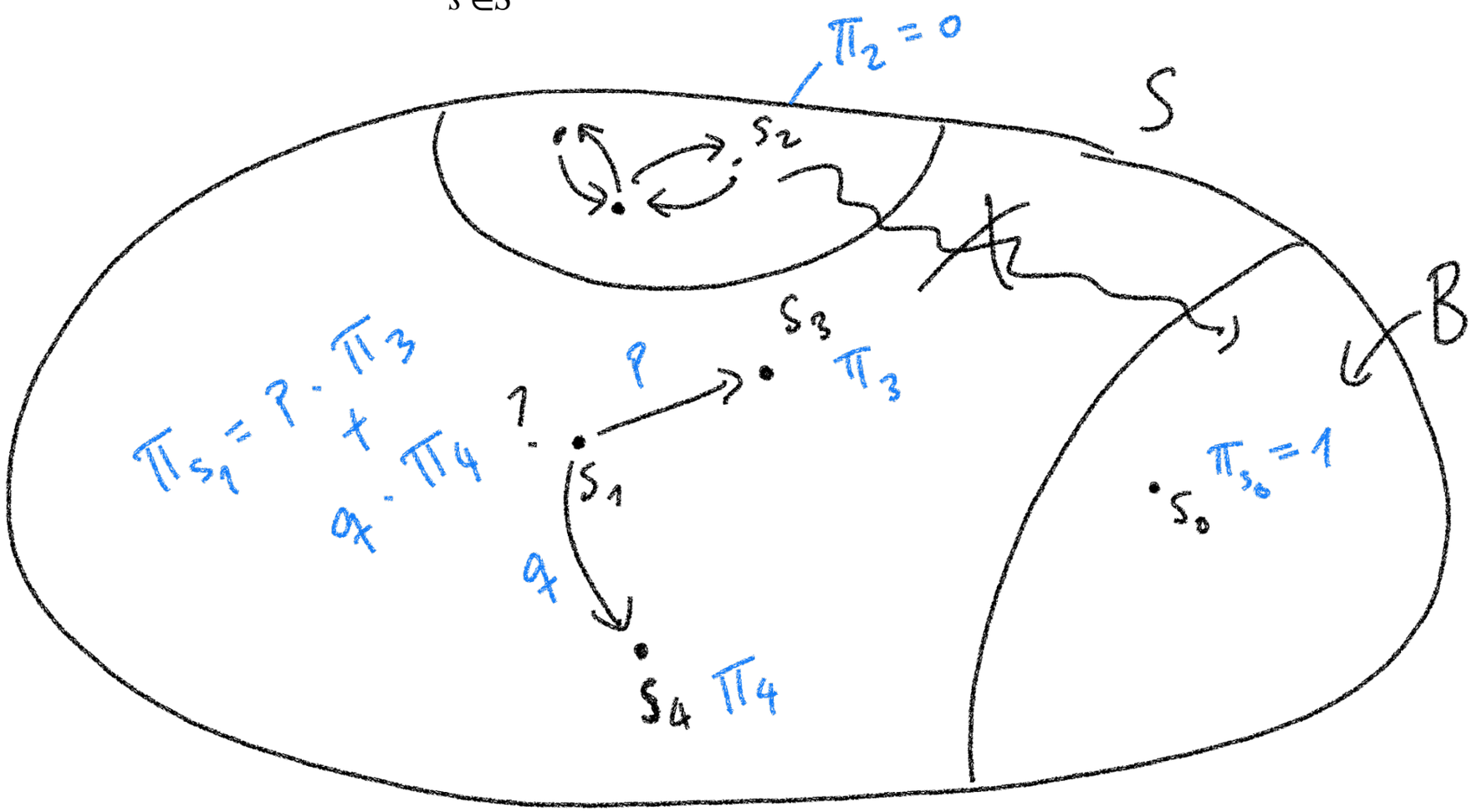
Computing $Pr_s(\Diamond B)$ $= \pi_s$

It is as simple as solving the following system of equations

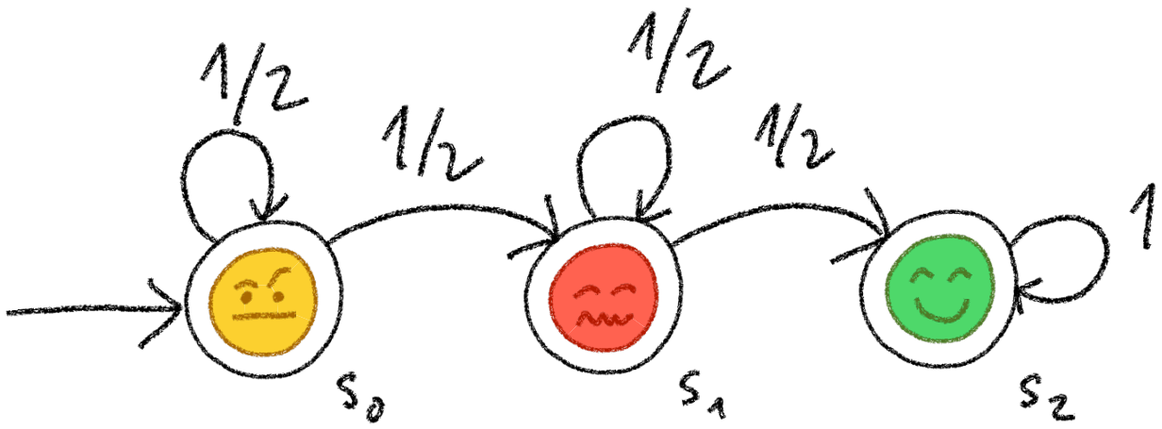
$$Pr_s(\Diamond B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(\Diamond B) = 0 \qquad \text{if } s \models \neg \exists \Diamond B$$

$$Pr_s(\Diamond B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(\Diamond B) \qquad \text{otherwise}$$

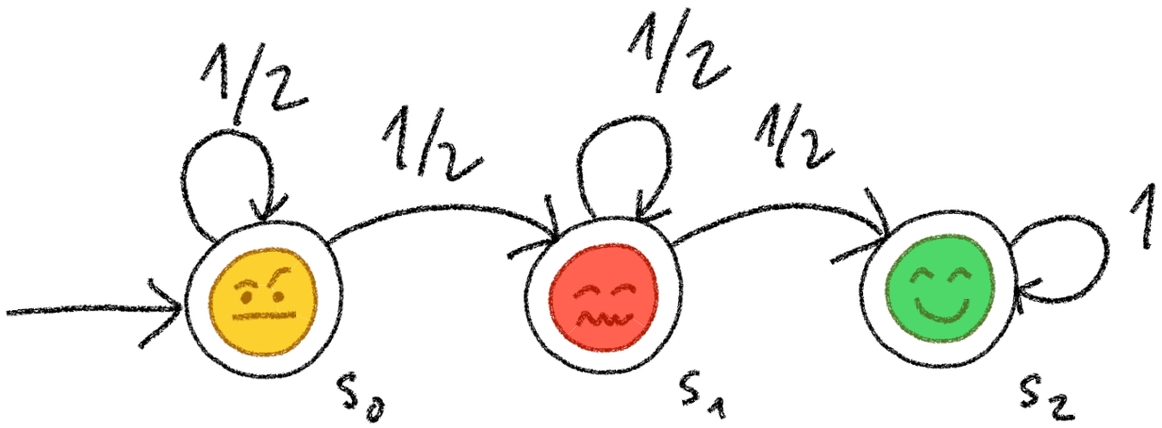


Let’s try with an example



$Pr_s(\Diamond$  $) = \ldots$

Let's try with an example

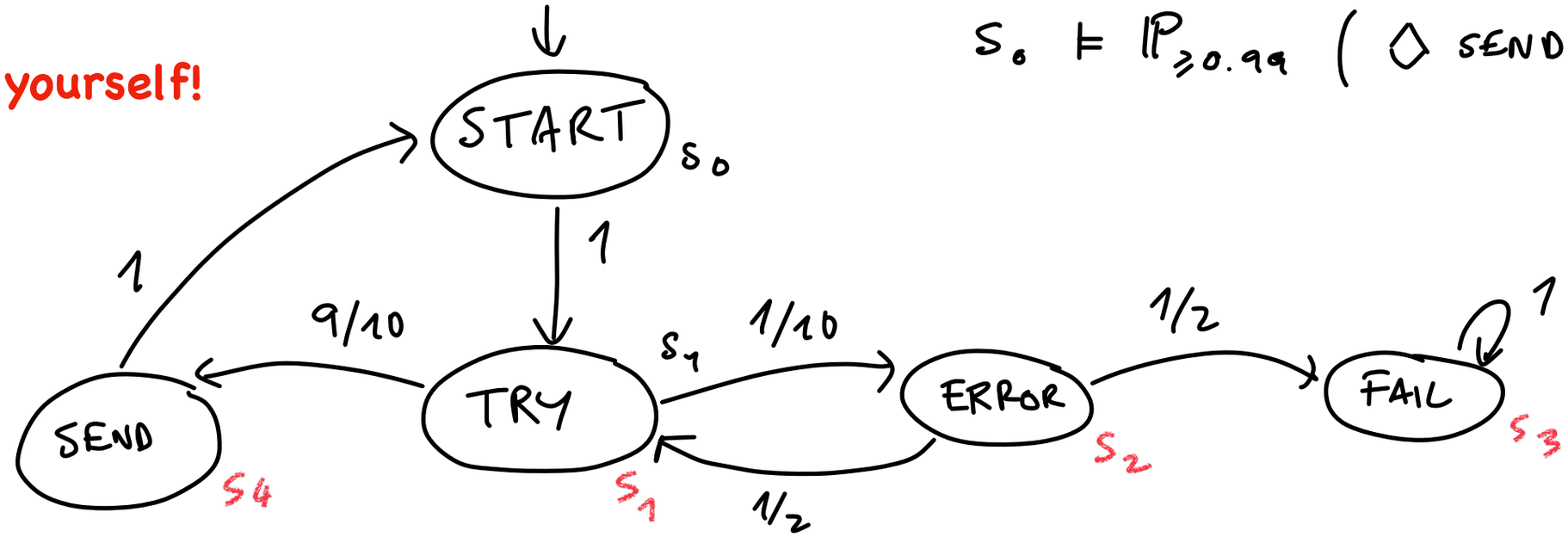


$\pi_s = Pr_s(\Diamond$  $) = \dots$

$$\left.\begin{aligned} \pi_0 &= 1/2 \cdot \pi_0 + 1/2 \cdot \pi_1 \\ \pi_1 &= 1/2 \cdot \pi_1 + 1/2 \cdot \pi_2 \\ \pi_2 &= 1 \end{aligned}\right\} \Longrightarrow \begin{aligned} \pi_0 &= 1 \\ \pi_1 &= 1 \\ \pi_2 &= 1 \end{aligned}$$

Try yourself!

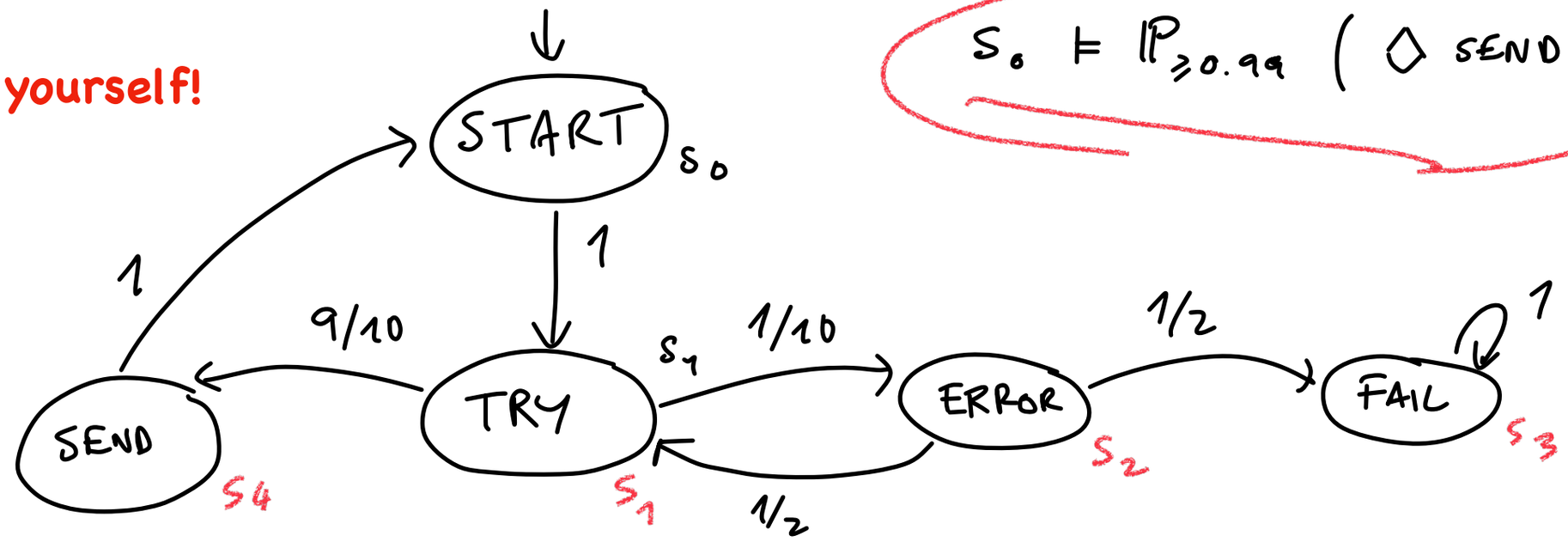
$s_0 \models \mathbb{P}_{\geq 0.99}(\Diamond \text{SEND})$?



See file **lossy-channel.py**

Try yourself!

$s_0 \models \mathbb{P}_{\geq 0.99}(\Diamond\, \text{SEND})$ ✗



$$Pr_{s_0}(\Diamond\, \text{SEND})$$

$$\pi_s = Pr_s(\Diamond\, \text{SEND}) = \begin{cases} 1 & \text{if } s \models \text{SEND} \\ 0 & \text{if } s \not\models \exists\, \Diamond\, \text{SEND} \\ \sum_{s' \in S} P(s,s') \cdot Pr_{s'}(\Diamond\, \text{SEND}) & \end{cases}$$

$\pi_0 = 1 \cdot \pi_1 = \pi_1 = 18/19 = 0.947\ldots$

$\pi_1 = 9/10 \cdot \pi_4 + 1/10 \cdot \pi_2 = 9/10 + 1/10 \cdot \pi_2 = 9/10 + 1/20\, \pi_1$

$\pi_2 = 1/2 \cdot \pi_1 + 1/2 \cdot \pi_3 = 1/2 \cdot \pi_1 = 18/38$

$\pi_3 = 0$

$\pi_4 = 1$

$\pi_1 = 9/10 + 1/20\, \pi_1$

$19/20\, \pi_1 = 9/10$

$\pi_1 = 9/10 \cdot \frac{20}{19} = 180/190$

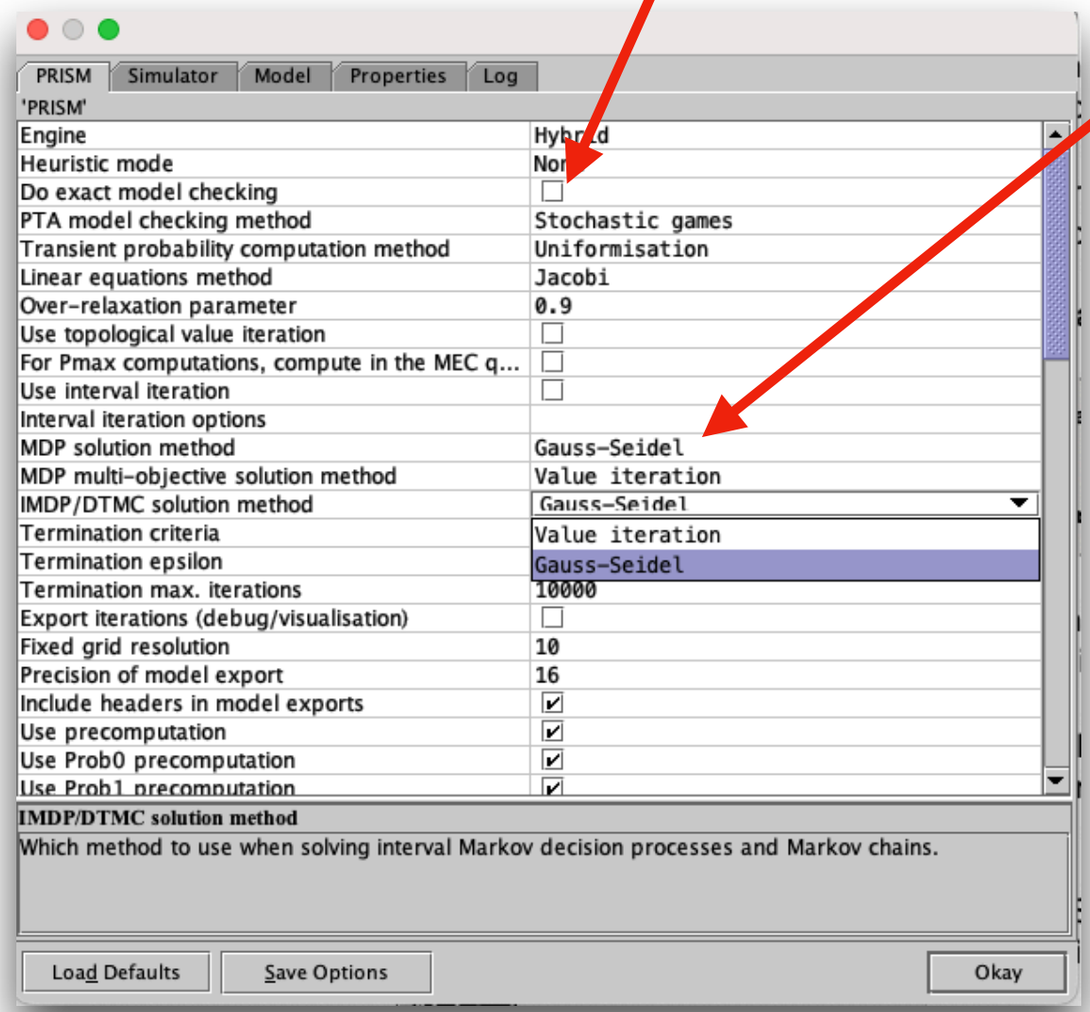
See file **lossy-channel.py**

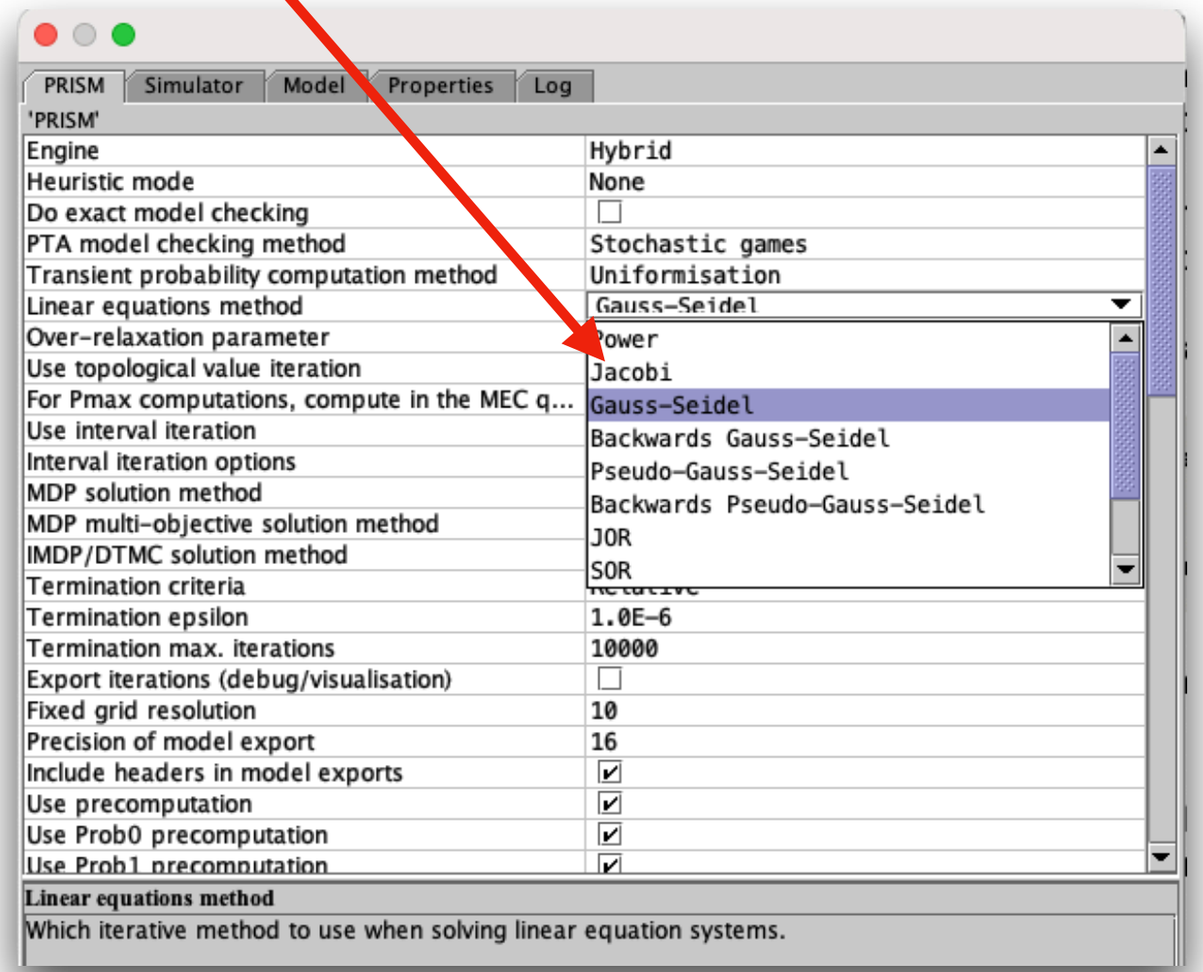
Algorithms for solving the equations…

arbitrary precision (rationals) vs floating point precision (doubles/floats)

direct (precise solutions, “expensive”)

Iterative (approximations, “faster”)





Recall your linear algebra (cf. MAT01) lectures :)

See files **`lossy-channel.prism`** **`lossy-channel.props`**

Compare results with **`lossy-channel.py`**

Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\circ B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

Computing $Pr_s(C\ \mathsf{U}^{\leq n}\ B)$

We can use recursion!

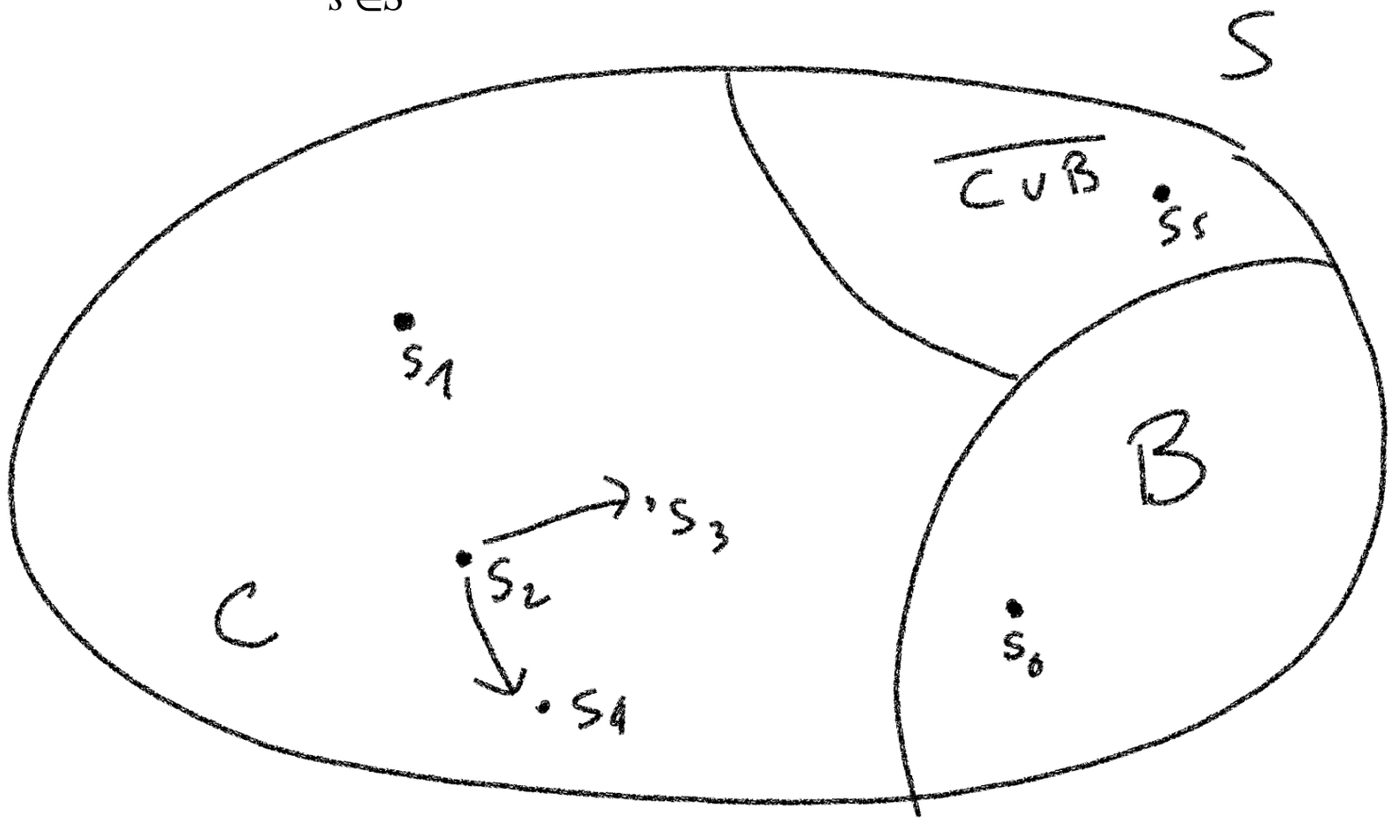
$$Pr_s(C \mathsf{U}^{\leq n} B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(C \mathsf{U}^{\leq 0} B) = 0 \qquad \text{if } s \notin B$$

$$Pr_s(C \mathsf{U}^{\leq n} B) = 0 \qquad \text{if } s \notin (B \cup C)$$

$$Pr_s(C \mathsf{U}^{\leq n+1} B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(C \mathsf{U}^{\leq n} B) \qquad \text{otherwise}$$

NEW!



Computing $Pr_s(C \ \mathsf{U}^{\leq n} \ B)$

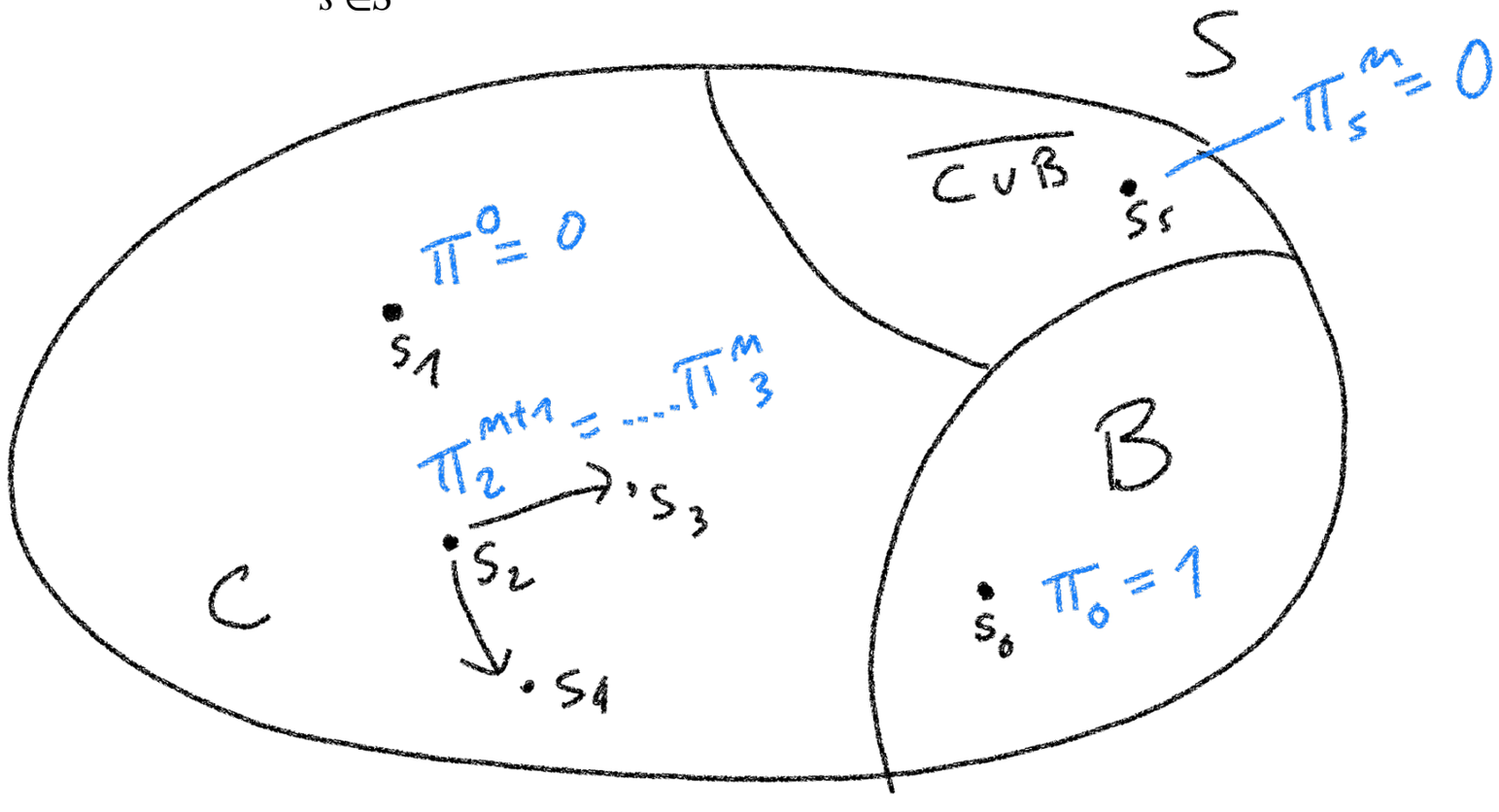
We can use recursion!

$$Pr_s(C \mathsf{U}^{\leq n} B) = 1 \qquad \text{if } s \in B$$

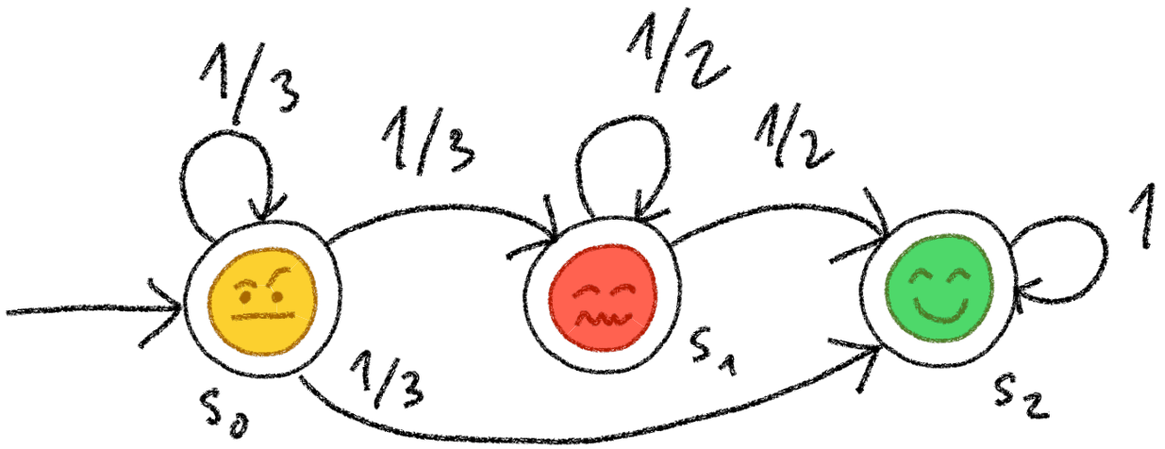
$$Pr_s(C \mathsf{U}^{\leq 0} B) = 0 \qquad \text{if } s \notin B$$

$$Pr_s(C \mathsf{U}^{\leq n} B) = 0 \qquad \text{if } s \notin (B \cup C)$$

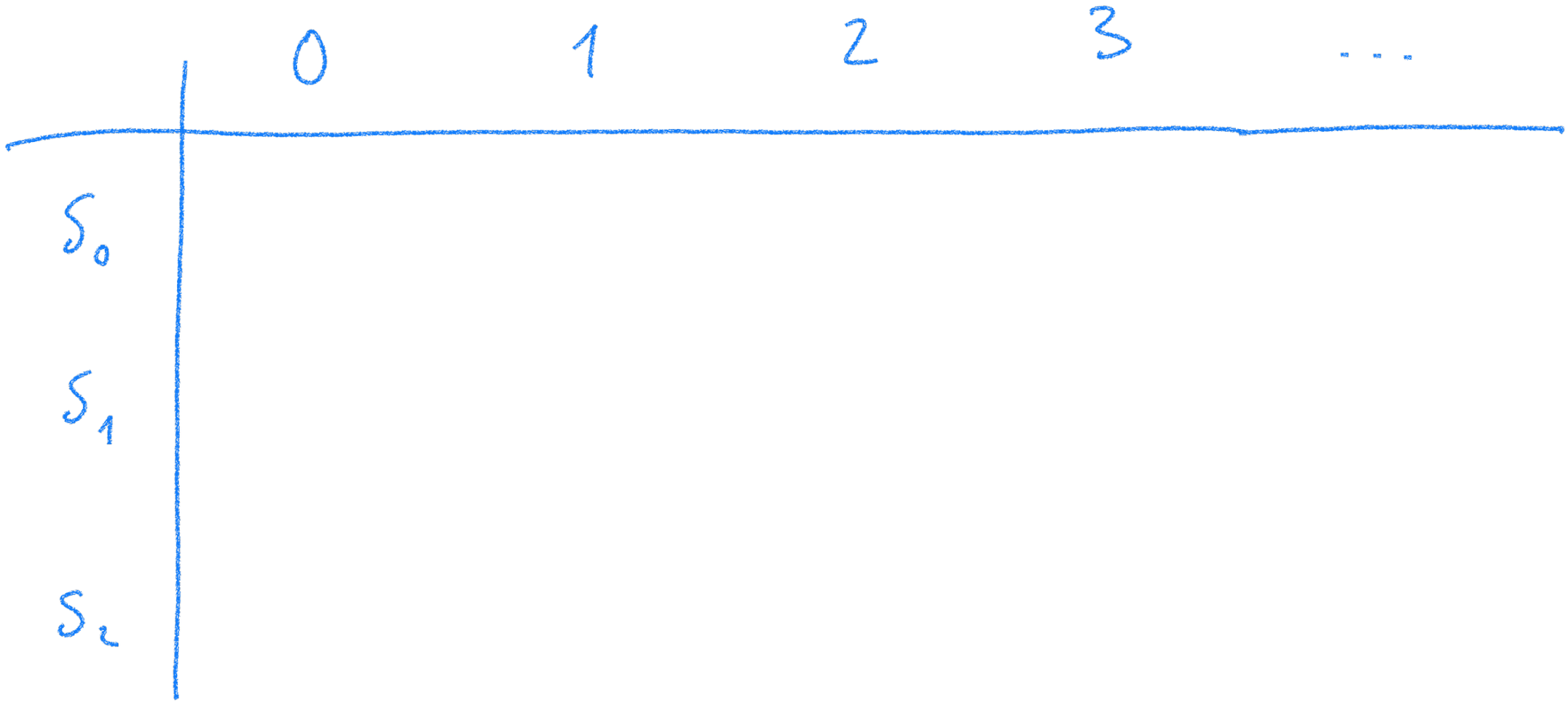
$$Pr_s(C \mathsf{U}^{\leq n+1} B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(C \mathsf{U}^{\leq n} B) \qquad \text{otherwise}$$



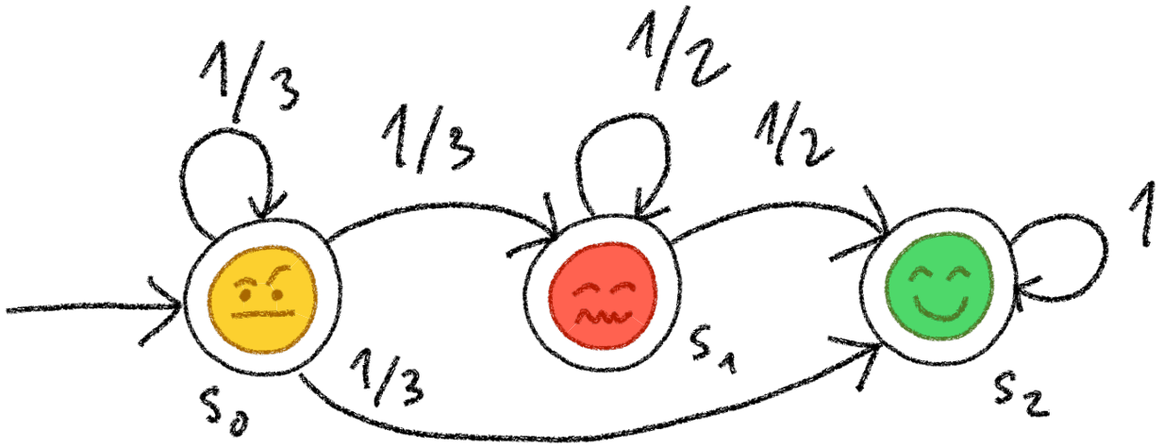
Let's try with an example



$Pr_{s_0}(\neg$  $\mathsf{U}^{\leq n}$  $) = \ldots$



Let's try with an example



$Pr_{s_0}(\neg$  $\mathsf{U}^{\leq n}$  $) = \ldots$

	0	1	2	3	...
s_0	0	$1/3 \cdot 0 + 1/3 \cdot 0 + 1/3 \cdot 1 = 1/3$	...	...	...
s_1	0	0	0	0	0
s_2	1	1	1	1	1

Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\circ B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

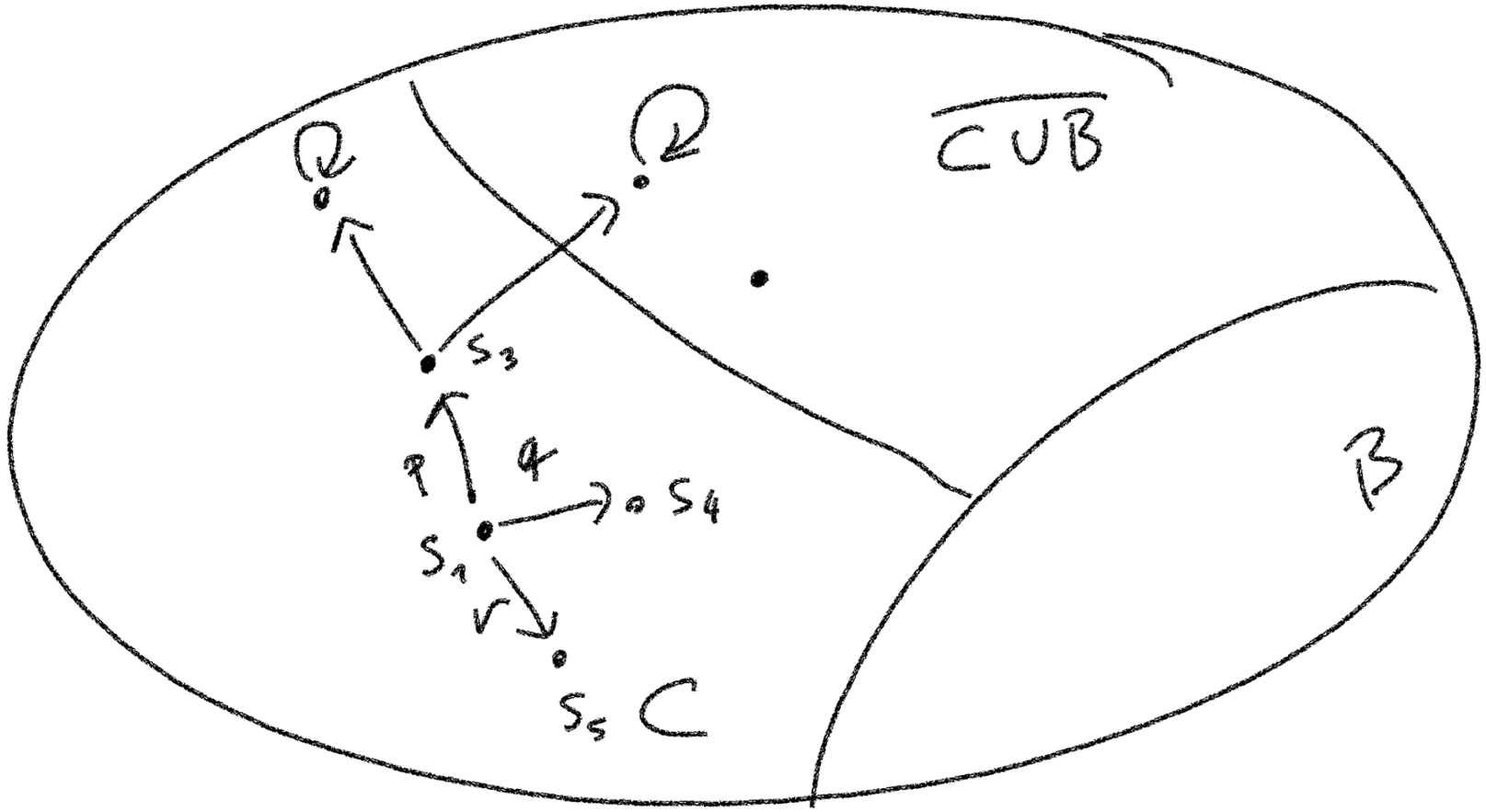
Computing $Pr_s(C \cup B)$

It is as simple as solving the following system of equations

$$Pr_s(C \cup B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(C \cup B) = 0 \qquad \text{if } s \models \neg \exists C \cup B$$

$$Pr_s(C \cup B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(C \cup B) \qquad \text{otherwise}$$



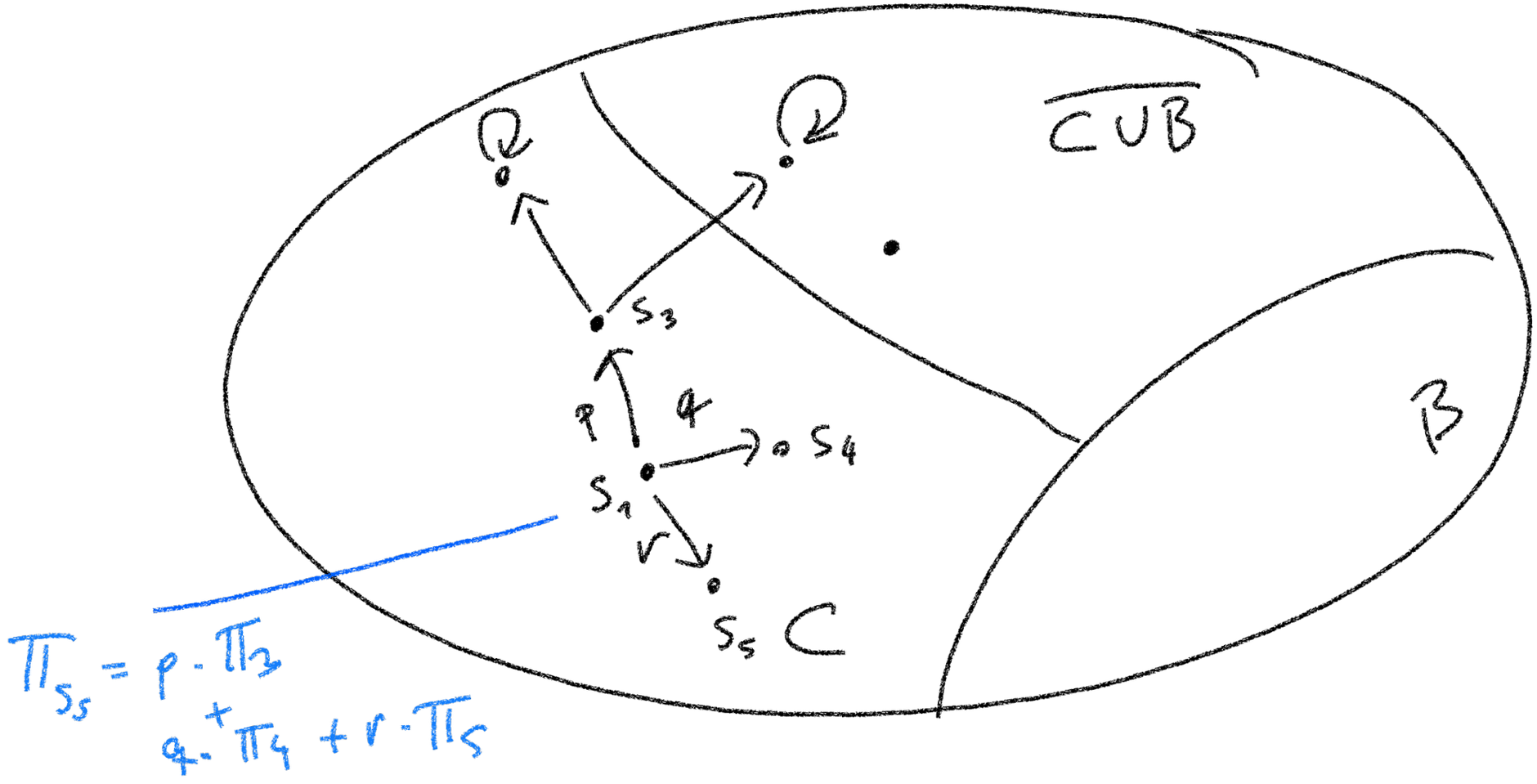
Computing $Pr_s(C \cup B)$

It is as simple as solving the following system of equations

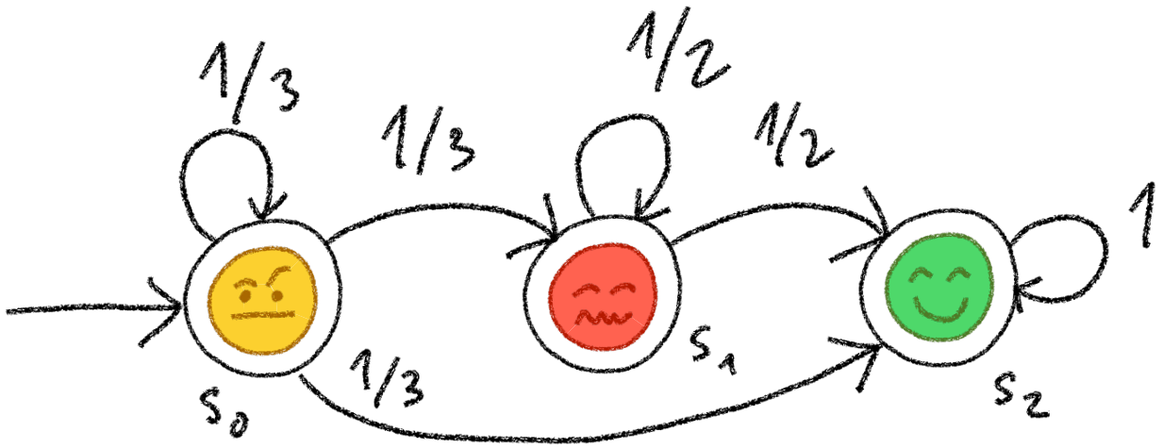
$$Pr_s(C \cup B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(C \cup B) = 0 \qquad \text{if } s \models \neg \exists C \cup B$$

$$Pr_s(C \cup B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(C \cup B) \qquad \text{otherwise}$$



Let's try with an example



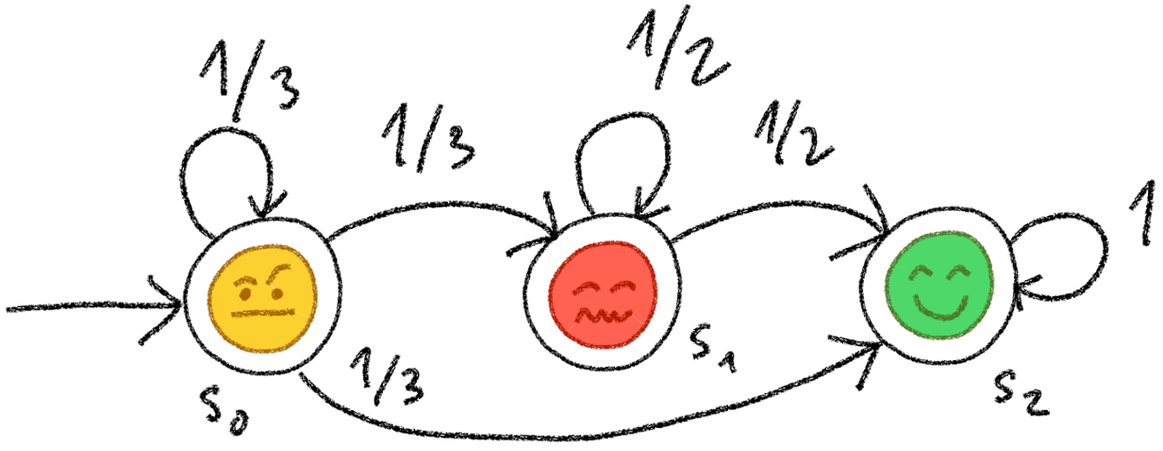
$Pr_{s_0}(\neg$  U  $) = \ldots$

$\pi_0 =$

$\pi_1 =$

$\pi_2 =$

Let's try with an example



$Pr_{s_0}(\neg$  U  $) = \ldots$

$\pi_0 = 1/3\,\pi_0 + 1/3\,\pi_1 + 1/3\,\pi_2 \rightarrow \pi_0 = 1/2$

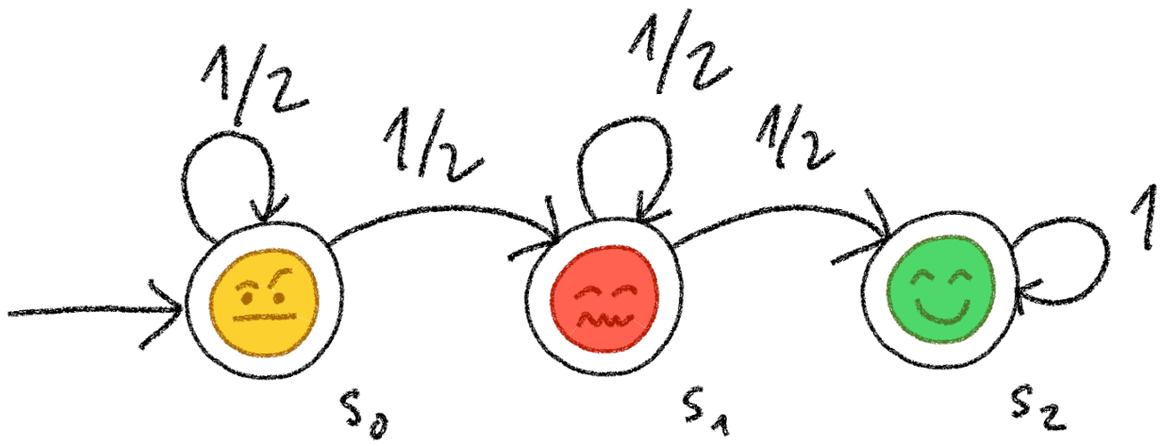
$\pi_1 = 0$

$\pi_2 = 1$

Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\circ B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

Probabilistic Bounded Reachability via Transient Distribution?

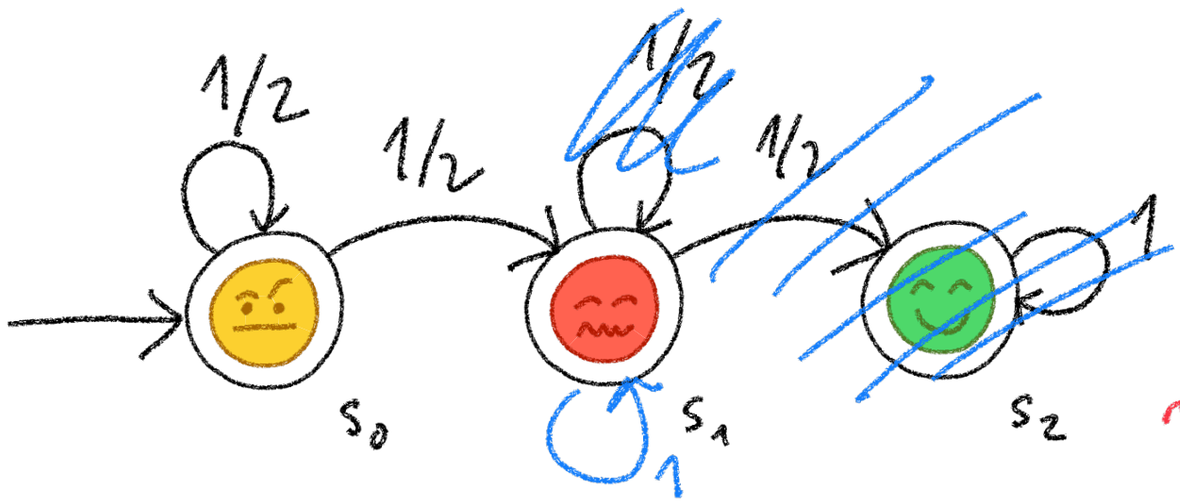


What is the probability of getting infected in n days or less?

$$Pr_{s_0}(\Diamond^{\leq n} \text{😷}) = \dots$$

n	$Pr_{s_0}(\Diamond^{\leq n}$ 😷$)$		$\theta_n($ 😷 $)$
0	0		0
1	1/2		1/2
2	3/4	$\neq$	1/2
3			
4			

Probabilistic Bounded Reachability via Transient Distribution?



What is the probability of getting infected in n days or less?

$$Pr_{s_0}(\Diamond^{\leq n} \text{😷}) = \dots$$

↑ they coincide on modified DTMC (see book)

n	$Pr_{s_0}(\Diamond^{\leq n}$ 😷$)$		$\theta_n($ 😷$)$
0	0		0
1	1/2		1/2
2	3/4	$\neq$	1/2
3			
4			

PCTL via CTL?

If only we could encode PCTL into CTL we could skip the next lessons!

Do you think the following statements hold?

$s \models \mathbb{P}_{>0}(\Diamond \phi)$ iff $s \models \exists \Diamond \phi$

$s \models \mathbb{P}_{=1}(\Diamond \phi)$ iff $s \models \forall \Diamond \phi$

PCTL via CTL?

If only we could encode PCTL into CTL we could skip the next lessons!

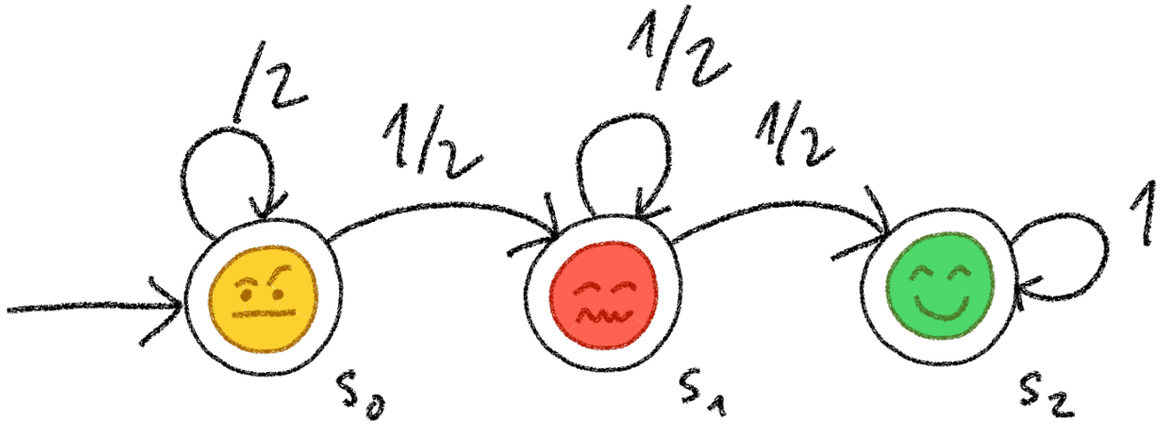
Do you think the following statements hold?

$s \models \mathbb{P}_{>0}(\Diamond \phi)$ iff $s \models \exists \Diamond \phi$

holds!

$s \models \mathbb{P}_{=1}(\Diamond \phi)$ iff $s \models \forall \Diamond \phi$

does not hold!



$\models \mathbb{P}_{=1}(\Diamond$ 😊 $)$

$\not\models \forall \Diamond$ 😊

BONUS: parameter synthesis

We have focused on the classical model checking problem, the the model M is fully specified.

$$M \models \Phi ?$$

… and on SAT/SMT where we look for a model

$$\exists M \,.\, M \models \Phi ?$$

We could also consider cases where we have partially specified models:

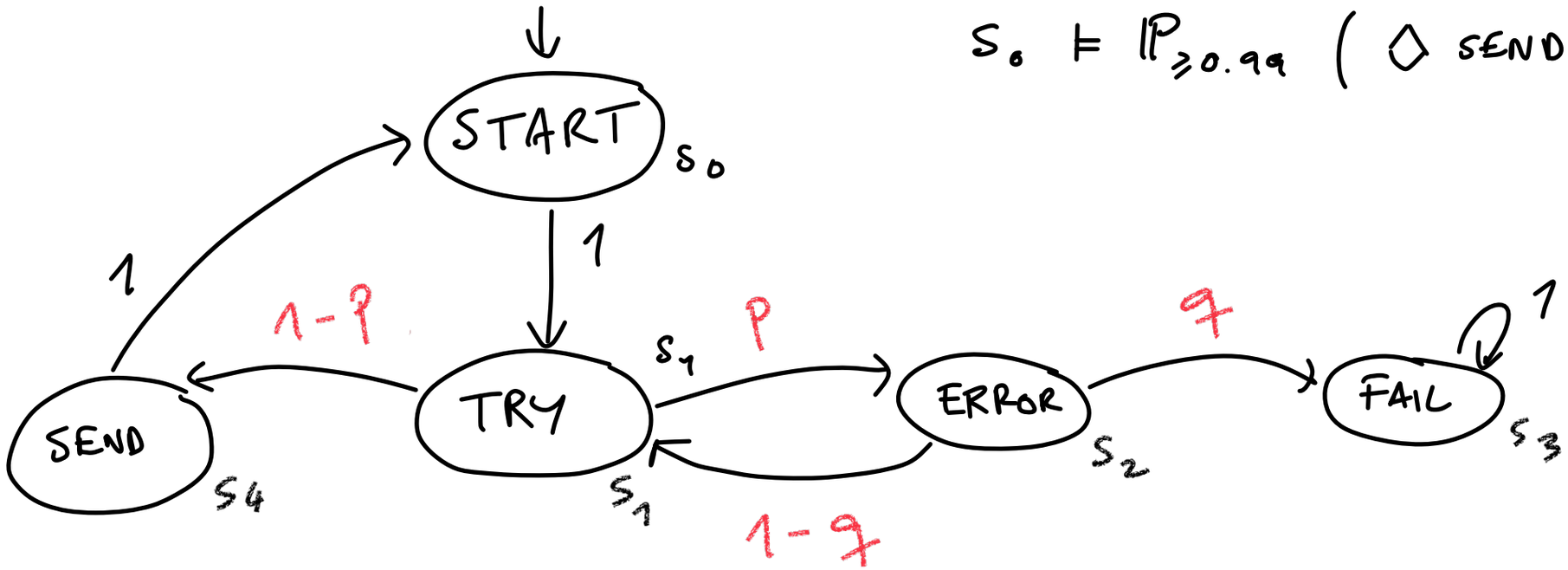
$$\exists x, y, z \,.\, M(x, y, z) \models \Phi ?$$

We don't want the probability of eventually sending above a certain threshold.

We can choose the probability of error p and failure q (imagine that we can spend spend more or less on the transmission hardware).

How do we choose them?

Find p, q such that $s_0 \models \mathbb{P}_{\geq 0.99}(\Diamond \text{SEND})$?



Computing $Pr_s(\Diamond B)$

It is as simple as solving the following system of equations

$$Pr_s(\Diamond B) = 1 \qquad \text{if } s \in B$$

$$Pr_s(\Diamond B) = 0 \qquad \text{if } s \models \neg\exists\Diamond B$$

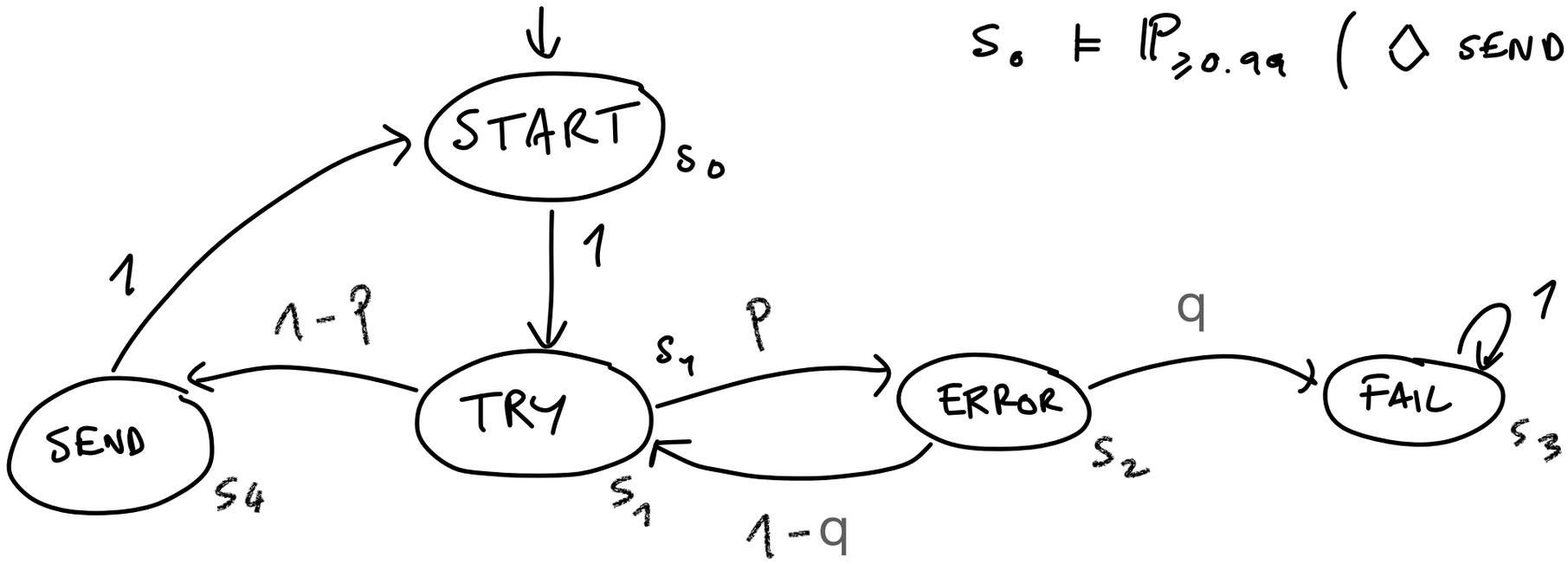
$$Pr_s(\Diamond B) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Pr_{s'}(\Diamond B) \qquad \text{otherwise}$$

What happens if P(s,s') is (partially) unknown?

Let's try!

Find p, q such that

$s_0 \models \mathbb{P}_{\geq 0.99}(\Diamond \text{SEND})$?



$\exists p, q. \quad x_{s_0} \geq 0.99$

given $x_0 = \ldots$

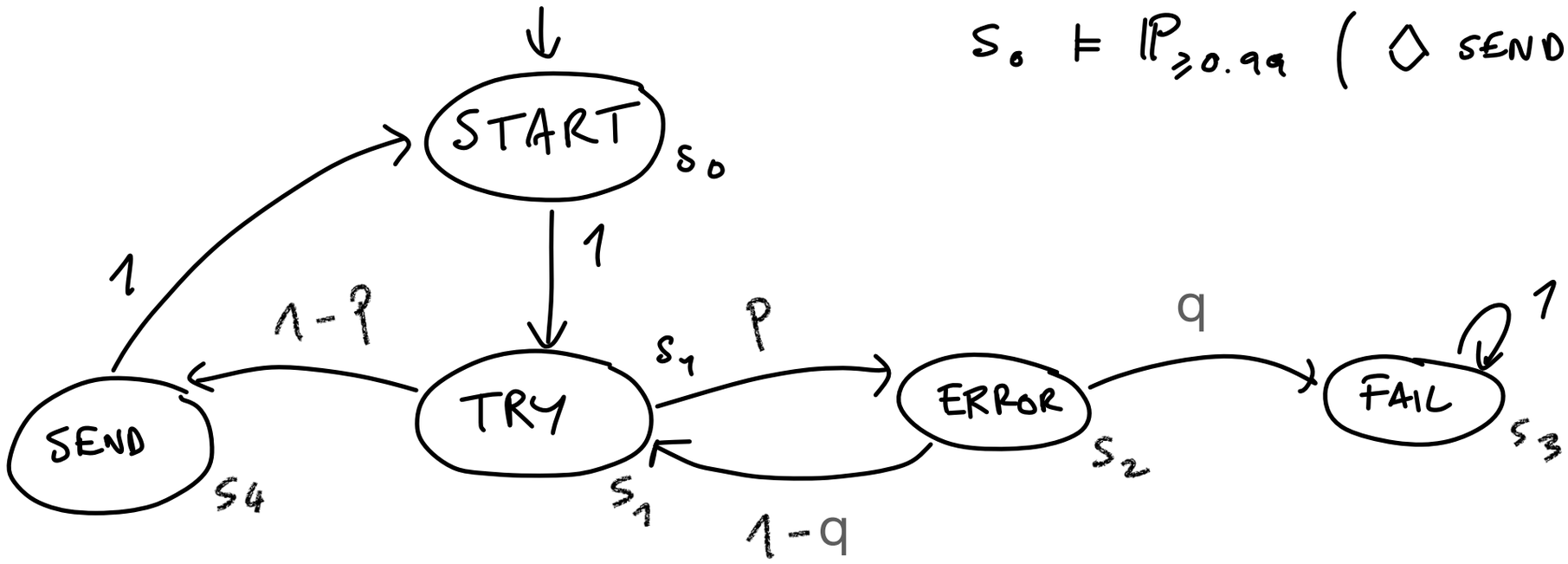
$x_1 = \ldots$

$\vdots$

See file **lossy-channel-parametric.py**

Find p, q such that

$s_0 \models \mathbb{P}_{\geq 0.99}(\Diamond \text{SEND})$?



$\exists p, q. \quad x_{s_0} \geq 0.99$

given $x_0 = \ldots$

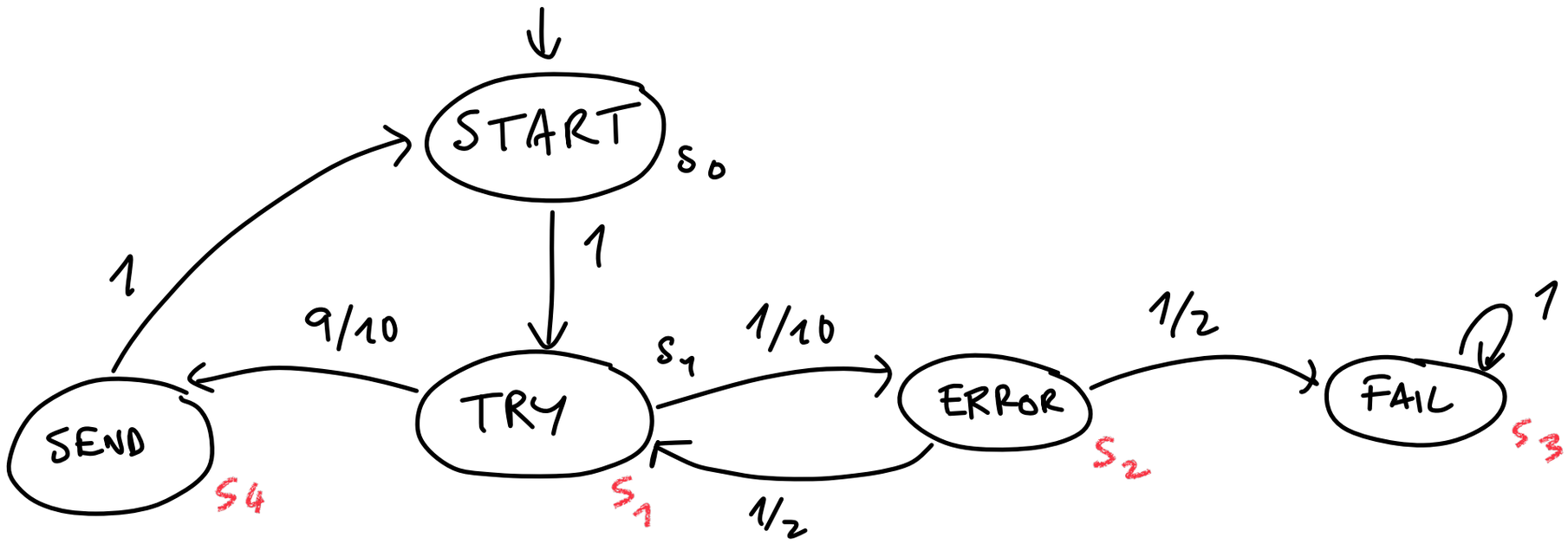
$x_1 = \ldots$

$\vdots$

```
~/Desktop/lecture 08 > python3 lossy-channel-parametric.py
Hoorray! Here is a possible solution:
/0 = [(63/128, 1/2) -> 63/64, else -> 0]
p = 1/2
q = 1/127
x0 = 127/128
x1 = 127/128
x2 = 63/64
x3 = 0
x4 = 1
```

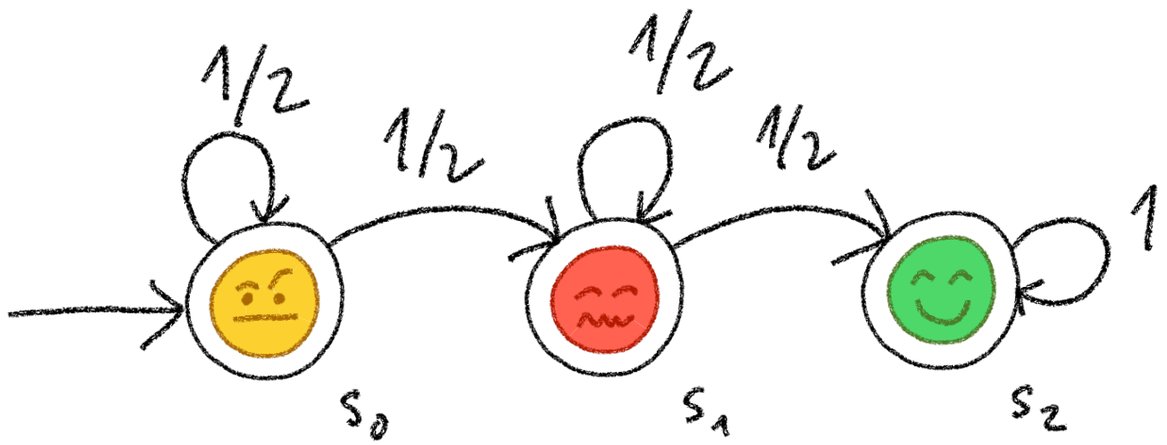
See file **lossy-channel-parametric.py**

MORE BONUS: Expectations



Expected number of errors until the message is sent or we fail?

Expected number of tries until the message is sent or we fail?



Expected number of days until sick?

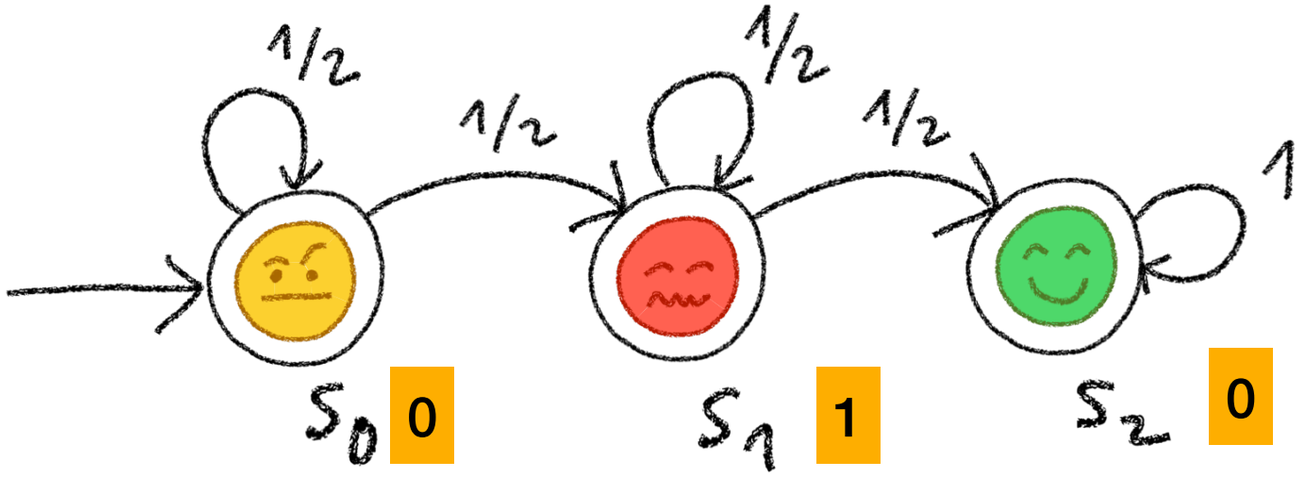
Expected number of sick days sick until recovered?

DTMCs with rewards/costs

Definition 10.69. Markov Reward Model (MRM)

A *Markov reward model* (MRM) is a tuple $(\mathcal{M}, rew)$ with $\mathcal{M}$ a Markov chain with state space S and $rew : S \to \mathbb{N}$ a reward function that assigns to each state $s \in S$ a non-negative integer reward $rew(s)$. ■

Example with rew function counting “sick days”



i.e. rew(s0) = 0, rew(s1) = 1, rew(s2) = 0

Expected reward $ExpRew_s(\Diamond B)$

Formally…

cumulative reward of path fragments until B,

the *cumulative reward* for a finite path $\widehat{\pi} = s_0\, s_1 \ldots s_n$ is defined by

$$rew(\widehat{\pi}) = rew(s_0) + rew(s_1) + \ldots + rew(s_{n-1}).$$

$$ExpRew(s \models \Diamond B) = \sum_{\widehat{\pi}} \mathbf{P}(\widehat{\pi}) \cdot rew(\widehat{\pi})$$

where $\widehat{\pi}$ ranges over all finite paths $s_0 \ldots, s_n$ with $s_n \in B$, $s_0 = s$ and $s_0, \ldots, s_{n-1} \notin B$.

Recursive definition:

$$ExpRew(s \models \Diamond B) = rew(s) \qquad \text{if } s \in B$$

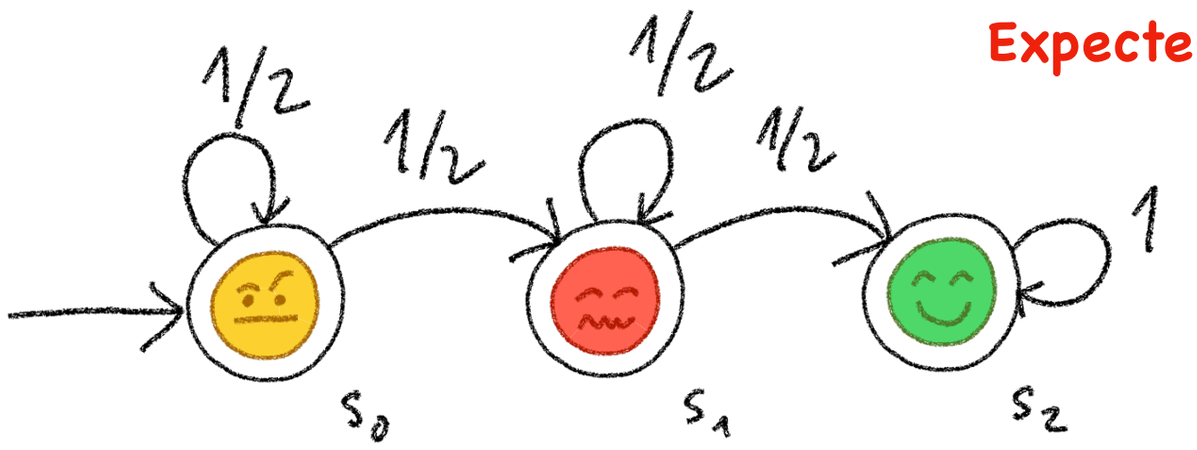
$$ExpRew(s \models \Diamond B) = +\infty \text{ or } 0 \qquad \text{if } s \models \neg \exists \Diamond B$$

$$ExpRew(s \models \Diamond B) = rew(s) + \sum_{s' \in S} \mathbf{P}(s, s') \cdot ExpRew(s' \models \Diamond B) \qquad \text{otherwise}$$

For the sake of simplicity, let's consider models where this does not happen and that all states $Pr_s(\Diamond B) = 1$

(Ask the teacher or check the book to open this dooor:))

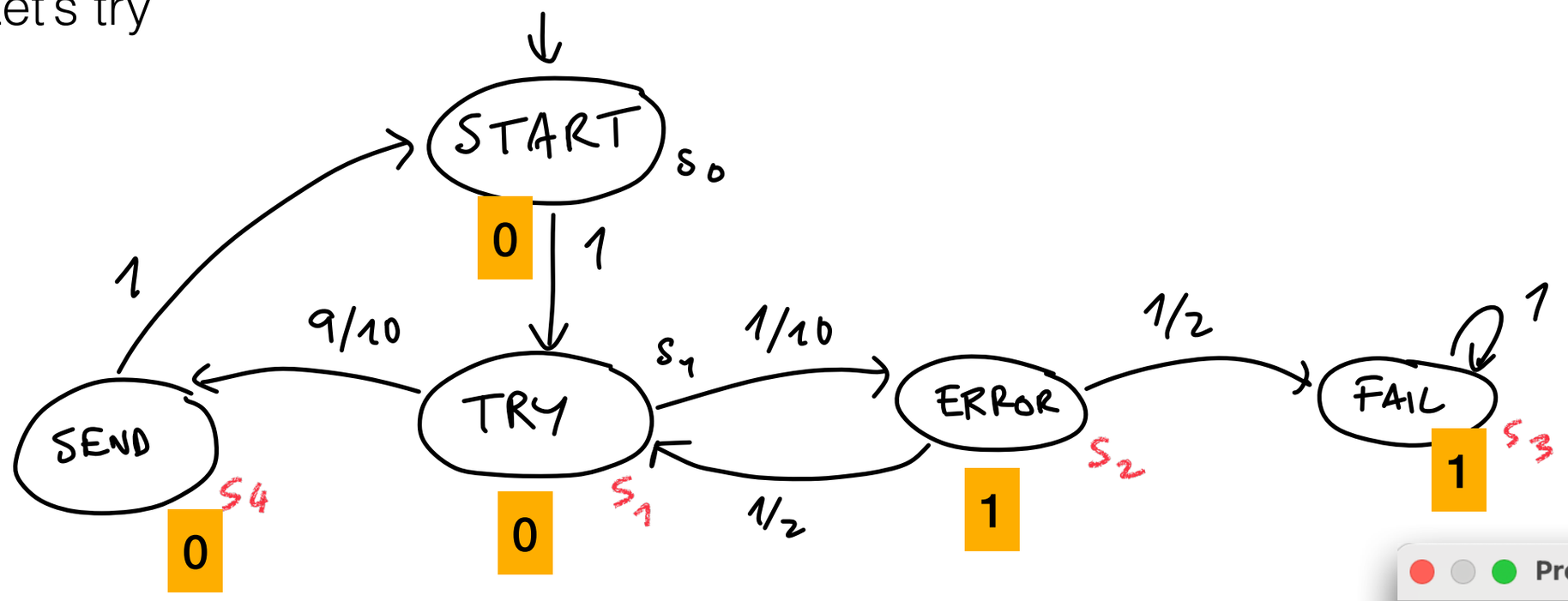
Let's try

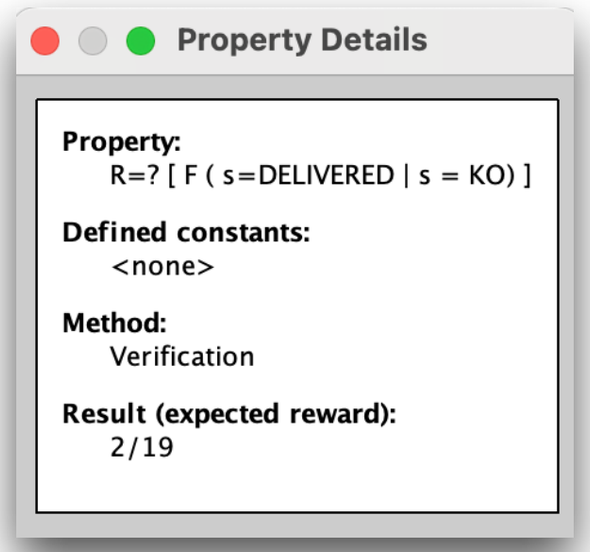


Expected number of days until sick?

Expected number of sick days sick until recovered?

Let's try





Expected number of errors until message send or we fail?

See files **lossy-channel-rewards.prism**
lossy-channel-rewards.pctl
lossy-channel-rewards.py

```
x0 == 0 + x1 ,
x1 == 0 + Q(1,10) * x2 + Q(9,10) * x4 ,
x2 == 1 + Q(1,2) * x1 + Q(1,2) * x4 ,
x3 == 0 ,
x4 == 0
```

```
~/Desktop/lecture 08 > python3 lossy-channel-rewards.py
Hoorray! Here is a possible solution:
x0 = 2/19
x1 = 2/19
x2 = 20/19
x3 = 0
x4 = 0
```

Expected number of tries until….?

(Similar, with reward 1 in state s1, and 0 elsewhere)

Cost/rewards in PRISM

See

https://www.prismmodelchecker.org/manual/ThePRISMLanguage/CostsAndRewards

and:

https://www.prismmodelchecker.org/manual/PropertySpecification/Reward-basedProperties

For a deeper treatment of cost/rewards in probabilistic systems you can check our textbook (section 10.6) or the references indicated in the PRISM manual.

For examples, you can check the PRISM models in the PRISM distribution under /prism-models/

Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\bigcirc B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

Key points of this lecture

Model checking PCTL is done bottom-up (as for CTL)

Model checking PCTL requires to compute probabilities of bounded and unbounded reachability.

Recursive algorithms for bounded reachability (similar to transient distribution).

Algorithms for unbounded reachability based on solving systems of equations (similar to steady states).

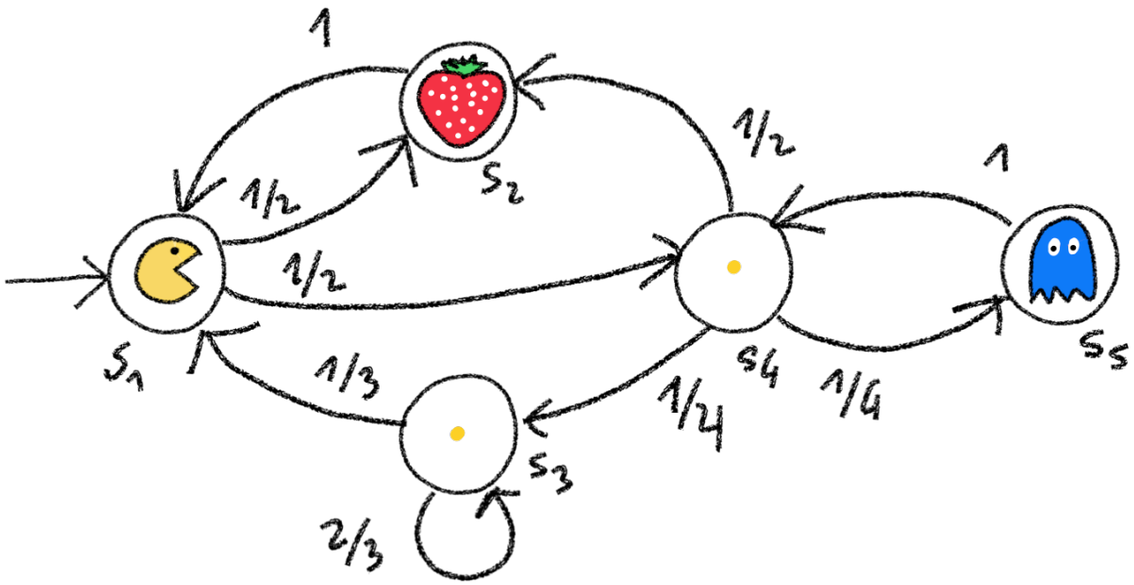
Model Checking PCTL

- PCTL syntax and semantics
- Global algorithm
- Model checking $\mathbb{P}(\bigcirc B)$
- Model checking $\mathbb{P}(\Diamond^{\leq n} B)$
- Model checking $\mathbb{P}(\Diamond B)$
- Model checking $\mathbb{P}(C\ U^{\leq n}\ B)$
- Model checking $\mathbb{P}(C\ U\ B)$
- Alternative algorithms
- Exercises & Homework

APPENDIX: Exercises

Exercise 8.1

Apply the model checking algorithm to determine the set of states that satisfy the following PCTL formulas in the below transition system



a) $\mathbb{P}_{=1}(\bigcirc(\text{🍓} \vee \mathbb{P}_{\geq 1/2}(\bigcirc \text{🍓})))$

b) $\mathbb{P}_{<1/2}(\Diamond \text{👻})$

c) $\mathbb{P}_{>9/10}(\neg \text{👻} \cup \text{🍓})$

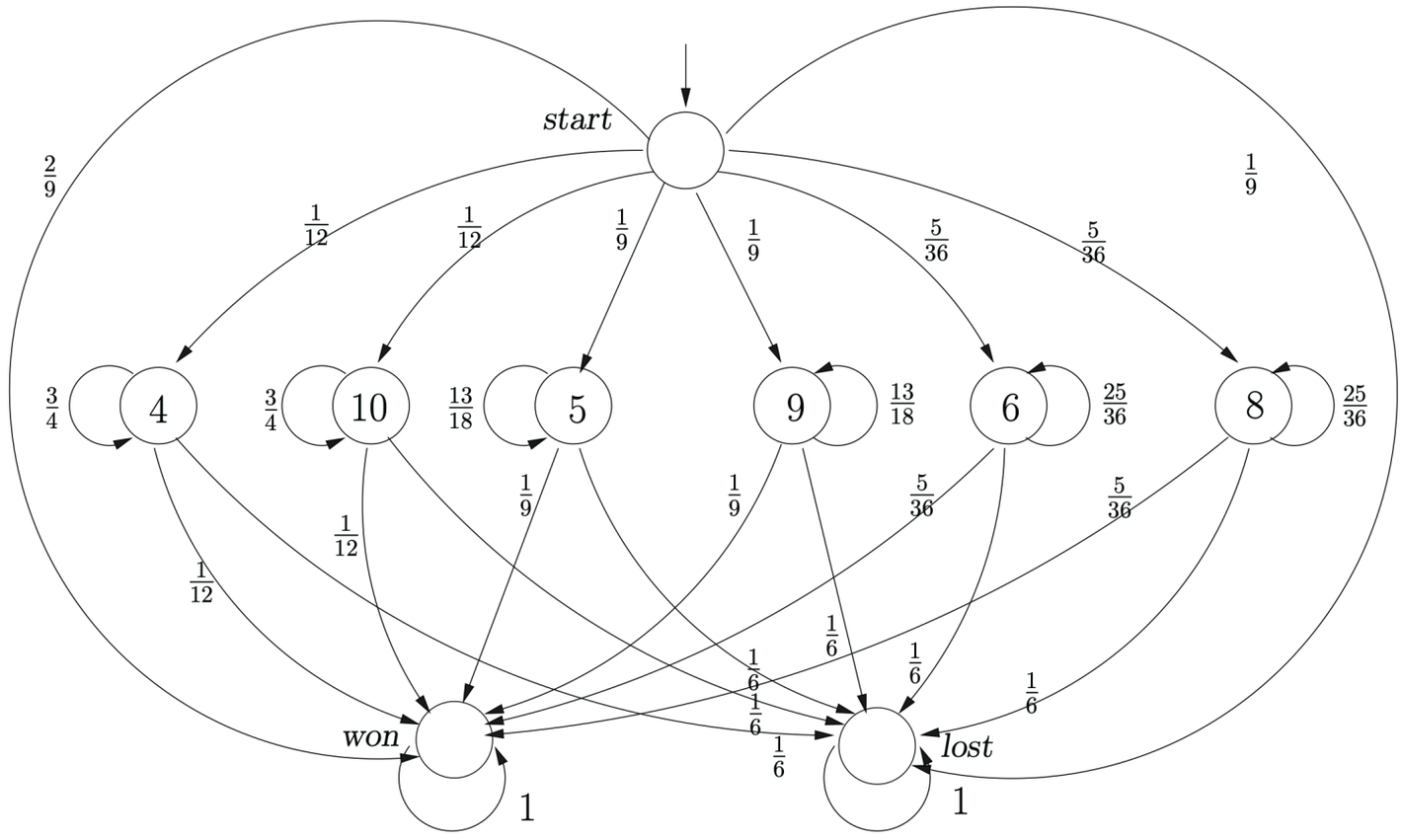
d) $\mathbb{P}_{>3/4}(\Box \neg \text{👻})$

Use Z3 to solve any equation system that may be needed to solve the exercises.

Use PRISM to verify your results. Does the results of PRISM and your results agree?

Exercise 08.2

Apply the model checking algorithm to determine if the probability of winning in the craps game (see book and DMTC below) is above 0.5.

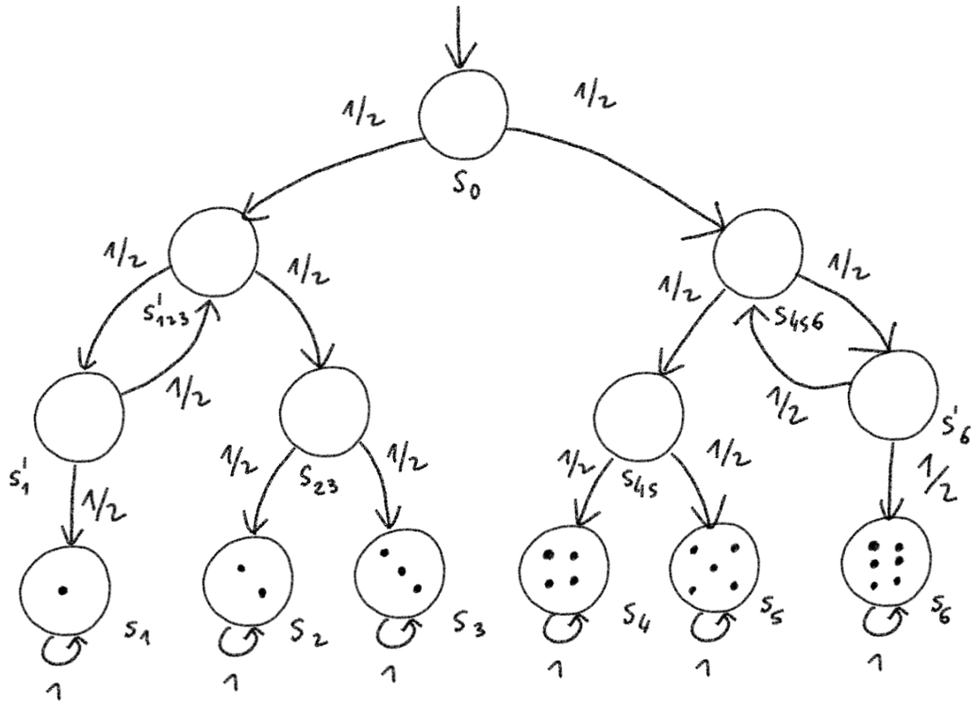


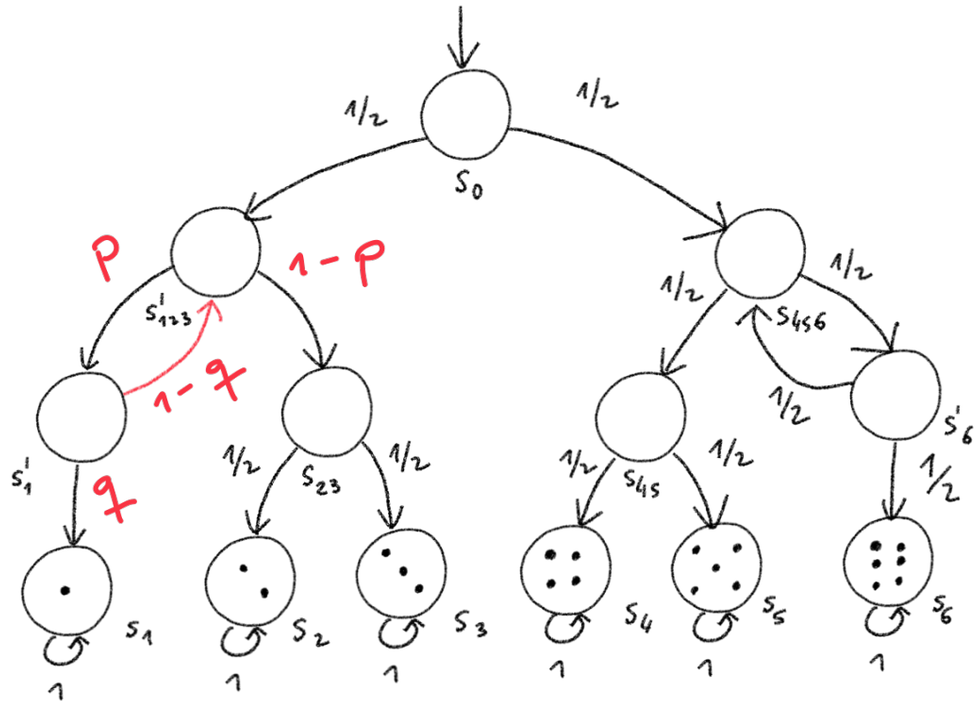
Use Z3 to solve any equation system that may be needed to solve the exercise.

Use PRISM to verify your results. Does the results of PRISM and your results agree?

Exercise 08.3

Consider the dice model on the left: we saw in the lectures (and in the book).





We want to replace the coin used at state S'_123 and S'_1 by two biased (unfair) coins:

- The first one does heads with probability p and tails with probability 1-p
- The first one does heads with probability q and tails with probability 1-q

Can we find such biased coins while still having a fair dice where all numbers are obtained with equal probability?

Exercise 08.4

Consider the IPv4 zeroconf protocol described in the book and sketched below.

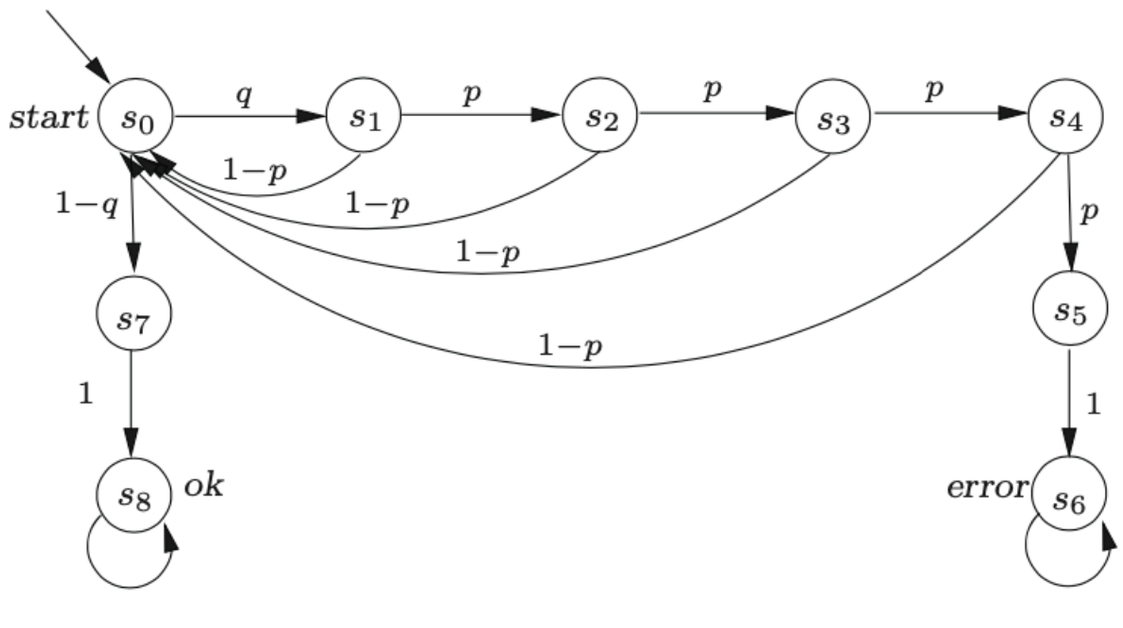


Figure 10.4: Markov chain of the IPv4 zeroconf protocol (for n=4 probes).

Assume that the probability q of choosing the IP address of another device is 1/10, assume that the probability p that a device does not reply to a probe is 1/3. Is the probability of a collision error below 1/100?

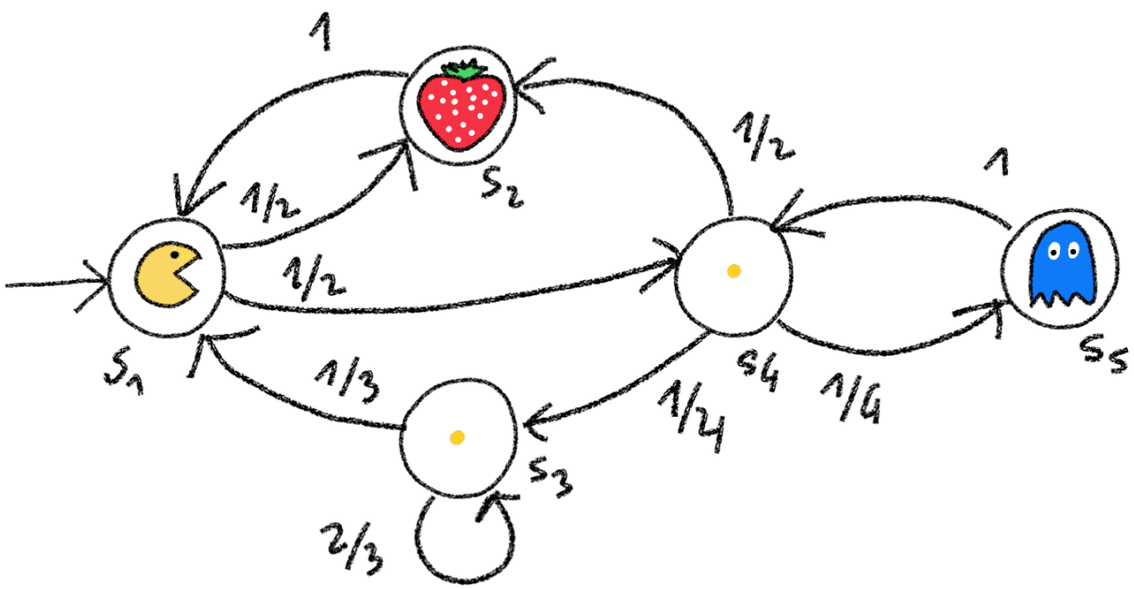
Use Z3 to solve any equation system that may be needed to solve the exercise.

Use PRISM to verify your results. Does the results of PRISM and your results agree?

Solutions

Exercise 08.1

Apply the model checking algorithm to determine the set of states that satisfy the following PCTL formulas in the below transition system



a) $\mathbb{P}_{=1}(\bigcirc(\text{🍓} \vee \mathbb{P}_{\geq 1/2}(\bigcirc \text{🍓})))$

b) $\mathbb{P}_{<1/2}(\Diamond \text{👻})$

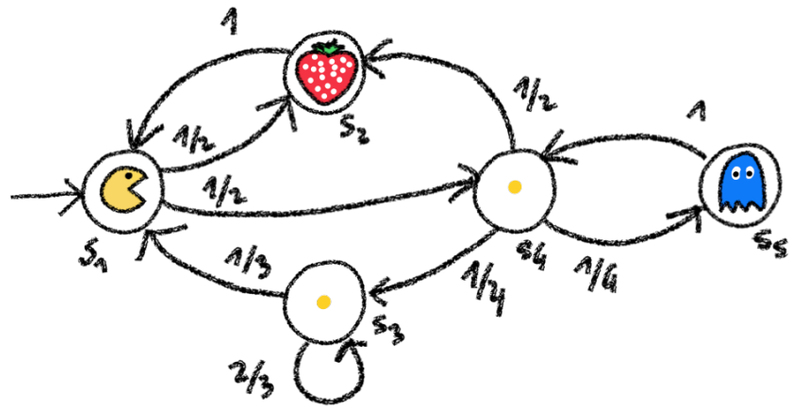
c) $\mathbb{P}_{>9/10}(\neg \text{👻} \cup \text{🍓})$

d) $\mathbb{P}_{>3/4}(\Box \neg \text{👻})$

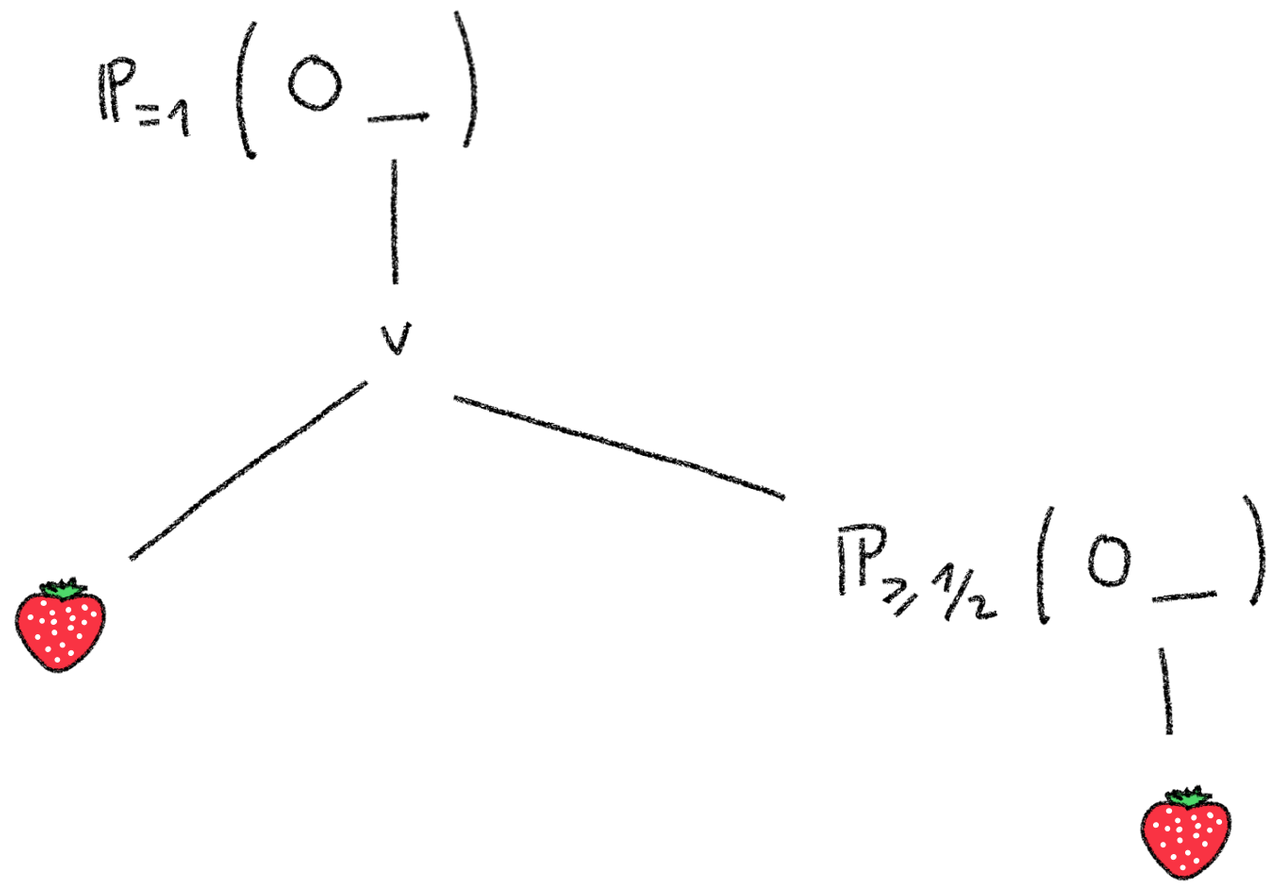
Use Z3 to solve any equation system that may be needed to solve the exercises.

Use PRISM to verify your results. Does the results of PRISM and your results agree?

a) $\mathbb{P}_{=1}\left(\bigcirc\left(\text{🍓} \vee \mathbb{P}_{\geq 1/2}\left(\bigcirc \text{🍓}\right)\right)\right)$



(1) We start depicting the parse tree of the formula



(2) We compute the satisfaction sets of each node/subformula

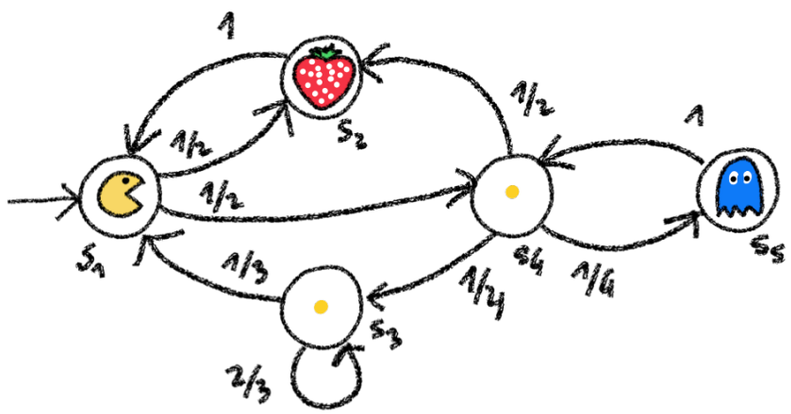
$sat(\mathbb{P}_{=1}(\bigcirc \_)) = \{s_1, s_2, s_5\}$ ← details in next slide

$sat(\vee) = \{s_1, s_2, s_4\}$

$sat(\text{🍓}) = \{s_2\}$

$sat(\mathbb{P}_{\geq 1/2}(\bigcirc \_)) = \{s_1, s_4\}$

$sat(\text{🍓}) = \{s_2\}$



(3) To compute $sat(\mathbb{P}_{\geq 1/2}(\bigcirc \{s_2\}))$ we need to compute

$Pr_s(\bigcirc \{s_2\})$ for all states s:

$Pr_{s_1}(\bigcirc \{s_2\}) = 1/2 \geq 1/2$

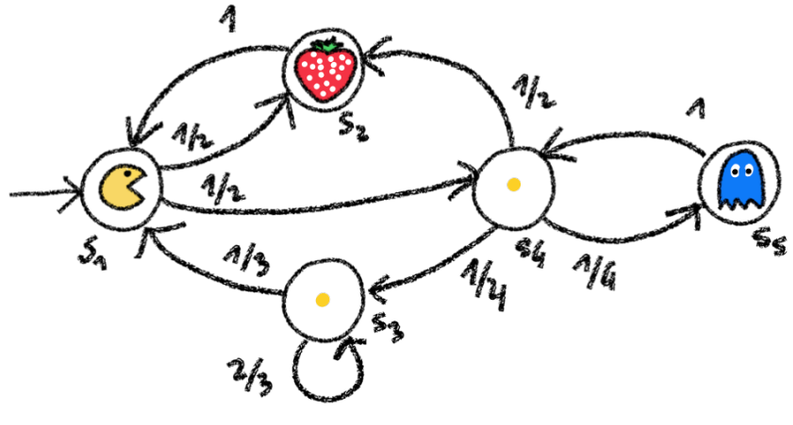
$Pr_{s_2}(\bigcirc \{s_2\}) = 0 \not\geq 1/2$

$Pr_{s_3}(\bigcirc \{s_2\}) = 0 \not\geq 1/2$

$Pr_{s_4}(\bigcirc \{s_2\}) = 1/2 \geq 1/2$

$Pr_{s_5}(\bigcirc \{s_2\}) = 0 \not\geq 1/2$

$sat(\mathbb{P}_{\geq 1/2}(\bigcirc \{s_2\})) = \{s_1, s_4\}$



(3) To compute $sat(\mathbb{P}_{=1}(\bigcirc\{s_1,s_2,s_4\}))$ we need to compute

$Pr_s(\bigcirc\{s_1,s_2,s_4\})$ for all states s:

$Pr_{s_1}(\bigcirc\{s_1,s_2,s_4\}) = 1 \quad = 1$

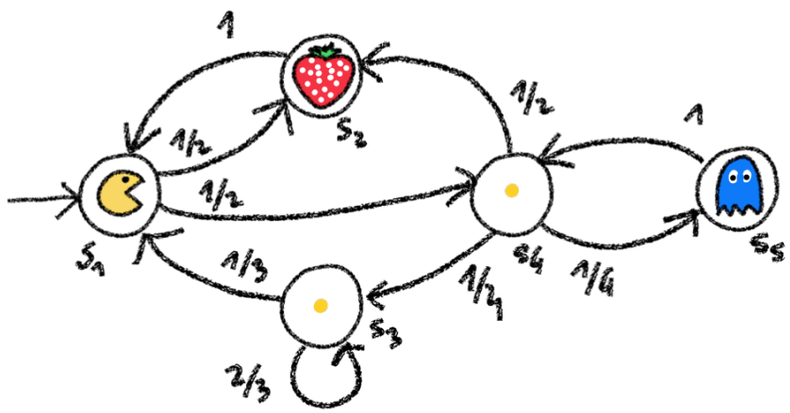
$Pr_{s_2}(\bigcirc\{s_1,s_2,s_4\}) = 1 \quad = 1$

$Pr_{s_3}(\bigcirc\{s_1,s_2,s_4\}) = 1/3 \quad \neq 1$

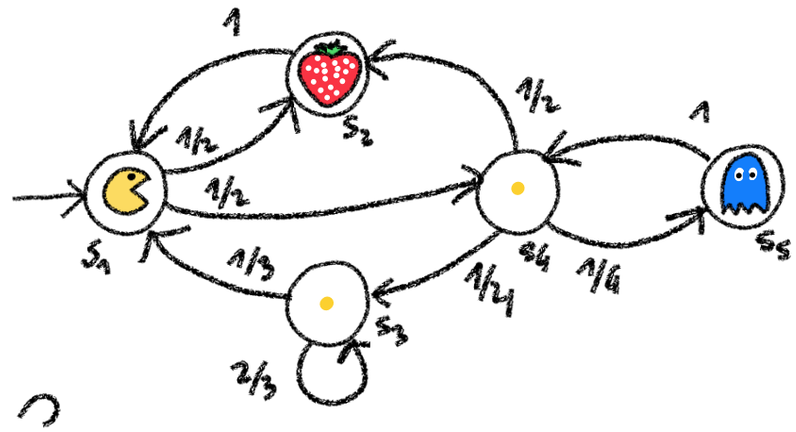
$Pr_{s_4}(\bigcirc\{s_1,s_2,s_4\}) = 1/2 \quad \neq 1$

$Pr_{s_5}(\bigcirc\{s_1,s_2,s_4\}) = 1 \quad = 1$

$sat(\mathbb{P}_{=1}(\bigcirc\{s_1,s_2,s_4\})) = \{s_1, s_2, s_5\}$



b) $\mathbb{P}_{<1/2}(\Diamond$ 👻 $)$



To compute $sat(\mathbb{P}_{<1/2}(\Diamond$ 👻 $))$

we need to compute $Pr_s(\Diamond$ 👻 $)$ for all s

First we write down the system of equations:

$$Pr_{s_1}(\Diamond \text{👻}) = 1/2 \cdot Pr_{s_2}(\Diamond \text{👻}) + 1/2 \cdot Pr_{s_4}(\Diamond \text{👻})$$

$$Pr_{s_2}(\Diamond \text{👻}) = Pr_{s_1}(\Diamond \text{👻})$$

$$Pr_{s_3}(\Diamond \text{👻}) = 2/3 \cdot Pr_{s_3}(\Diamond \text{👻}) + 1/3 \cdot Pr_{s_1}(\Diamond \text{👻})$$

$$Pr_{s_4}(\Diamond \text{👻}) = 1/2 \cdot Pr_{s_2}(\Diamond \text{👻}) + 1/4 \cdot Pr_{s_3}(\Diamond \text{👻}) + 1/4\, Pr_{s_5}(\Diamond \text{👻})$$

$$Pr_{s_5}(\Diamond \text{👻}) = 1 \quad \text{since } s_5 \models \text{👻}$$

Next, we solve it (details below)

$Pr_{s_1}(\Diamond$  $) = 1 \not< 1/2$

$Pr_{s_2}(\Diamond$  $) = 1 \not< 1/2$

$Pr_{s_3}(\Diamond$  $) = 1 \not< 1/2$

$Pr_{s_4}(\Diamond$  $) = 1 \not< 1/2$

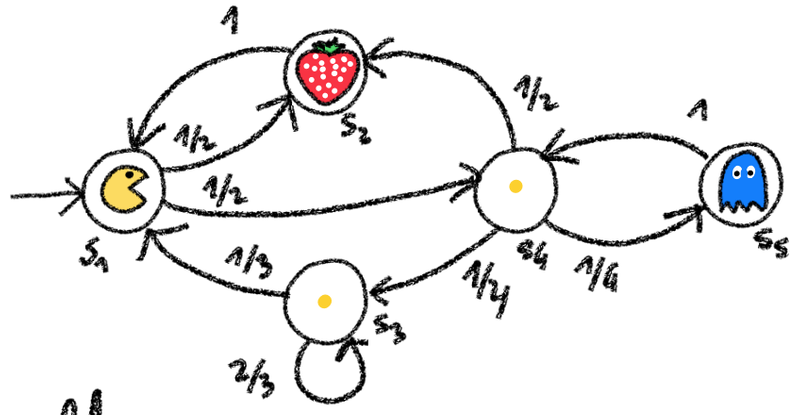
$Pr_{s_5}(\Diamond$  $) = 1 \not< 1/2$

$sat(\mathbb{P}_{<1/2}(\Diamond$  $)) = \emptyset$

See files on Learn

```
~/Desktop/lecture 08 > python3 exercise08.1.py
Hoorray! Here the solution to the equations needed in exercsie 08.1.b:
x1 = 1
x2 = 1
x3 = 1
x4 = 1
x5 = 1
```

c) $\mathbb{P}_{>9/10}(\neg \text{👻} \cup \text{🍓})$



To compute $sat(\mathbb{P}_{>9/10}(\neg \text{👻} \cup \text{🍓}))$

we need to compute $Pr_s(\neg \text{👻} \cup \text{🍓})$ for all s

First we write down the system of equations:

$$Pr_{s_1}(\neg \text{👻} \cup \text{🍓}) = 1/2\, Pr_{s_2}(\neg \text{👻} \cup \text{🍓}) + 1/2\, Pr_{s_4}(\neg \text{👻} \cup \text{🍓})$$

$$Pr_{s_2}(\neg \text{👻} \cup \text{🍓}) = 1 \text{ since } s_2 \models \text{🍓}$$

$$Pr_{s_3}(\neg \text{👻} \cup \text{🍓}) = 1/3\, Pr_{s_1}(\neg \text{👻} \cup \text{🍓}) + 2/3\, Pr_{s_3}(\neg \text{👻} \cup \text{🍓})$$

$$Pr_{s_4}(\neg \text{👻} \cup \text{🍓}) = 1/2\, Pr_{s_2}(\neg \text{👻} \cup \text{🍓}) + 1/4\, Pr_{s_3}(\neg \text{👻} \cup \text{🍓})$$

$$Pr_{s_5}(\neg \text{👻} \cup \text{🍓}) = 0 \text{ since } s_5 \models \text{👻}$$

Next, we solve it (details below)

$Pr_{s_1}(\neg$ 👻 $\cup$ 🍓$) = 6/7 \not> 9/10$

$Pr_{s_2}(\neg$ 👻 $\cup$ 🍓$) = 1 > 9/10$

$Pr_{s_3}(\neg$ 👻 $\cup$ 🍓$) = 6/7 \not> 9/10$

$Pr_{s_4}(\neg$ 👻 $\cup$ 🍓$) = 4/7 \not> 9/10$

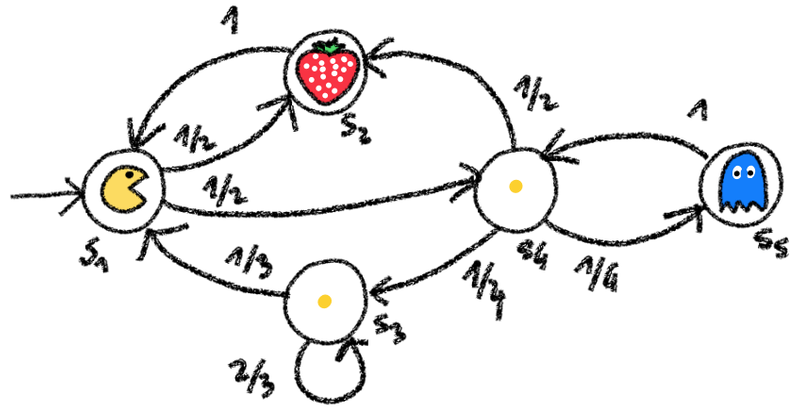
$Pr_{s_5}(\neg$ 👻 $\cup$ 🍓$) = 0 \not> 9/10$

$sat(\mathbb{P}_{>9/10}(\neg$ 👻 $\cup$ 🍓$)) = \{ s_2 \}$

See files on Learn

```
Hoorray! Here the solution to the equations needed in exercsie 08.1.c:
x1 = 6/7
x2 = 1
x3 = 6/7
x4 = 5/7
x5 = 0
```

d) $\mathbb{P}_{>3/4}(\Box\, \neg$ 👻 $)$



Since we don't have an algorithm for this case we need to transform it first into "pure" PCTL

$$\mathbb{P}_{>3/4}(\Box\, \neg \text{👻}) = \mathbb{P}_{\leq 1/4}(\Diamond\, \neg\, \neg \text{👻}) = \mathbb{P}_{\leq 1/4}(\Diamond \text{👻})$$

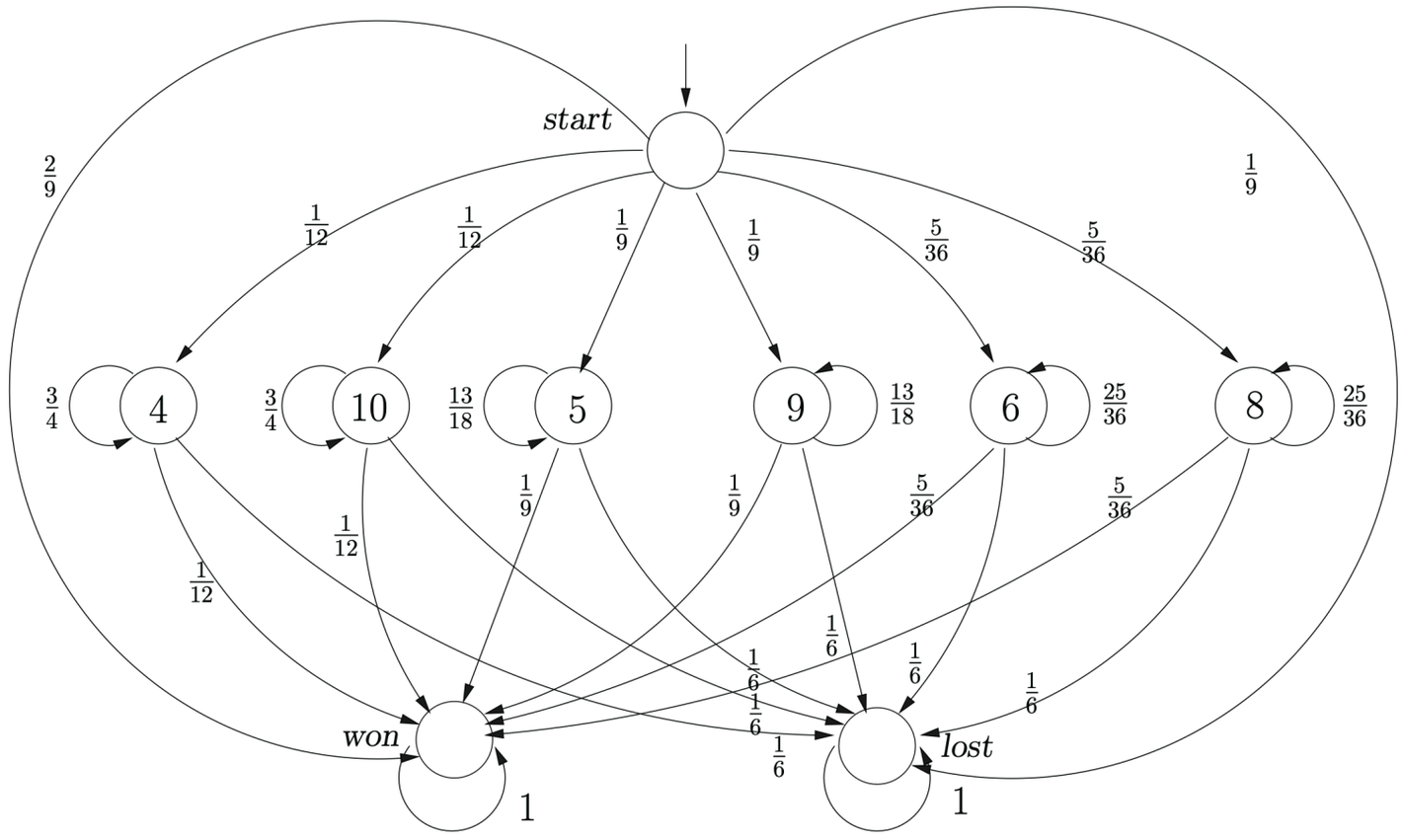
$\Downarrow$

We saw in (b) that $Pr_s(\Diamond$ 👻 $) = 1$ for all states, hence

$$sat(\mathbb{P}_{>3/4}(\Box\, \neg \text{👻})) = \emptyset$$

Exercise 08.2

Apply the model checking algorithm to determine if the probability of winning in the craps game (see book and DMTC below) is above 0.5.



Use Z3 to solve any equation system that may be needed to solve the exercise.

Use PRISM to verify your results. Does the results of PRISM and your results agree?

Exercise 08.2

We provide here directly the solution obtained by Z3 for the equations that need to be computed to determine the probability of winning:

See files on Learn

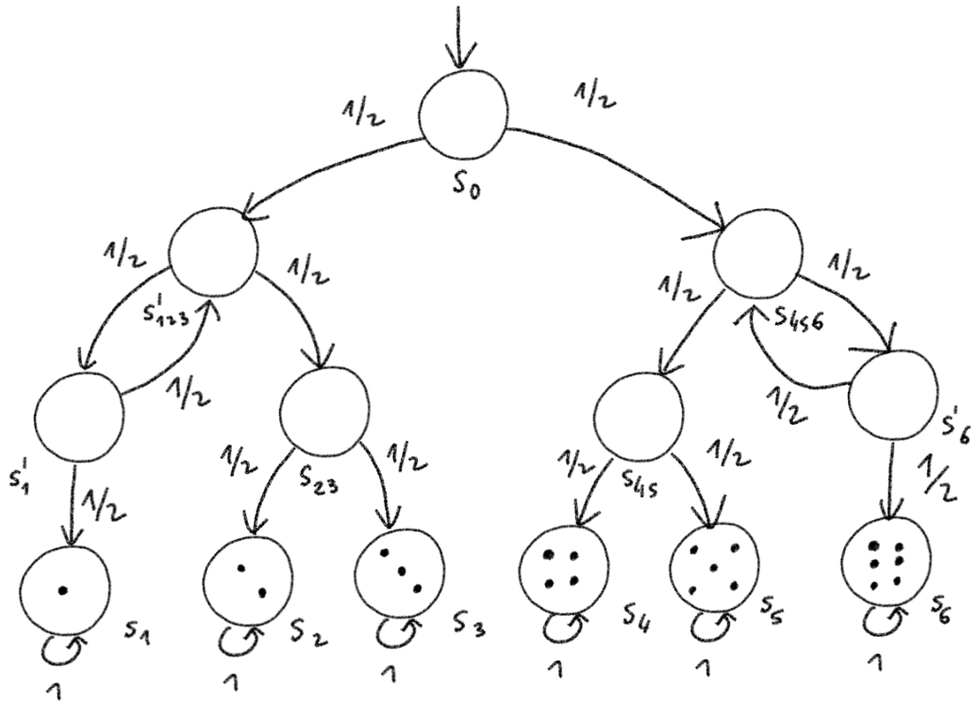
```
dice = Solver()
dice.add(
  # Reachability probabilty equations
  xSTART == Q(1,12)*x4 + Q(1,12)*x10 + Q(1,9)*x5 + Q(1,9)*x9
          + Q(5,36)*x6 + Q(5,36)*x8 + Q(2,9)*xWON + Q(1,9)*xLOST ,
  x4 == Q(3,4)*x4 + Q(1,12)*xWON + Q(1,6)*xLOST ,
  x10 == Q(3,4)*x10 + Q(1,12)*xWON + Q(1,6)*xLOST ,
  x5 == Q(13,18)*x5 + Q(1,9)*xWON + Q(1,6)*xLOST ,
  x9 == Q(13,18)*x9 + Q(1,9)*xWON + Q(1,6)*xLOST ,
  x6 == Q(25,36)*x6 + Q(5,36)*xWON + Q(1,6)*xLOST ,
  x8 == Q(25,36)*x8 + Q(5,36)*xWON + Q(1,6)*xL
  xWON == 1 ,
  xLOST == 0
)
```

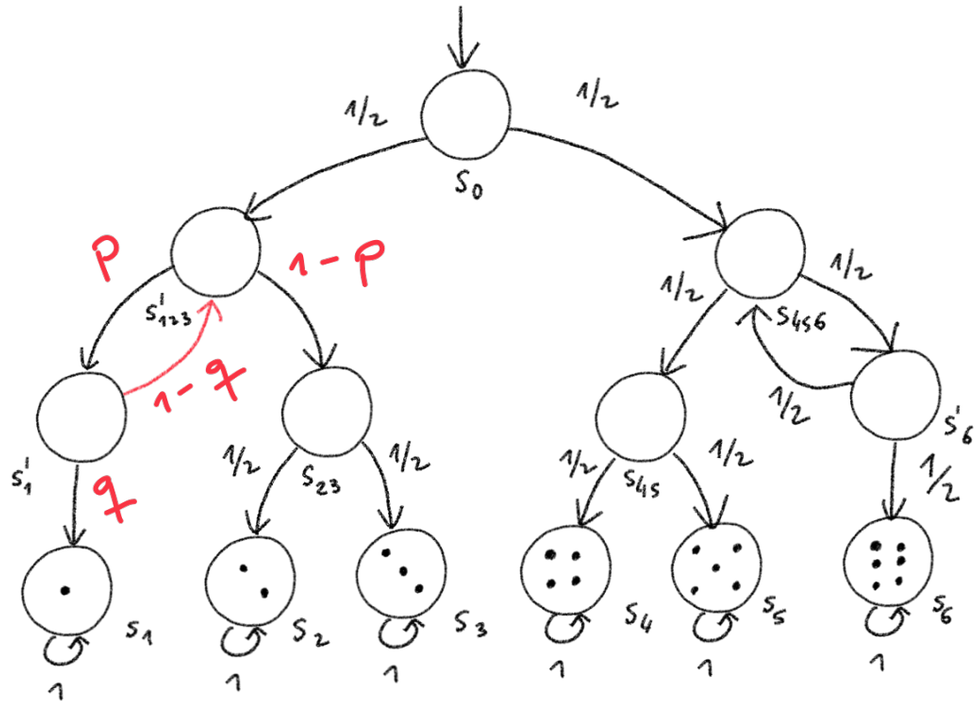
```
~/Desktop/lecture 08 > python3 exercise08.2.py
Hoorray! Here are the probabilities in the game:
x10 = 1/3
x4 = 1/3
x5 = 2/5
x6 = 5/11
x8 = 5/11
x9 = 2/5
xLOST = 0
xSTART = 244/495
xWON = 1
```

Hence, the answer is NO, the probability of winning is not above 50%.

Exercise 08.3

Consider the dice model on the left: we saw in the lectures (and in the book).





We want to replace the coin used at state S'_123 and S'_1 by two biased (unfair) coins:

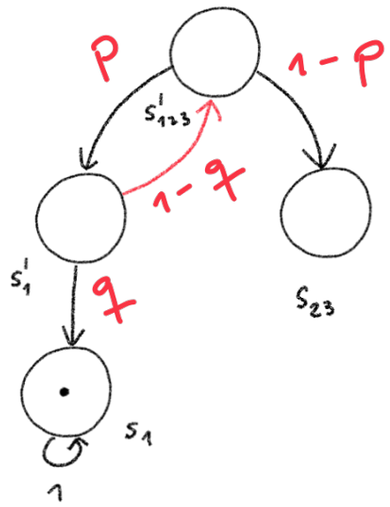
- The first one does heads with probability p and tails with probability 1-p
- The first one does heads with probability q and tails with probability 1-q

Can we find such biased coins while still having a fair dice where all numbers are obtained with equal probability?

Exercise 08.3

We can encode the problem as an SMT problem.

For simplicity we focus on this sub-DTMC to solve he sub-problem $\mathbb{P}_{=1/3}(\Diamond \bullet)$



Which requires that we solve these system of equations:

```
dice = Solver()
dice.add(
  # Reachability probabilty equations
  x123prime == p*x1prime + (1-p)*x23,
  x1prime == (1-q)*x123prime + q*x1,
  x23 == 0 ,
  x1 == 1 ,
  # Coins are biased/unfair
  p != Q(1,2),
  q != Q(1,2),
  p > 0 , p <1 , q > 0 , q < 1 ,
  # We want the probability of reaching s1
  # from s123prime to be 1/3
  x123prime == Q(1,3)
)
```

See files on Learn

```
p = 3/4
q = 1/6
x1 = 1
x123prime = 1/3
x1prime = 4/9
x23 = 0
```

The answer is YES, we can find unfair coins to simulate a fair dice.

Exercise 08.4

Consider the IPv4 zeroconf protocol described in the book and sketched below.

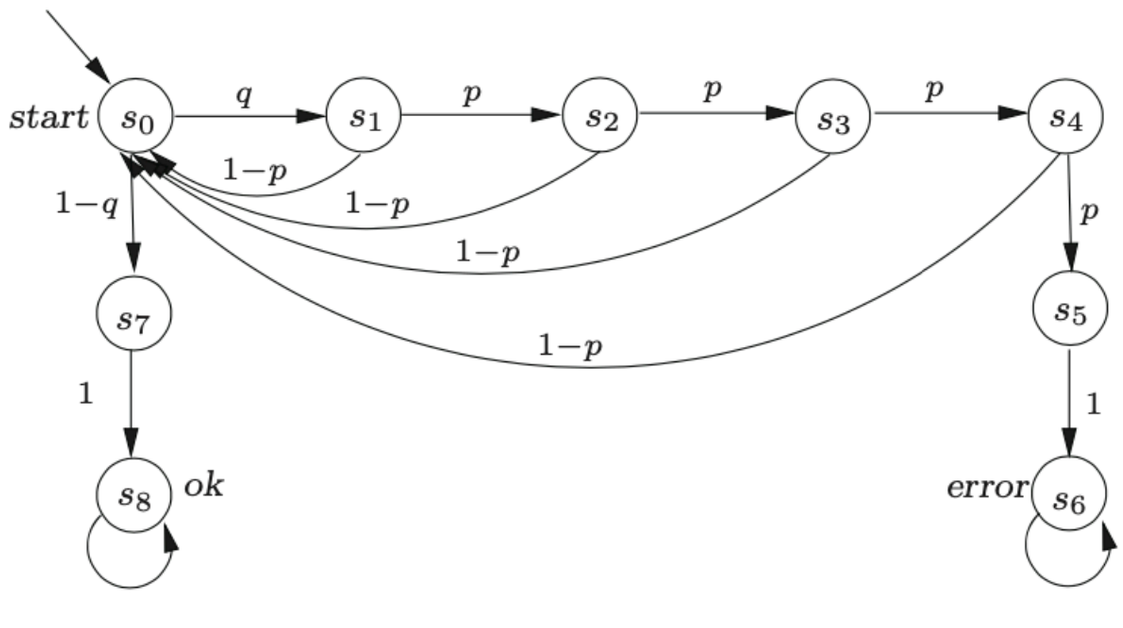


Figure 10.4: Markov chain of the IPv4 zeroconf protocol (for n=4 probes).

Assume that the probability q of choosing the IP address of another device is 1/10, assume that the probability p that a device does not reply to a probe is 1/3. Is the probability of a collision error below 1/1000?

Use Z3 to solve any equation system that may be needed to solve the exercise.

Use PRISM to verify your results. Does the results of PRISM and your results agree?

Exercise 08.4

Here is one way of modelling the needed system of equations in Z3 (omitting details)

See files on Learn

```
ipv4 = Solver()
ipv4.add(
  # Concrete values for p and q
  p == Q(1,3) ,
  q == Q(1,10) ,
  # Reachability probabilty equations

  x0 == q*x1 + (1-q)*x7,
  x1 == p*x2 + (1-p)*x0,
  x2 == p*x3 + (1-p)*x0,
  x3 == p*x4 + (1-p)*x0,
  x4 == p*x5 + (1-p)*x0,
  x5 == x6,
  x6 == 1,
  x7 == x8,
  x8 == 0
)
```

One obtains the following solution

```
~/Desktop/lecture 08 > python3 exercise08.4.py
Hoorray! Here is a possible solution:
p = 1/3
q = 1/10
x0 = 1/730
x1 = 1/73
x2 = 14/365
x3 = 41/365
x4 = 122/365
x5 = 1
x6 = 1
x7 = 0
x8 = 0
```

Reaching the collision error has probability higher than 1/1000!

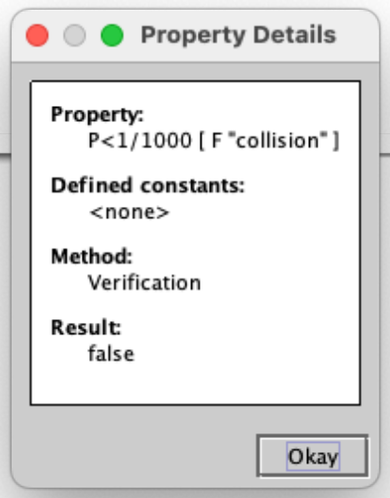
So the answer is NO :(.

Exercise 08.4

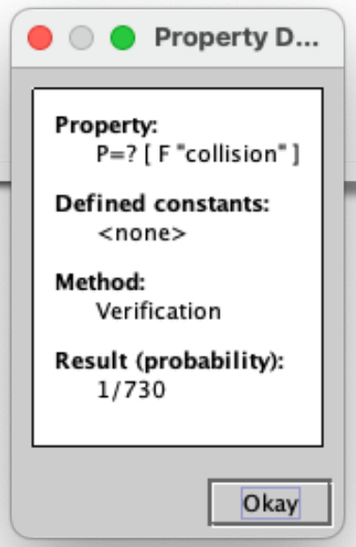
Here is one way of modelling the IPv4 zeroconf protocol in PRISM (more general than needed)

```
dtmc

const MPROBES = 4;
const TOTAL_ADDRESSES = 1000;                    // Total number of addresses
const TAKEN_ADDRESSES = 100;                     // Already taken addresses

formula p = 1/3;                                 // Probability that a probe gets no response
formula q = (TAKEN_ADDRESSES/TOTAL_ADDRESSES);   // Probability to pick an already assigned address

const CONFIGURING = 0;
const SUCCEEDING  = 1;
const FAILING     = 2;
const SUCCESS     = 3;
const FAILURE     = 4;

module ZEROCONF

  s : [0..4] init 0;
  probes : [0..MPROBES] init 0;

  [] (s=CONFIGURING & probes=0) ->     q : (probes'=1)                // Bad luck: picked someone else's address
                                + 1.0-q : (s'=SUCCEEDING);            // Got lucky and picked a free adsress

  [] (s=CONFIGURING & probes>0 & probes<MPROBES) ->
                                        p : (probes'=probes+1)        // Bad luck: address ownwer did not respond, try once more...
                                + 1.0 - p : (probes'=0);              // Address owner replies and we try to pick another address

  [] (s=CONFIGURING & probes=MPROBES) ->       p : (s'=FAILING)       // Bad luck: started using the address and a collision happened
                                + 1.0 - p : (probes'=0);              // Address owner replies and we try to pick another address

  [] (s=SUCCEEDING) -> 1.0 : (s'=SUCCESS);
  [] (s=FAILING) -> 1.0 : (s'=FAILURE);
  [] (s=SUCCESS | s=FAILURE) -> 1.0 : true ;

endmodule

label "collision" = (s = FAILURE);
label "no_collision" = (s = SUCCESS);
```

And the result



In detail…



Same as in Z3 :)