Probabilistic Model Checking

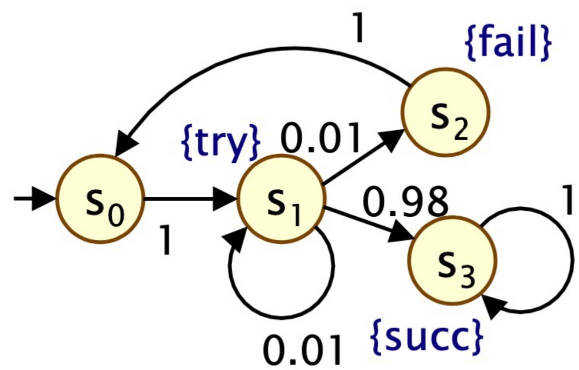
Wishnu Prasetya

wishnu@cs.uu.nl
www.cs.uu.nl/docs/vakken/pv

additional slides DTMC

Probability of taking a path or a set of paths

A "DTMC" :

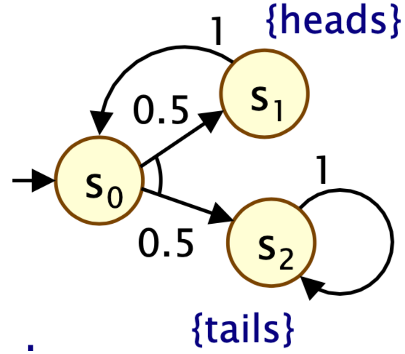


- Consider a path ω e.g. $s_0,s_1,s_2,s_0.$ The probability that the system follows this path when executed with the starting state s_0 is denoted by $P_{s0}(\omega)$. Or simply $P(\omega)$ if it is clear which s_0 is meant. It is the product of the probability of each transition in ω.

  Example: for the above ω, $P(\omega) = 1 * 0.01 * 1 = 0.01$

- For a **set of of paths** U (starting from s0), the probability that the system's execution follows **one of** the paths in U, denoted by P(U), is $\sum_{\omega \in U} P(\omega)$.

Probability of taking a path or a set of paths

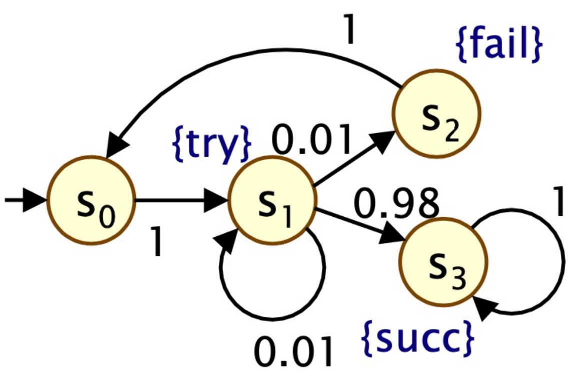


Example: consider U = the set of paths that ends in s_2. Note that U is infinite: U = { 02, 0102, 010102, … }. But we can calculate P(U).

$$P(U) = 0.5 + 0.52 + 0.53 + \ldots = \sum_{k\geq 0} 0.5^k$$

$$= 0.5 * \frac{1}{1 - 0.5} = 1$$

Probability Matrix Representation



P:

	s0	s1	s2	s3
s0	0	1	0	0
s1	0	0.01	0.01	0.98
s2	1	0	0	0
s3	0	0	0	1

$P_{i,k}$ = the value at the i-th row and k-th column. It specifies the probability of taking the transition $s_i \rightarrow s_k$, if we are now at s_i.

For example the circle red value above is $P_{1,2}$, specifying the probability of taking the transition from s_1 to s_2 (check the picture), which is 0.01.

Basic Operations on Probability Matrix

- Multiplying P with itself: P^n
- Multiplying a vector with P: $u \times P$
- Multiplying P with a vector: $P \times v$

P^n

- $P^0 = I$ (identity matrix)
  $P^{n+1} = P \times P^n$
- $P^n_{i,k}$ is the probability of ending up in state s_k in n-steps, given we start in the state s_i.
- For example, wirth the previous P, let's look at P^2:

0	1	0	0
0	0.01	0.01	0.98
1	0	0	0
0	0	0	1

×

0	1	0	0
0	0.01	0.01	0.98
1	0	0	0
0	0	0	1

→

?	?	0.01	?
?	?	?	?
?	?	?	?
?	?	?	?

$$P^2_{0,2} = (P \times P)_{0,2}$$
$$= P_{0,0} * P_{0,2} + P_{0,1} * P_{1,2} + P_{0,2} * P_{2,2} + P_{0,3} * P_{3,2}$$

Probabity distribution of the next state, given the current distribution

- A **probability distribution** of the current state is the probability of currently being in various states. It can be given by a vector of size K, if K is the number of possible states. E.g. if $u = [\,0\,,\,0.5\,,\,0.5,\,0\,]$ is the probability distribution of the current state, it says e.g. that there is 0.5 probability that currently we are in the state s_1, but 0 probability that we are in the state s_0.
- The product $u \times P$ (we often simply write it as uP) gives a new vector u' of size K, that gives us the probability distribution of the next state.

$$\underbrace{\begin{bmatrix} 0 & 0.5 & 0.5 & 0 \end{bmatrix}}_{u} \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Rightarrow \underbrace{\begin{bmatrix} 0.5 & 0.005 & 0.005 & 0.49 \end{bmatrix}}_{u'}$$

e.g. $u'_1 = u \bullet$ the green collumn (dot product)

$$= u_0 * P_{0,1} + u_1 * P_{1,1} + u_2 * P_{2,1} + u_3 * P_{3,1}$$

Probabilty vector

- Sometimes we also want to know what the probability to end up in state, say, s_1 or s_2 as the **next** state, if we start in the state s1.
- We can represent "end up in either s_1 or s_2" with a vector $v = [0,1,1,0]$.
- Let v^t is the *transpose* of v. The product $P \times v^t$ gives a w such that w is a (transposed) vector, where w_i is the probabilty to end up in one of the states specified in v, if we start in s_i.

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \underbrace{\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}}_{v^t} \Rightarrow \underbrace{\begin{bmatrix} 1 \\ 0.02 \\ 0 \\ 0 \end{bmatrix}}_{w}$$

e.g. w_1 = the green row • v^t (dot product)

$= P_{1,0} * v_0 + P_{1,1} * v_1 + P_{1,2} * v_2 + P_{1,3} * v_3$

Probability vector

- Prob(φ) (notice the underscore) is a probility vector e.g. w =

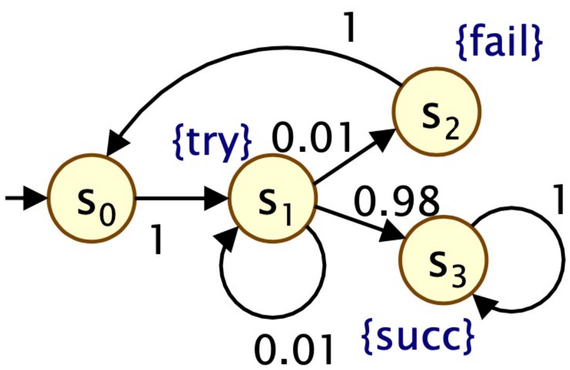
1
0.02
0
0

  such that the i-th element tells us what the probability that the system would behave as φ if executed in state s_i.

- Example: the above w (blue) happens to be equal to Prob(**X**(try ∨ fail)).

- This notation Prob will be used later when we discuss model checking of probabilistic-CTL.

The construction of P’ for bounded Until

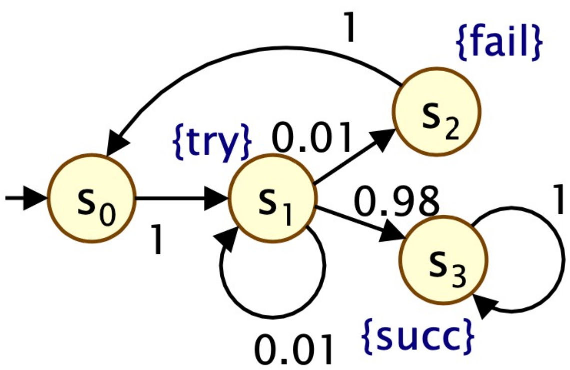


0	1	0	0
0	0.01	0.01	0.98
1	0	0	0
0	0	0	1

The probability matric P of the DTMC on the left.

- Consider as an example to check whether the DTMC satisfies $P_{>0.99}$[try ∨ ¬fail **U**$^{\leq 2}$ succ].
  1. Calculate first the probability vector Prob[try ∨ ¬fail **U**$^{\leq 2}$ succ].
  2. From there you can calculate the set Sat($P_{>0.99}$[try ∨ ¬fail **U**$^{\leq 2}$ succ]).
  3. **If the initial state s_0 is in** the blue Sat-set then the property $P_{>0.99}$[try ∨ ¬fail **U**$^{\leq 2}$ succ holds on the DTMC.

The construction of P’ for bounded Until

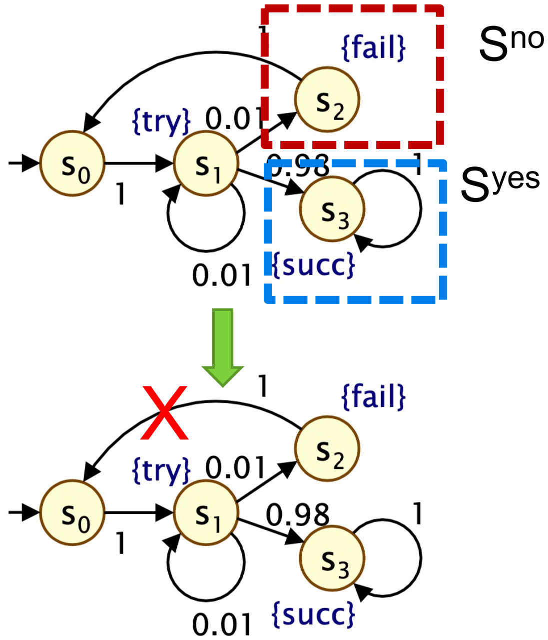


0	1	0	0
0	0.01	0.01	0.98
1	0	0	0
0	0	0	1

The probability matric P of the DTMC on the left.

- To calculate the probability vector Prob[try ∨ ¬fail **U**$^{\leq 2}$ succ], we would like to use the matrix P above, however it will also ”contain” transitions that cause you to break the green-property. So the idea is to use a “modified” matrix P’.
- We pre-calculate first the S^{yes} = Sat(succ) = {s3}. On all states in S^{yes}, you have the green property immediately (in 0 step).
- We pre-calculate S^{no}, we take S^{no} = Sat(¬ (try ∨ ¬fail) ∧ ¬ succ) = { s2 }. Executions starting from S^{no} won’t satisfy your green-property above,

The construction of P' for bounded Until



P :

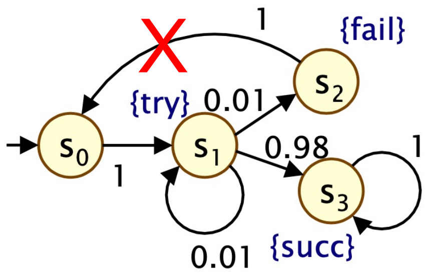
0	1	0	0
0	0.01	0.01	0.98
1	0	0	0
0	0	0	1

P' :

0	1	0	0
0	0.01	0.01	0.98
0	0	0	0
0	0	0	1

1. We remove outgoing arrows from the states in S^{no} and S^{yes}.
2. We keep all arrows that go out from states which are **not** in S^{no} nor S^{yes}.
3. We add a self-loop $s \rightarrow s$ with probability 1 for any state s in S^{yes}.

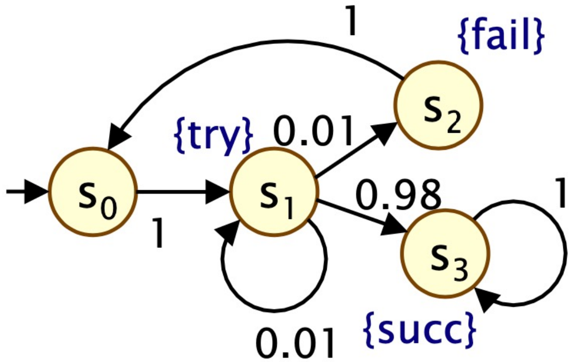
Using P’ for bounded Until



$$P' : \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- We now use P’ to iteratively calcualte Prob[try ∨ ¬fail $\mathbf{U}^{\le 2}$ succ]
- From S^{yes} you know that Prob[try ∨ ¬fail $\mathbf{U}^{\le 0}$ succ] = $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$
- Prob[try ∨ ¬fail $\mathbf{U}^{\le 1}$ succ] = P’ × $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.98 \\ 0 \\ 1 \end{pmatrix}$
- Prob[try ∨ ¬fail $\mathbf{U}^{\le 2}$ succ] = P’ × $\begin{pmatrix} 0 \\ 0.98 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.98 \\ 0.9898 \\ 0 \\ 1 \end{pmatrix}$

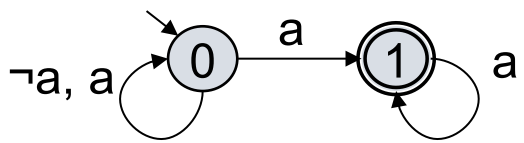
So, does the property hold?



- We have calculated <u>Prob</u>[try ∨ ¬fail $\mathbf{U}^{\leq 2}$ succ] = [0.98, 0.9898, 0, 1]
- So, the set Sat($P_{>0.99}$[try ∨ ¬fail $\mathbf{U}^{\leq 2}$ succ]) = $\{s_3\}$
- So we conclude that the DTMC does **not** satisfy the claimed property $P_{>0.99}$[try ∨ ¬fail $\mathbf{U}^{\leq 2}$ succ].

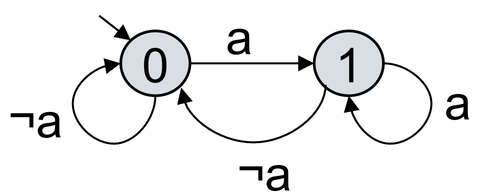
Buchi vs Rabin

- Consider the LTL property $\Diamond\Box a$. This can be described by this Buchi automaton, wirth {1} as the accepting state:



Notice that this Buchi is non-deterministic. As such, we can use it for model checking on a probabilistic model such as DTMC.

- We can however represent the property with a deterministic Rabin automaton, with the pair ({0}, {1}) as its accepting condition.



additional slides MDP

Calculating pmin(s, φ_1 **U** φ_2)

pmin(s, φ) = the **minimum** probability for having executions starting from 2 satisfying φ, regardless the adversary.

To calculate pmin(s, φ_1 **U** φ_2) :

1. Calculate first the sets S^{yes} and S^{no}.
   - S^{yes} = { s | P(s, φ_1 U φ_2) ≥ 1, for **all** adversaries } → with algorithm **Prob1A**.
   - S^{no} = { s | P(s, φ_1 U φ_2) ≤ 0, for **some** adversary } → with algorithm **Prob0E**.
2. For any state s in S^{yes}, we then know that pmin(s, φ_1 **U** φ_2) ≥ 1.
3. For any state s in S^{no} we have pmin(s, φ_1 **U** φ_2) ≤ 0.
4. We then proceed with calculating the pmin for the remaining states (which are not in S^{yes} nor S^{no}).

Algorithm **Prob0E**

- The algorithm below first calculates the set R of all states s satisfying E[s, φ_1 U φ_2], regardless the adversary. So, for any state in R, and for any adversary Prob(s, φ_1 U φ_2) > 0.
- S^{no} is just complement S/R.
- Sat(φ_1) and Sat(φ_2) and in the paremeters are the set of states on which φ_1 and φ_2 respectively hold.

$\textsc{Prob0E}(Sat(\phi_1), Sat(\phi_2))$

$R := Sat(\phi_2)$
$done :=$ **false**
while $(done =$ **false**$)$
  $R' := R \cup \{s \in Sat(\phi_1) \mid \forall \mu \in Steps(s) \,.\, \exists s' \in R \,.\, \mu(s') > 0\}$
  if $(R' = R)$ **then** $done :=$ **true**
  $R := R'$
endwhile
return $S \backslash R$

Prob1A

- Calculate first the set F of states from where we have a path passing exclusively through states satisfying $\varphi_1 \wedge \neg\varphi_2$ and ends in S^{no}, under **some** adversary.
  By definition this F also includes S^{no}.
- So any state s in F has Prob(s, φ_1 U φ_2) < 1, for some adversary. In other words pmin(s, φ_1 U φ_2) < 1.
- P^{yes} in the the complement S/F.

- **Note**: for the calculation of pmin(s, φ_1 **U** φ_2), we can also just take S^{yes} = Sat(φ_2). The calculation would still works, though it would take more steps to get its final results.
-

Calculating pmax(s, φ_1 **U** φ_2)

pmax(s, φ) = the **maximum** probability for having executions starting from 2 satisfying φ, regardless the adversary.

To calculate pmax(s, φ_1 **U** φ_2) :

1. Calculate first the sets S^{yes} and S^{no}.
   - S^{yes} = { s | P(s, φ_1 U φ_2) ≥ 1, for **some** adversaries } → with algorithm prob1E.
   - S^{no} = { s | P(s, φ_1 U φ_2) ≤ 0, for **all** adversary } → with algorithm prob0A.
2. For any state s in S^{yes}, we then know that pmax(s, φ_1 **U** φ_2) ≥ 1.
3. For any state s in S^{no} we have pmax(s, φ_1 **U** φ_2) ≤ 0.
4. We then proceed with calculating the pmin for the remaining states (which are not in S^{yes} nor S^{no}).

Algorithm **Prob0A**

- The algorithm below first calculates the set R of all states s satisfying E[s, φ_1 U φ_2], for some adversary. So, for any state in R, there is an adversary such that Prob(s, φ_1 U φ_2) > 0.
- S^{no} is just complement S/R.
- Sat(φ_1) and Sat(φ_2) in the paremeters are the set of states on which φ_1 and φ_2 respectively hold.

PROB0A($Sat(\phi_1), Sat(\phi_2)$)
1. $R := Sat(\phi_2)$
2. $done :=$ **false**
3. **while** $(done =$ **false**$)$
4. $\quad R' := R \cup \{s \in Sat(\phi_1) \mid \exists \mu \in Steps(s) . \exists s' \in R . \mu(s') > 0\}$
5. $\quad$ **if** $(R' = R)$ **then** $done :=$ **true**
6. $\quad R := R'$
7. **endwhile**
8. **return** $S \backslash R$

Prob1E

- More complicated. See Dave's slides on MDP.

Prob1E