Probabilistic Model Checking

Stefan Pranger

03. 06. 2025

Outline

- Recap & Homework

Outline

- Recap & Homework
- **Part I:** PCTL and Markov Chains

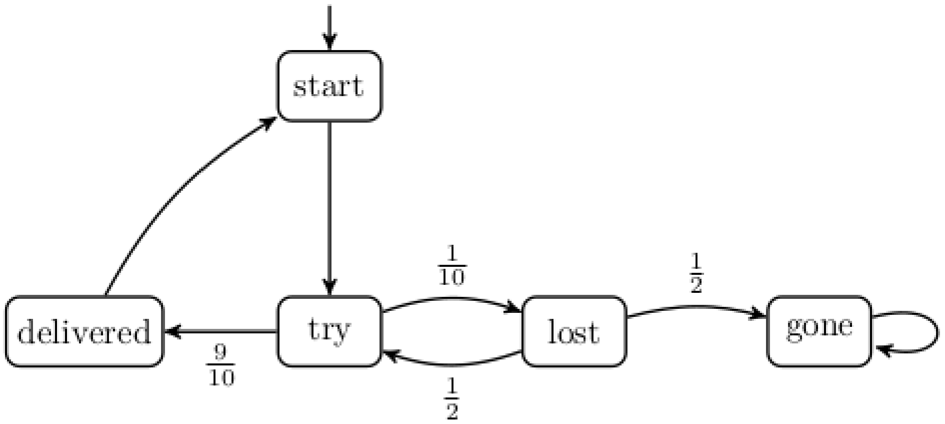
Outline

- Recap & Homework
- **Part I:** PCTL and Markov Chains
- **Part II:** Markov Decision Processes and Reachability Probabilities

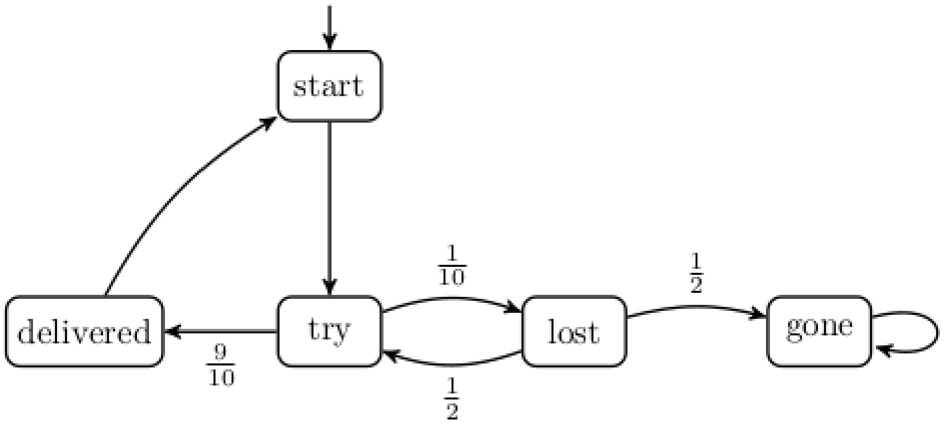
Outline

- Recap & Homework
- **Part I:** PCTL and Markov Chains
- **Part II:** Markov Decision Processes and Reachability Probabilities
- **Part III:** Probabilistic Shielding

Communication Protocol with Faults



Communication Protocol with Faults



$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -\frac{1}{10} \\ 0 & -\frac{1}{2} & 1 \end{bmatrix} \cdot \mathbf{x} = \begin{pmatrix} 0 \\ \frac{9}{10} \\ 0 \end{pmatrix} \rightarrow \mathbf{x} = \begin{pmatrix} \frac{18}{19} \\ \frac{18}{19} \\ \frac{9}{19} \end{pmatrix}$$

2D RobotGrid Random Walk

Part I

PCTL and Markov Chains

Recap: Reachability Probabilities

- Computing $Pr(\mathcal{M}, s_0 \models \mathbf{F} B)$
- We have used a linear equation solver to compute the probability of satisfying the reachability problem.

Probabilistic Computation Tree Logic

Probabilistic Computation Tree Logic [PCTL] is the probabilistic extension of CTL.

- Boolean state representation.
- $\forall$ and $\exists$ are replaced by $\mathrm{Pr}_J(\varphi)$, where $J \subseteq [0,1]$
  - The interpretation for each state $s \in S$: $Pr(\mathcal{M}, s \models \varphi) \in J$

PCTL - Syntax

Subdivision into *state* (Φ)- and *path*-formulae (φ):

$$\begin{aligned}\Phi ::= &\ \mathit{true}\\ &\mid a\\ &\mid \Phi_1 \wedge \Phi_2\\ &\mid \neg\Phi\\ &\mid \Pr_J(\varphi)\end{aligned} \qquad\qquad \begin{aligned}\varphi ::= &\ \mathbf{X}\Phi\\ &\mid \Phi_1\ \mathbf{U}\ \Phi_2\\ &\mid \Phi_1\ \mathbf{U}^{\leq n}\ \Phi_2\end{aligned}$$

where $a \in AP$ and $J \subseteq [0,1]$.

PCTL - Satisfaction Relation

For $s \in S, \Phi, \Psi$ PCTL *state* formulae, and φ a PCTL *path* formula.

For *state* formulae, we have:

$$
\begin{aligned}
s &\models a & &\text{iff } a \in L(s), \\
s &\models \neg\Phi & &\text{iff } s \not\models \Phi, \\
s &\models \Phi \wedge \Psi & &\text{iff } s \models \Phi \text{ and } s \models \Psi, \\
s &\models \Pr_J(\varphi) & &\text{iff } Pr(s \models \varphi) \in J
\end{aligned}
$$

PCTL - Satisfaction Relation

For $s \in S, \Phi, \Psi$ PCTL *state* formulae, and φ a PCTL *path* formula.

For *state* formulae, we have:

$$
\begin{array}{ll}
s \models a & \text{iff } a \in L(s), \\
s \models \neg\Phi & \text{iff } s \not\models \Phi, \\
s \models \Phi \wedge \Psi & \text{iff } s \models \Phi \text{ and } s \models \Psi, \\
s \models \mathrm{Pr}_J(\varphi) & \text{iff } Pr(s \models \varphi) \in J
\end{array}
$$

For *paths* $\pi \in \mathcal{M}$, we have:

$$
\begin{array}{lll}
\pi \models \mathbf{X}\varphi & \text{iff} & \pi[1] \models \varphi \\
\pi \models \varphi\,\mathbf{U}\,\psi & \text{iff} & \exists j \geq 0.\ (\pi[j] \models \psi \wedge (\forall 0 \leq k < j.\ \pi[k] \models \varphi) \\
\pi \models \varphi\,\mathbf{U}^{\leq n}\psi & \text{iff} & \exists 0 \leq j \leq n.\ (\pi[j] \models \psi \wedge (\forall 0 \leq k < j.\ \pi[k] \models \varphi)
\end{array}
$$

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?

Constrained Reachability

We are interested in $Pr(\mathcal{M}, s_0 \models C \ \mathbf{U} \ B)$

Constrained Reachability

We are interested in $Pr(\mathcal{M}, s_0 \models C \; \mathbf{U} \; B)$

Modify Step 1) of our algorithm:

1) Identify three disjoint subsets of S:

- $S_{=1}$: The set of states with a probability of 1 to **satisfy** $C \; \mathbf{U} \; B$.
- $S_{=0}$: The set of states with a probability of 0 to **satisfy** $C \; \mathbf{U} \; B$.
- $S_{?}$: The set of states with a probability $\in (0, 1)$ to **satisfy** $C \; \mathbf{U} \; B$.

Constrained Reachability

- $S_{=0}$: The set of states with a probability of 0 to **satisfy** C $\mathbf{U}$ B.

Constrained Reachability

- $S_{=0}$: The set of states with a probability of 0 to **satisfy** $C \ \mathbf{U} \ B$.
  - $Pr(\mathcal{M}, s_0 \models C \ \mathbf{U} \ B) = 0$ **iff** $G_{\mathcal{M}}, s_0 \not\models \exists(C \ \mathbf{U} \ B)$

Constrained Reachability

- $S_{=0}$: The set of states with a probability of 0 to **satisfy** $C \ \mathbf{U} \ B$.
  - $Pr(\mathcal{M}, s_0 \models C \ \mathbf{U} \ B) = 0$ **iff** $G_{\mathcal{M}}, s_0 \not\models \exists(C \ \mathbf{U} \ B)$
- $S_{=1}$: The set of states with a probability of 1 to **satisfy** $C \ \mathbf{U} \ B$.

Constrained Reachability

- $S_{=0}$: The set of states with a probability of 0 to **satisfy** $C\ \mathbf{U}\ B$.
  - $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B) = 0$ **iff** $G_{\mathcal{M}}, s_0 \not\models \exists(C\ \mathbf{U}\ B)$
- $S_{=1}$: The set of states with a probability of 1 to **satisfy** $C\ \mathbf{U}\ B$.
  - Modify $\mathcal{M}$ to $\mathcal{M}'$ by making states in $B \cup S \setminus (C \cup B)$ absorbing, i.e.

$$\mathbb{P}'(s,t) = \begin{cases} 1 & s = t \text{ and } s \in B \cup S \setminus (C \cup B) \\ 0 & s \neq t \text{ and } s \in B \cup S \setminus (C \cup B) \\ \mathbb{P}(s,t) & otherwise. \end{cases}$$

Constrained Reachability

- $S_{=0}$: The set of states with a probability of 0 to **satisfy** $C\ \mathbf{U}\ B$.
  - $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B) = 0$ **iff** $G_{\mathcal{M}}, s_0 \not\models \exists(C\ \mathbf{U}\ B)$
- $S_{=1}$: The set of states with a probability of 1 to **satisfy** $C\ \mathbf{U}\ B$.
  - Modify $\mathcal{M}$ to $\mathcal{M}'$ by making states in $B \cup S \setminus (C \cup B)$ absorbing, i.e.

$$\mathbb{P}'(s,t) = \begin{cases} 1 & s = t \text{ and } s \in B \cup S \setminus (C \cup B) \\ 0 & s \neq t \text{ and } s \in B \cup S \setminus (C \cup B) \\ \mathbb{P}(s,t) & otherwise. \end{cases}$$

  - Compute $S_{=1} = S \setminus Pre^*(S \setminus Pre^*(B))$
  - $Pre*(A)$ ... set of states that can reach A through a finite path fragment

Constrained Reachability

- $S_{=0}$: The set of states with a probability of 0 to **satisfy** C $\mathbf{U}$ B.
  - $Pr(\mathcal{M}, s_0 \models C \mathbf{U} B) = 0$ **iff** $G_{\mathcal{M}}, s_0 \not\models \exists(C \mathbf{U} B)$
- $S_{=1}$: The set of states with a probability of 1 to **satisfy** C $\mathbf{U}$ B.
  - Modify $\mathcal{M}$ to $\mathcal{M}'$ by making states in $B \cup S \setminus (C \cup B)$ absorbing, i.e.

$$\mathbb{P}'(s,t) = \begin{cases} 1 & s = t \text{ and } s \in B \cup S \setminus (C \cup B) \\ 0 & s \neq t \text{ and } s \in B \cup S \setminus (C \cup B) \\ \mathbb{P}(s,t) & otherwise. \end{cases}$$

  - Compute $S_{=1} = S \setminus Pre^*(S \setminus Pre^*(B))$
  - $Pre*(A)$ ... set of states that can reach A through a finite path fragment
- $S_?$: The set of states with a probability $\in (0,1)$ to **satisfy** C $\mathbf{U}$ B.

Constrained Reachability

- $S_{=0}$: The set of states with a probability of 0 to **satisfy** $C\ \mathbf{U}\ B$.
  - $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B) = 0$ **iff** $G_{\mathcal{M}}, s_0 \not\models \exists(C\ \mathbf{U}\ B)$
- $S_{=1}$: The set of states with a probability of 1 to **satisfy** $C\ \mathbf{U}\ B$.
  - Modify $\mathcal{M}$ to $\mathcal{M}'$ by making states in $B \cup S \setminus (C \cup B)$ absorbing, i.e.

$$\mathbb{P}'(s,t) = \begin{cases} 1 & s = t \text{ and } s \in B \cup S \setminus (C \cup B) \\ 0 & s \neq t \text{ and } s \in B \cup S \setminus (C \cup B) \\ \mathbb{P}(s,t) & otherwise. \end{cases}$$

  - Compute $S_{=1} = S \setminus Pre^*(S \setminus Pre^*(B))$
  - $Pre * (A)$ ... set of states that can reach A through a finite path fragment
- $S_?$: The set of states with a probability $\in (0,1)$ to **satisfy** $C\ \mathbf{U}\ B$.
  - No changes $\rightarrow$ solve the linear equation system ✓

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X} B)$?

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}B)$?
  - Also easy: Simple Matrix-Vector-Multiplication! ✓

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}B)$?
  - Also easy: Simple Matrix-Vector-Multiplication! ✓
- How can we compute bounded reachability: $Pr(\mathcal{M}, s_0 \models \mathbf{F}^{<=k}B)$?

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}B)$?
  - Also easy: Simple Matrix-Vector-Multiplication! ✓
- How can we compute bounded reachability: $Pr(\mathcal{M}, s_0 \models \mathbf{F}^{<=k}B)$?
  - Compute $\mathcal{M}' = \mathcal{M}_B$, with states in B *absorbing*

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}B)$?
  - Also easy: Simple Matrix-Vector-Multiplication! ✓
- How can we compute bounded reachability: $Pr(\mathcal{M}, s_0 \models \mathbf{F}^{<=k}B)$?
  - Compute $\mathcal{M}' = \mathcal{M}_B$, with states in B *absorbing*
  - Again: Simple Matrix-Vector-Multiplication(s)! ✓

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}B)$?
  - Also easy: Simple Matrix-Vector-Multiplication! ✓
- How can we compute bounded reachability: $Pr(\mathcal{M}, s_0 \models \mathbf{F}^{<=k}B)$?
  - Compute $\mathcal{M}' = \mathcal{M}_B$, with states in B *absorbing*
  - Again: Simple Matrix-Vector-Multiplication(s)! ✓
- How can we compute bounded constrained reachability: $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}^{<=k}B)$?

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}B)$?
  - Also easy: Simple Matrix-Vector-Multiplication! ✓
- How can we compute bounded reachability: $Pr(\mathcal{M}, s_0 \models \mathbf{F}^{<=k}B)$?
  - Compute $\mathcal{M}' = \mathcal{M}_B$, with states in B *absorbing*
  - Again: Simple Matrix-Vector-Multiplication(s)! ✓
- How can we compute bounded constrained reachability: $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}^{<=k}B)$?
  - Compute $\mathcal{M}' = \mathcal{M}_{B\cup(S\setminus(C\cup B))}$, with states in $B \cup S \setminus (C \cup B)$ *absorbing*

Model Checking a PCTL Formula

- Checking the propositional part of PCTL is easy ✓
- How to compute $Pr(\mathcal{M}, s_0 \models C \mathbf{U} B)$?
  - We solve a linear equation system. ✓
- How to compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}B)$?
  - Also easy: Simple Matrix-Vector-Multiplication! ✓
- How can we compute bounded reachability: $Pr(\mathcal{M}, s_0 \models \mathbf{F}^{<=k}B)$?
  - Compute $\mathcal{M}' = \mathcal{M}_B$, with states in B *absorbing*
  - Again: Simple Matrix-Vector-Multiplication(s)! ✓
- How can we compute bounded constrained reachability: $Pr(\mathcal{M}, s_0 \models C \mathbf{U}^{<=k}B)$?
  - Compute $\mathcal{M}' = \mathcal{M}_{B\cup(S\setminus(C\cup B))}$, with states in $B \cup S \setminus (C \cup B)$ *absorbing*
  - Compute bounded reachability in $\mathcal{M}'$: $Pr(\mathcal{M}', s_0 \models \mathbf{F}^{<=k}B)$ ✓

Model Checking a PCTL Formula

- We know how to
  - check the propositional part ✓
  - compute $Pr(\mathcal{M}, s_0 \models C \; \mathbf{U} \; B)$ ✓
  - compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}a)$ ✓
  - compute bounded constrained reachability: $Pr(\mathcal{M}, s_0 \models C \; \mathbf{U} \,^{<=k} B)$ ✓

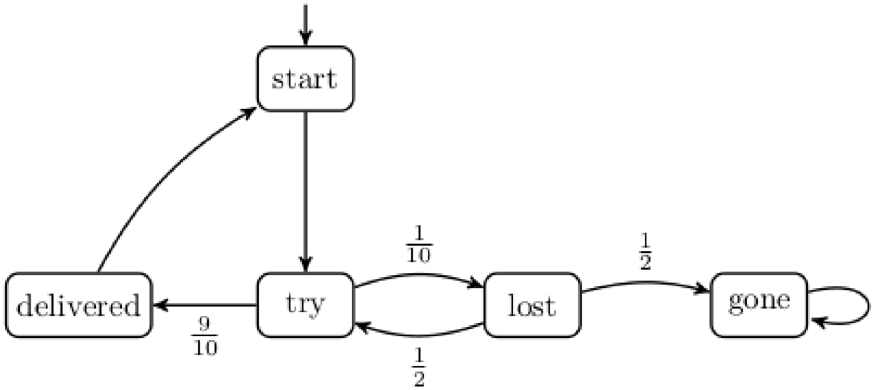
Model Checking a PCTL Formula

- We know how to
  - check the propositional part ✓
  - compute $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}\ B)$ ✓
  - compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}a)$ ✓
  - compute bounded constrained reachability: $Pr(\mathcal{M}, s_0 \models C\ \mathbf{U}^{<=k} B)$ ✓
- With that, we can answer $Pr(s \models \varphi) \in J$:

Model Checking a PCTL Formula

- We know how to
  - check the propositional part ✓
  - compute $Pr(\mathcal{M}, s_0 \models C \mathbf{U} B)$ ✓
  - compute $Pr(\mathcal{M}, s_0 \models \mathbf{X}a)$ ✓
  - compute bounded constrained reachability: $Pr(\mathcal{M}, s_0 \models C \mathbf{U}^{<=k} B)$ ✓
- With that, we can answer $Pr(s \models \varphi) \in J$:
- To check a PCTL formula Φ we traverse the parse tree

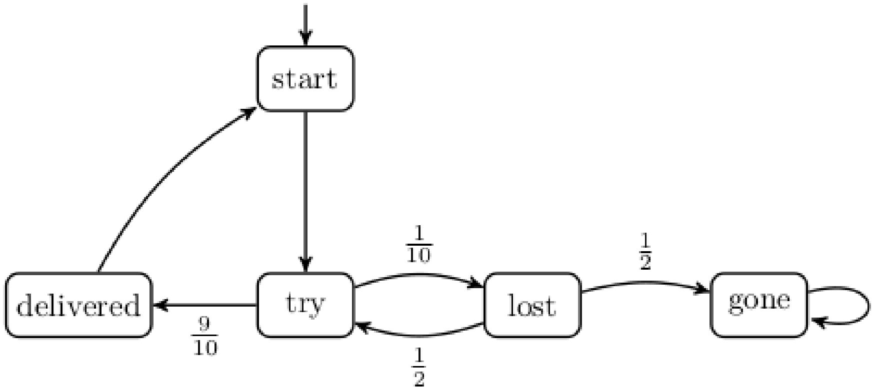
Communication Protocol



- *States where the message has not yet been lost can ensure 0.99 probability of sending the message within 6 time steps and after the message has been lost, there is a probability of 0.45 of successfully sending the message within the next try*:

```
(\"lost\" | \"gone\") | P>=0.99 [F<=6 \"delivered\"] & (!\"lost\") | P>=0.45 [ F<=2 \"delivered\"]
```

Communication Protocol



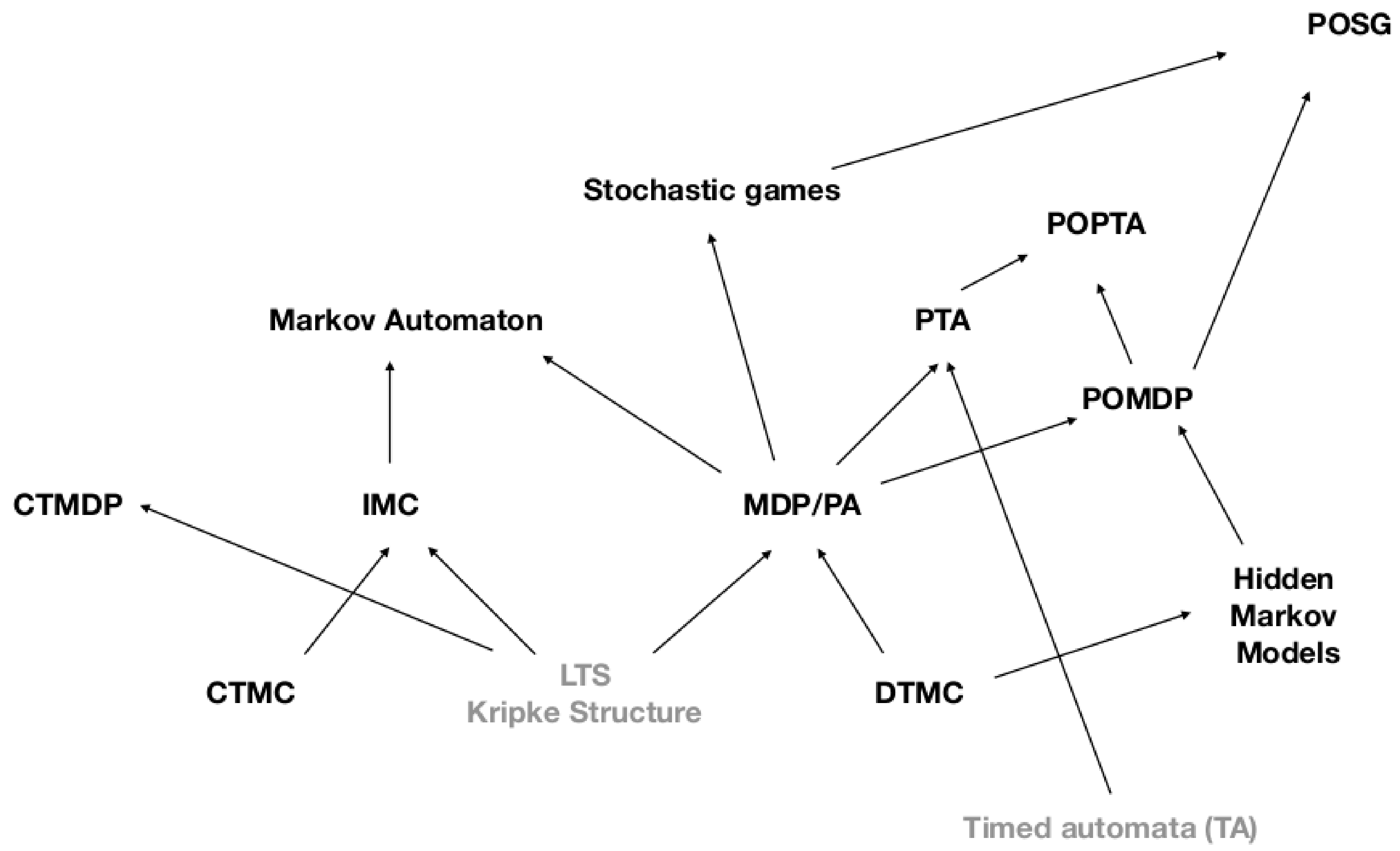
- *States where the message has not yet been lost can ensure 0.99 probability of sending the message within 6 time steps and after the message has been lost, there is a probability of 0.45 of successfully sending the message within the next try*:

```
(\"lost\" | \"gone\") | P>=0.99 [F<=6 \"delivered\"] & (!\"lost\") | P>=0.45 [ F<=2 \"delivered\"]
```

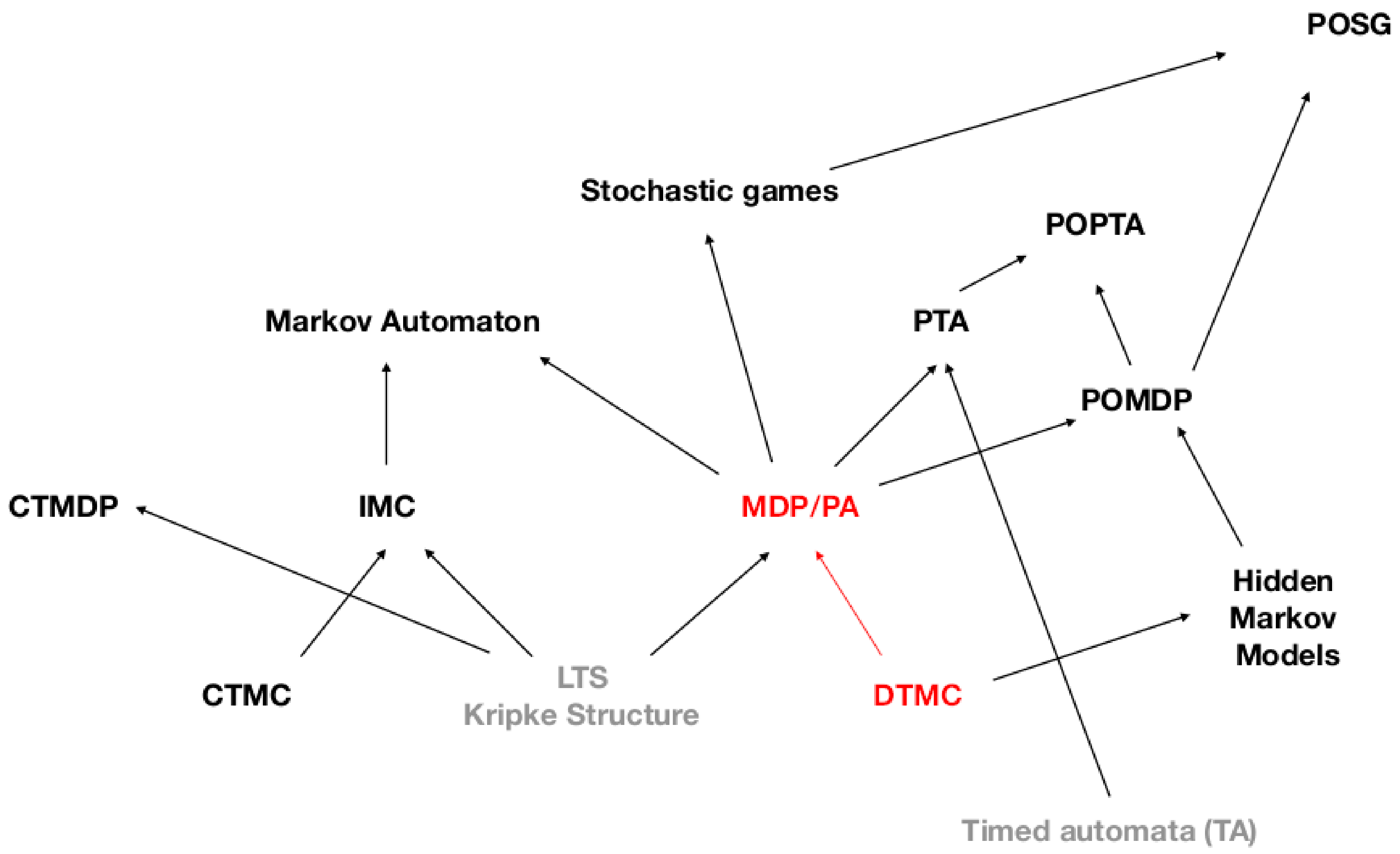
- *If the probability of completely losing the message within 2 time steps is greater or equal to 0.1 then the probability of eventually delivering the message is smaller than 0.9*

```
!(P>=0.1 [F<=2 \"gone\"] ) | P<0.9 [F \"delivered\"]
```

Model Zoo


POSG
Stochastic games
POPTA
Markov Automaton
PTA
POMDP
CTMDP
IMC
MDP/PA
Hidden
Markov
Models
CTMC
LTS
Kripke Structure
DTMC
Timed automata (TA)

Model Zoo


POSG
Stochastic games
POPTA
Markov Automaton
PTA
POMDP
CTMDP
IMC
MDP/PA
Hidden
Markov
Models
CTMC
LTS
Kripke Structure
DTMC
Timed automata (TA)

Part II

Markov Decision Processes and Reachability Probabilities

Markov Decision Processes

$\mathit{Markov\ Decision\ Process}\ \mathcal{M} = (S, \mathit{Act}, \mathbb{P}, s_0, AP, L)$

- S a set of states and initial state s_0,
- Act a set of actions,
- $\mathbb{P} : S \times \mathit{Act} \times S \to [0,1]$, s.t.

$$\sum_{s' \in S} \mathbb{P}(s, a, s') = 1 \ \forall (s, a) \in S \times \mathit{Act}$$

- AP set of atomic states and $L : S \to 2^{AP}$ a labelling function.

Markov Decision Processes

Markov Decision Process $\mathcal{M} = (S, Act, \mathbb{P}, s_0, AP, L)$

- S a set of states and initial state s_0,
- Act a set of actions,
- $\mathbb{P} : S \times Act \times S \to [0, 1]$, s.t.

$$\sum_{s' \in S} \mathbb{P}(s, a, s') = 1 \; \forall (s, a) \in S \times Act$$

- AP set of atomic states and $L : S \to 2^{AP}$ a labelling function.

The decision a defines the distribution over the next state.

Markov Decision Processes in Code and Memory

- Commands:

```
[moveNorth] x<HEIGHT -> 0.9: (x'=x+1) + 0.1: true;
[moveEast]  y<WIDTH  -> 0.9: (y'=y-1) + 0.1: true;
```

Markov Decision Processes in Code and Memory

- Commands:

```
[moveNorth] x<HEIGHT -> 0.9: (x'=x+1) + 0.1: true;
[moveEast]  y<WIDTH  -> 0.9: (y'=y-1) + 0.1: true;
```

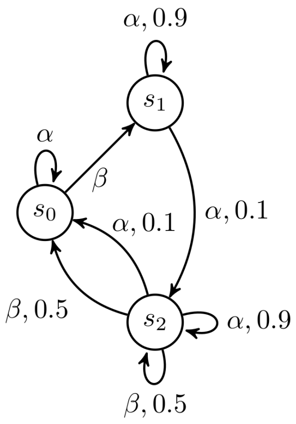
  - Guards do not need to be mutually exclusive anymore!

Markov Decision Processes in Code and Memory

- Commands:

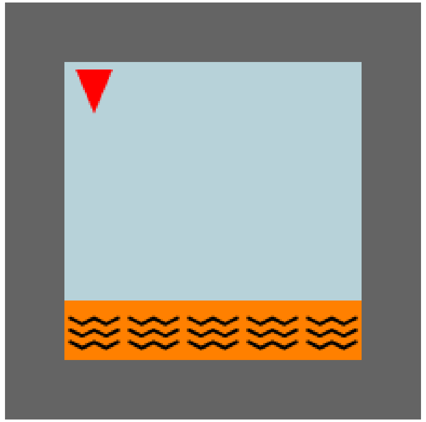
```
[moveNorth] x<HEIGHT -> 0.9: (x'=x+1) + 0.1: true;
[moveEast]  y<WIDTH  -> 0.9: (y'=y-1) + 0.1: true;
```

  - Guards do not need to be mutually exclusive anymore!



$$\left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hline 0 & \frac{9}{10} & \frac{1}{10} \\ \hline \frac{1}{10} & 0 & \frac{9}{10} \\ \frac{5}{10} & 0 & \frac{5}{10} \end{array}\right]$$

Coding Example



```
mdp

...

module controllable_robot









endmodule
```

Paths in an MDP

- We extend our definition of a path for an MDP $\mathcal{M}$ as such:
- $\pi = s_0 a_0 s_1 a_1 s_2 a_2 \ldots \in (S \times Act)^\omega$, s.t. $\mathbb{P}(s_i, a_i, s_{i+1}) > 0, \forall i \geq 0$

Paths in an MDP

- We extend our definition of a path for an MDP $\mathcal{M}$ as such:
- $\pi = s_0 a_0 s_1 a_1 s_2 a_2 \ldots \in (S \times Act)^\omega$, s.t. $\mathbb{P}(s_i, a_i, s_{i+1}) > 0, \forall i \geq 0$
- Reasoning about events in an MDP resorts to the resolution of any non-determinism
  - This is done by the use of *schedulers* (or: strategies/policies/adversaries).

Schedulers

- A scheduler is a function that given the history of the current path returns a distribution over actions to be taken:

$$\sigma : S^* \times S \to Distr(Act)$$

Schedulers

- A scheduler is a function that given the history of the current path returns a distribution over actions to be taken:

$$\sigma : S^* \times S \to \mathit{Distr}(\mathit{Act})$$

- For simple properties such as unbounded reachability so-called *memoryless deterministic* scheduler suffice:

$$\sigma : S \to \mathit{Act}$$

Schedulers

- A scheduler is a function that given the history of the current path returns a distribution over actions to be taken:

$$\sigma : S^* \times S \rightarrow Distr(Act)$$

- For simple properties such as unbounded reachability so-called *memoryless deterministic* scheduler suffice:

$$\sigma : S \rightarrow Act$$

  - This means that the scheduler σ fixes an actions for each state.

Schedulers

- A scheduler is a function that given the history of the current path returns a distribution over actions to be taken:

$$\sigma : S^* \times S \rightarrow Distr(Act)$$

- For simple properties such as unbounded reachability so-called *memoryless deterministic* scheduler suffice:

$$\sigma : S \rightarrow Act$$

  - This means that the scheduler σ fixes an actions for each state.
  - We can then define the probability of eventually reaching under sched

$$Pr^\sigma(\mathcal{M}, s \models \mathbf{F} B)$$

Induced Markov Chain

Consider an MDP $\mathcal{M}$ and a memoryless deterministic scheduler:

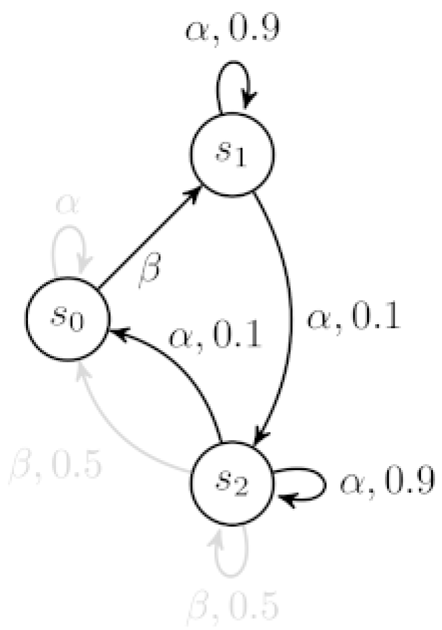
$$\sigma : S \to Act$$

Induced Markov Chain

Consider an MDP $\mathcal{M}$ and a memoryless deterministic scheduler:

$$\sigma : S \to Act$$

$$\begin{aligned} s_0 &\mapsto \beta \\ s_1 &\mapsto \alpha \\ s_1 &\mapsto \alpha \end{aligned}$$



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \hline 0 & \frac{9}{10} & \frac{1}{10} \\ \hline \frac{1}{10} & 0 & \frac{9}{10} \\ \frac{5}{10} & 0 & \frac{5}{10} \end{bmatrix}$$

Reachability in MDPs

- We have (re-)introduced nondeterminism into probabilistic models

Reachability in MDPs

- We have (re-)introduced nondeterminism into probabilistic models
- Satisfaction of $Pr^{\sigma}(\mathcal{M}, s \models \mathbf{F} B)$ depends on σ, which can either

Reachability in MDPs

- We have (re-)introduced nondeterminism into probabilistic models
- Satisfaction of $Pr^{\sigma}(\mathcal{M}, s \models \mathbf{F} B)$ depends on σ, which can either
  - *maximize* or
  - *minimize* the probability to satisfy $\mathbf{F} B$

Reachability in MDPs

- We have (re-)introduced nondeterminism into probabilistic models
- Satisfaction of $Pr^{\sigma}(\mathcal{M}, s \models \mathbf{F}B)$ depends on σ, which can either
  - *maximize* or
  - *minimize* the probability to satisfy $\mathbf{F}B$
- We are therefore interested in a *worst-case analysis* ranging over all schedulers σ

Reachability in MDPs

- We have (re-)introduced nondeterminism into probabilistic models
- Satisfaction of $Pr^{\sigma}(\mathcal{M}, s \models \mathbf{F} B)$ depends on σ, which can either
  - *maximize* or
  - *minimize* the probability to satisfy $\mathbf{F} B$
- We are therefore interested in a *worst-case analysis* ranging over all schedulers σ
- Formally, we have:
  - $Pr^{max}(\mathcal{M}, s \models \mathbf{F} B) = sup_{\sigma} Pr^{\sigma}(\mathcal{M}, s \models \mathbf{F} B)$ and
  - $Pr^{min}(\mathcal{M}, s \models \mathbf{F} B) = inf_{\sigma} Pr^{\sigma}(\mathcal{M}, s \models \mathbf{F} B)$

Example: Maximal Probability of Reaching B

Assume we are interested in the probability of staying safe:

$$Pr^{\sigma}(\mathcal{M}, s \models \mathbf{G}\neg B)$$

Example: Maximal Probability of Reaching B

Assume we are interested in the probability of staying safe:

$$Pr^{\sigma}(\mathcal{M}, s \models \mathbf{G}\neg B)$$

- Worst-case analysis: What is the *maximum* probability of ever reaching B?

Example: Maximal Probability of Reaching B

Assume we are interested in the probability of staying safe:

$$Pr^{\sigma}(\mathcal{M}, s \models \mathbf{G}\neg B)$$

- Worst-case analysis: What is the *maximum* probability of ever reaching B?
- Compute $Pr^{max}(\mathcal{M}, s \models \mathbf{F}B)$

Example: Maximal Probability of Reaching B

Assume we are interested in the probability of staying safe:

$$Pr^{\sigma}(\mathcal{M}, s \models \mathbf{G}\neg B)$$

- Worst-case analysis: What is the *maximum* probability of ever reaching B?
- Compute $Pr^{max}(\mathcal{M}, s \models \mathbf{F}B)$
- If $Pr^{max}(\mathcal{M}, s \models \mathbf{F}B) \leq \varepsilon$ then $Pr^{\sigma}(\mathcal{M}, s \models \mathbf{G}\neg B) \geq 1 - \varepsilon$
  - *Regardless of the resolution of nondeterminism, the probability of staying safe is* $\geq 1 - \varepsilon$

Computing Maximum Reachability Probabilities in MDPs

Computing Maximum Reachability Probabilities in MDPs

We want to compute $(x_s) = Pr^{max}(\mathcal{M}, s \models \mathbf{F} B)$ using the following equation system:

Computing Maximum Reachability Probabilities in MDPs

We want to compute $(x_s) = Pr^{max}(\mathcal{M}, s \models \mathbf{F} B)$ using the following equation system:

- If $s \in B$: $x_s = 1$

Computing Maximum Reachability Probabilities in MDPs

We want to compute $(x_s) = Pr^{max}(\mathcal{M}, s \models \mathbf{F} B)$ using the following equation system:

- If $s \in B$: $x_s = 1$
- If $s \not\models \exists \mathbf{F} B$: $x_s = 0$

Computing Maximum Reachability Probabilities in MDPs

We want to compute $(x_s) = Pr^{max}(\mathcal{M}, s \models \mathbf{F}B)$ using the following equation system:

- If $s \in B$: $x_s = 1$
- If $s \not\models \exists\mathbf{F}B$: $x_s = 0$
- If $s \notin B$ and $s \models \exists\mathbf{F}B$
  - $x_s = \max\{\sum_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'} | a \in Act(s)\}$

Computing Maximum Reachability Probabilities in MDPs

We want to compute $(x_s) = Pr^{max}(\mathcal{M}, s \models \mathbf{F}B)$ using the following equation system:

- If $s \in B$: $x_s = 1$
- If $s \not\models \exists\mathbf{F}B$: $x_s = 0$
- If $s \notin B$ and $s \models \exists\mathbf{F}B$
  - $x_s = \max\{\sum_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'} | a \in Act(s)\}$
- Such that $\sum_{x \in S} x_s$ is minimal.

Linear Program - Method I

This can be expressed as a *linear program*:

- Minimize $\sum_{x \in S} x_s$, such that:
  - $0 \leq x_s \leq 1$,
  - $x_s = 1$, if $s \in B$,
  - $x_s = 0$, if $s \not\models \exists \mathbf{F} B$,
  - $x_s \geq \sum_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'}$, for all actions $a \in Act(s)$, if $s \notin B$ and $s \models \exists \mathbf{F} B$

Value Iteration - Method II

- Approximative method:
  - Compute the probability to reach B after n steps
  - Start with $n = 0$ and stop after some termination criterion is met

Value Iteration - Method II

- Approximative method:
  - Compute the probability to reach B after n steps
  - Start with $n = 0$ and stop after some termination criterion is met

More specifically:

$$
\begin{aligned}
x_s^{(0)} &= 1, \forall s \in B \\
x_s^{(n)} &= 0, \forall s \in S_{=0} \\
x_s^{(0)} &= 0, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \forall s \in S \setminus S_{=0} \\
x_s^{(n+1)} &= \max\{\textstyle\sum_{s' \in S} \mathbb{P}(s,a,s') \cdot x_{s'}^{(n)} | a \in Act(s)\}, \forall s \in S \setminus S_{=0}
\end{aligned}
$$

Value Iteration - Method II

- Approximative method:
  - Compute the probability to reach B after n steps
  - Start with $n = 0$ and stop after some termination criterion is met

More specifically:

$$\begin{aligned} x_s^{(0)} &= 1, \forall s \in B \\ x_s^{(n)} &= 0, \forall s \in S_{=0} \\ x_s^{(0)} &= 0, & \forall s \in S \setminus S_{=0} \\ x_s^{(n+1)} &= \max\{\sum_{s' \in S} \mathbb{P}(s,a,s') \cdot x_{s'}^{(n)} | a \in Act(s)\}, & \forall s \in S \setminus S_{=0} \end{aligned}$$

- Terminate as soon as $\max_{x_s \in S} |x_s^{(n+1)} - x_s^{(n)}| < \varepsilon$

Value Iteration - Method II

- Approximative method:
  - Compute the probability to reach B after n steps
  - Start with $n = 0$ and stop after some termination criterion is met

More specifically:

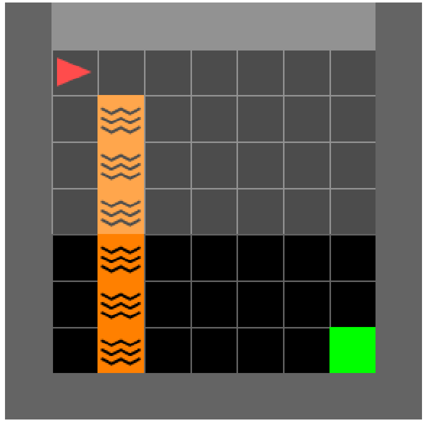
$$\begin{aligned} x_s^{(0)} &= 1, \forall s \in B \\ x_s^{(n)} &= 0, \forall s \in S_{=0} \\ x_s^{(0)} &= 0, \qquad \forall s \in S \setminus S_{=0} \\ x_s^{(n+1)} &= \max\{\sum\nolimits_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'}^{(n)} | a \in Act(s)\}, \forall s \in S \setminus S_{=0} \end{aligned}$$

- Terminate as soon as $\max_{x_s \in S} \left| x_s^{(n+1)} - x_s^{(n)} \right| < \varepsilon$
- More sophisticated methods use other means of checking for convergence

Part III

Probabilistic Shielding

Safety in Reinforcement Learning



Probabilistic Shielding

We are given an MDP $\mathcal{M}$ and a set of unsafe states B

Probabilistic Shielding

We are given an MDP $\mathcal{M}$ and a set of unsafe states B

Idea

- Limit the probability to reach B by *disallowing unsafe actions*

Probabilistic Shielding

We are given an MDP $\mathcal{M}$ and a set of unsafe states B

Idea

- Limit the probability to reach B by *disallowing unsafe actions*
- How to compute?

Value Iteration

$$
\begin{aligned}
x_s^{(0)} &= 1, \forall s \in B \\
x_s^{(n)} &= 0, \forall s \in S_{=0} \\
x_s^{(0)} &= 0, & \forall s \in S \setminus S_{=0} \\
x_s^{(n+1)} &= \max\{\textstyle\sum_{s' \in S} \mathbb{P}(s, a, s') \cdot x_{s'}^{(n)} \mid a \in Act(s)\}, & \forall s \in S \setminus S_{=0}
\end{aligned}
$$

Value Iteration

$$
\begin{aligned}
x_s^{(0)} &= 1, \forall s \in B \\
x_s^{(n)} &= 0, \forall s \in S_{=0} \\
x_s^{(0)} &= 0, \qquad \forall s \in S \setminus S_{=0} \\
x_s^{(n+1)} &= \max\{\sum\nolimits_{s' \in S} \mathbb{P}(s,a,s') \cdot x_{s'}^{(n)} | a \in Act(s)\}, \forall s \in S \setminus S_{=0}
\end{aligned}
$$

Value Iteration

$$
\begin{aligned}
x_s^{(0)} &= 1, \forall s \in B \\
x_s^{(n)} &= 0, \forall s \in S_{=0} \\
x_s^{(0)} &= 0, \qquad \forall s \in S \setminus S_{=0} \\
x_s^{(n+1)} &= \max\{\textstyle\sum_{s' \in S} \mathbb{P}(s,a,s') \cdot x_{s'}^{(n)} | a \in Act(s)\}, \forall s \in S \setminus S_{=0}
\end{aligned}
$$

- Value iterations computes the probability to satisfy $Pr^{max/min}(\mathcal{M}, s \models \mathbf{F} B)$ for every *state-action* pair

Value Iteration

$$
\begin{aligned}
x_s^{(0)} &= 1, \forall s \in B \\
x_s^{(n)} &= 0, \forall s \in S_{=0} \\
x_s^{(0)} &= 0, \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall s \in S \setminus S_{=0} \\
x_s^{(n+1)} &= \max\{\textstyle\sum_{s' \in S} \mathbb{P}(s,a,s') \cdot x_{s'}^{(n)} | a \in Act(s)\}, \forall s \in S \setminus S_{=0}
\end{aligned}
$$

- Value iterations computes the probability to satisfy $Pr^{max/min}(\mathcal{M}, s \models \mathbf{F} B)$ for every *state-action* pair
- Use the intermediate computation for
  - $Pr^{max}(s,a,n) = \Sigma_{s' \in S} \mathbb{P}(s,a,s') * x_s^{(n-1)}$

Probabilistic Shielding

- Limit the probability to reach B by *disallowing unsafe actions*

Probabilistic Shielding

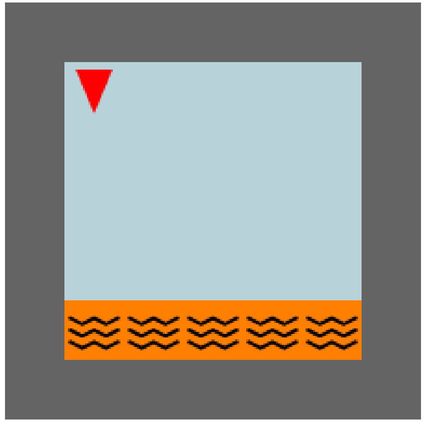
- Limit the probability to reach B by *disallowing unsafe actions*
- We now have:
  - $Pr^{max}(s, a, n) = \Sigma_{s' \in S} \mathbb{P}(s, a, s') * x_s^{(n-1)}$
  - For a specified *safety threshold* γ disallow actions for which: $Pr^{max}(s, a, n) < \gamma$

Probabilistic Shielding

- Limit the probability to reach B by *disallowing unsafe actions*
- We now have:
  - $Pr^{max}(s, a, n) = \Sigma_{s' \in S} \mathbb{P}(s, a, s') * x_s^{(n-1)}$
  - For a specified *safety threshold* γ disallow actions for which: $Pr^{max}(s, a, n) < \gamma$
  - Similary, we can disallow actions for which $Pr^{min}(s, a, n) > \gamma$

Probabilistic Shielding - Example

- Safety Property: *Do not step onto lava within 2 time steps*
- Disallow if $Pr^{min}(s, a, 2) > 0.05$ for (s, a)



```
Pre-Safety-Shield with absolute comparison (gamma = 0.050000):
      model state:    choice(s) [<value>: (<action {action label})>]:
    0: [x=1    & y=1]    0: (0 {})
    1: [x=2    & y=1]    0: (0 {})
    2: [x=3    & y=1]    0: (0 {})
    3: [x=4    & y=1]    0: (0 {})
    4: [x=5    & y=1]    0: (0 {})
    5: [x=2    & y=2]    0.0318: (2 {north})
    6: [x=3    & y=2]    0.0318: (2 {north})
    7: [x=4    & y=2]    0.0318: (2 {north})
    8: [x=5    & y=2]    0.0318: (1 {north})
    9: [x=2    & y=3]    0.0009: (0 {east});    0.0009: (1 {west});    0.0009: (2 {north});    0.0273: (3 {south})
   10: [x=3    & y=3]    0.0009: (0 {east});    0.0009: (1 {west});    0.0009: (2 {north});    0.0273: (3 {south})
   11: [x=4    & y=3]    0.0009: (0 {east});    0.0009: (1 {west});    0.0009: (2 {north});    0.0273: (3 {south})
   12: [x=5    & y=3]    0.0009: (0 {west});    0.0009: (1 {north});    0.0273: (2 {south})
   13: [x=2    & y=4]    0: (0 {east});    0: (1 {west});    0: (2 {north});    0: (3 {south})
   14: [x=3    & y=4]    0: (0 {east});    0: (1 {west});    0: (2 {north});    0: (3 {south})
   15: [x=4    & y=4]    0: (0 {east});    0: (1 {west});    0: (2 {north});    0: (3 {south})
   16: [x=5    & y=4]    0: (0 {west});    0: (1 {north});    0: (2 {south})
   17: [x=2    & y=5]    0: (0 {east});    0: (1 {west});    0: (2 {south})
   18: [x=3    & y=5]    0: (0 {east});    0: (1 {west});    0: (2 {south})
   19: [x=4    & y=5]    0: (0 {east});    0: (1 {west});    0: (2 {south})
   20: [x=5    & y=5]    0: (0 {west});    0: (1 {south})
   21: [x=1    & y=2]    0.0318: (1 {north})
   22: [x=1    & y=3]    0.0009: (0 {east});    0.0009: (1 {north});    0.0273: (2 {south})
   23: [x=1    & y=4]    0: (0 {east});    0: (1 {north});    0: (2 {south})
   24: [x=1    & y=5]    0: (0 {east});    0: (1 {south})
```