Tutorial AAMAS 2026: Formal Methods for Safe Reinforcement Learning

Alexander W. Goodall, Dr Edwin Hamel-De le Court, Dr Francesco Belardinelli

Imperial College London

May 26, 2026

Open and Important Problems in AI

AI systems are often deployed with no guarantees about their behaviour. We would like:

- **Generalisation** and **Robustness** w.r.t. out-od-distribution data.

[1] Y. Bengio et al.; *Superintelligent Agents Pose Catastrophic Risks: Can Scientist AI Offer a Safer Path?* 2025.

Open and Important Problems in AI

AI systems are often deployed with no guarantees about their behaviour. We would like:

- **Generalisation** and **Robustness** w.r.t. out-od-distribution data.
- **Safety:** rewards with formal semantics (avoids reward misspecification / reward hacking / goal misgeneralization), safe exploration, requirements & constraints, risk management.

[1] Y. Bengio et al.; *Superintelligent Agents Pose Catastrophic Risks: Can Scientist AI Offer a Safer Path?* 2025.

Open and Important Problems in AI

AI systems are often deployed with no guarantees about their behaviour. We would like:

- **Generalisation** and **Robustness** w.r.t. out-od-distribution data.
- **Safety:** rewards with formal semantics (avoids reward misspecification / reward hacking / goal misgeneralization), safe exploration, requirements & constraints, risk management.
- **Security:** avoids training data leakage, privacy, anonymity, robust to adversarial attacks.

[1] Y. Bengio et al.; *Superintelligent Agents Pose Catastrophic Risks: Can Scientist AI Offer a Safer Path?* 2025.

Open and Important Problems in AI

AI systems are often deployed with no guarantees about their behaviour. We would like:

- **Generalisation** and **Robustness** w.r.t. out-od-distribution data.
- **Safety:** rewards with formal semantics (avoids reward misspecification / reward hacking / goal misgeneralization), safe exploration, requirements & constraints, risk management.
- **Security:** avoids training data leakage, privacy, anonymity, robust to adversarial attacks.
- **AI Alignment:** AI that is aligned with human preferences and societal norms. [1]

[1] Y. Bengio et al.; *Superintelligent Agents Pose Catastrophic Risks: Can Scientist AI Offer a Safer Path?* 2025.

What we do: Safety through Formal Methods

- Who are we?

What we do: Safety through Formal Methods

- Who are we? the Lab on Formal Methods in AI (FMAI@ICL) `https://fmailab.doc.ic.ac.uk/`
- **Lab Mission:** to provide *formal* guarantees of the safety, trustworthiness and robustness of AI-based decision-making systems.

What we do: Safety through Formal Methods

- Who are we? the Lab on Formal Methods in AI (FMAI@ICL) `https://fmailab.doc.ic.ac.uk/`
- **Lab Mission:** to provide *formal* guarantees of the safety, trustworthiness and robustness of AI-based decision-making systems.
- Why formal methods? Provide strict guarantees through proofs.

What we do: Safety through Formal Methods

- Who are we? the Lab on Formal Methods in AI (FMAI@ICL) `https://fmailab.doc.ic.ac.uk/`
- **Lab Mission:** to provide *formal* guarantees of the safety, trustworthiness and robustness of AI-based decision-making systems.
- Why formal methods? Provide strict guarantees through proofs.
- Research Areas:
  - Model Checking
  - Runtime Verification
  - Safe RL, including Shielding
  - Multi-agent Systems

Open-source Library: MASA-Safe-RL

What is MASA-Safe-RL?

- **MASA-Safe-RL** is our open-source library for modular safe RL experiments.
- We will use it throughout the tutorial to connect the formalisms to executable examples, e.g., labelled MDPs, safety monitors etc.

Open-source Library: MASA-Safe-RL

What is MASA-Safe-RL?

- **MASA-Safe-RL** is our open-source library for modular safe RL experiments.
- We will use it throughout the tutorial to connect the formalisms to executable examples, e.g., labelled MDPs, safety monitors etc.

Links

- Library: `https://github.com/sacktock/MASA-Safe-RL`
- Tutorial notebooks: `https://github.com/sacktock/MASA-Safe-RL/tree/main/tutorial`
- Docs: `https://sacktock.github.io/MASA-Safe-RL/`

Open-source Library: MASA-Safe-RL

What is MASA-Safe-RL?

- **MASA-Safe-RL** is our open-source library for modular safe RL experiments.
- We will use it throughout the tutorial to connect the formalisms to executable examples, e.g., labelled MDPs, safety monitors etc.

Links

- Library: `https://github.com/sacktock/MASA-Safe-RL`
- Tutorial notebooks: `https://github.com/sacktock/MASA-Safe-RL/tree/main/tutorial`
- Docs: `https://sacktock.github.io/MASA-Safe-RL/`

Installation

See: tutorial/README.md

Plan

1. Why RL and Formal Methods?
   - Formal Methods Meets Learning
2. What is Reinforcement Learning?
   - Preliminaries
   - Markov Decision Processes
   - Reward Optimal Policies
   - Constrained Reinforcement Learning
3. A Quick Primer on Linear Temporal Logic
   - Reinforcement Learning with Linear Temporal Logic
4. Safety LTL and Product MDP
   - Product MDP Construction
   - Reachability and Safe Sets
5. Safety Compliance via Shielding
   - Shielding Methodologies: Absolute Safety
   - Shielding Methodologies: Probabilistic Safety
   - Shielding without Prior Knowledge
6. References

Why RL and Formal Methods?

Why RL and Formal Methods?
Formal Methods Meets Learning

Safe Learning Problem

How can we guarantee that learning agents will abide to safety requirements at deployment time (possibly at exploration time too!)

Safe Learning Problem

How can we guarantee that learning agents will abide to safety requirements at deployment time (possibly at exploration time too!)

Key question for:

- autonomous vehicles
- robotics (drones, swarms, smart warehouses)
- finance (automated trading)
- utility management (including power grid)
- ...

Safe Learning Problem

How can we guarantee that learning agents will abide to safety requirements at deployment time (possibly at exploration time too!)

Key question for:

- autonomous vehicles
- robotics (drones, swarms, smart warehouses)
- finance (automated trading)
- utility management (including power grid)
- ...

More formally: *find an optimal policy* π^* *such that* $\pi^* \models \phi$

where ϕ is our (safety) spec and $\models$ is a formal satisfaction notion.

Related Research Questions

- How do we formally define *safety*?

Related Research Questions

- How do we formally define *safety*?
- Are they other properties that we want to enforce, e.g., *liveness* / *fairness*?

Related Research Questions

- How do we formally define *safety*?
- Are they other properties that we want to enforce, e.g., *liveness* / *fairness*?
- How can we *enforce* satisfaction or compliance in dynamics environments with agents adapting at runtime?

Related Research Questions

- How do we formally define *safety*?
- Are they other properties that we want to enforce, e.g., *liveness* / *fairness*?
- How can we *enforce* satisfaction or compliance in dynamics environments with agents adapting at runtime?
- How can we *prove* satisfaction and compliance even in the presence of neural network controllers and complex dynamics?

Related Research Questions

- How do we formally define *safety*?
- Are they other properties that we want to enforce, e.g., *liveness* / *fairness*?
- How can we *enforce* satisfaction or compliance in dynamics environments with agents adapting at runtime?
- How can we *prove* satisfaction and compliance even in the presence of neural network controllers and complex dynamics?
- What is the best we can do in truly unknown environment dynamics? Can we obtain any guarantees in this setting?

Formal Methods to the Rescue

Answers through the application of formal methods:

> *"formal methods can be considered as the applied mathematics for modeling and analyzing ICT systems. Their aim is to establish system correctness with mathematical rigor." [Baier&Katoen, 2008]*

Formal Methods to the Rescue

Answers through the application of formal methods:

> *"formal methods can be considered as the applied mathematics for modeling and analyzing ICT systems. Their aim is to establish system correctness with mathematical rigor." [Baier&Katoen, 2008]*

- Knowledge Representation/Formal Languages
- Automata Theory
- Model Checking
- Theorem Proving
- Runtime Verification
- SAT/SMT Solving
- Controller Synthesis
- ...

Tutorial Promise

By the end, you should understand how a safety requirement can be used as a monitor, combine it with an MDP, compute safe/risk-bounded choices, and train an RL policy behind a shield.

Tutorial Promise

By the end, you should understand how a safety requirement can be used as a monitor, combine it with an MDP, compute safe/risk-bounded choices, and train an RL policy behind a shield.

You should also have a nice introduction into how to implement this things in practice with standard RL interfaces such as Gymnasiusm via the MASA-Safe-RL library.

Running Example

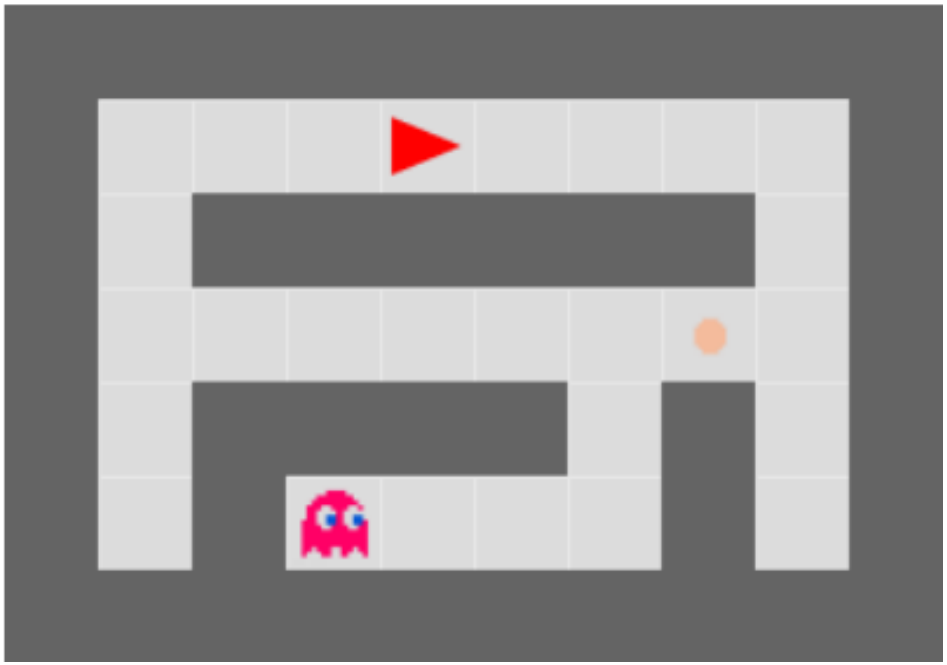


Figure: `MiniPacman` game.

Running Example

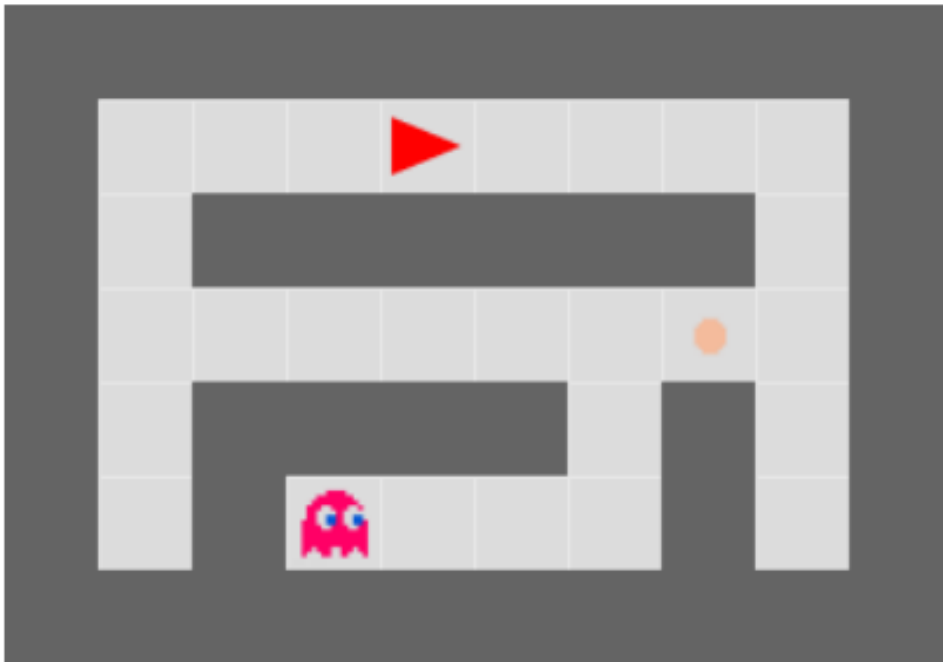


Figure: `MiniPacman` game.

Informally,

Running Example

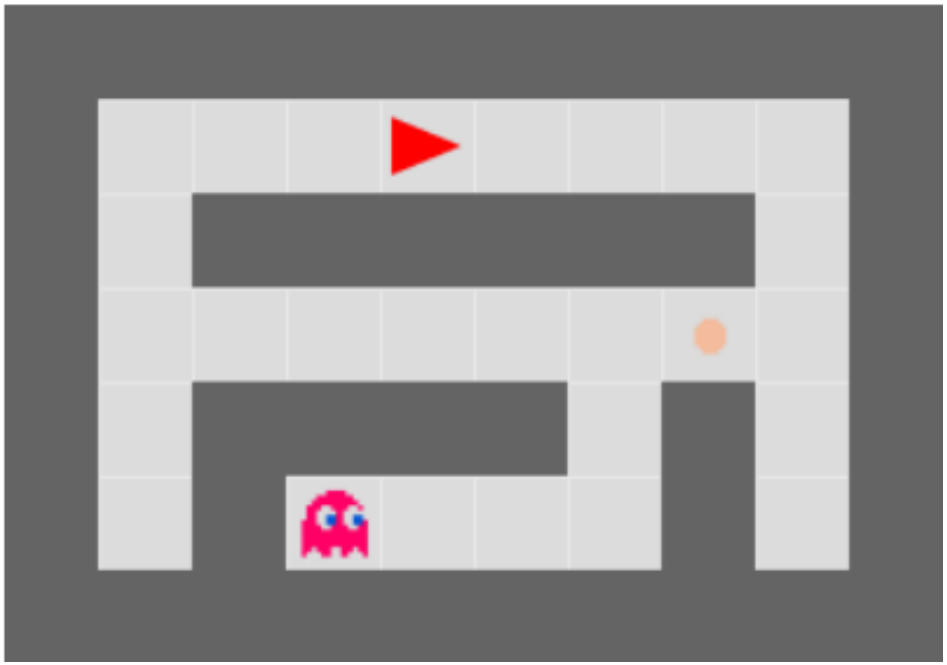


Figure: `MiniPacman` game.

Informally,

- Goal: "collect the food".

Running Example

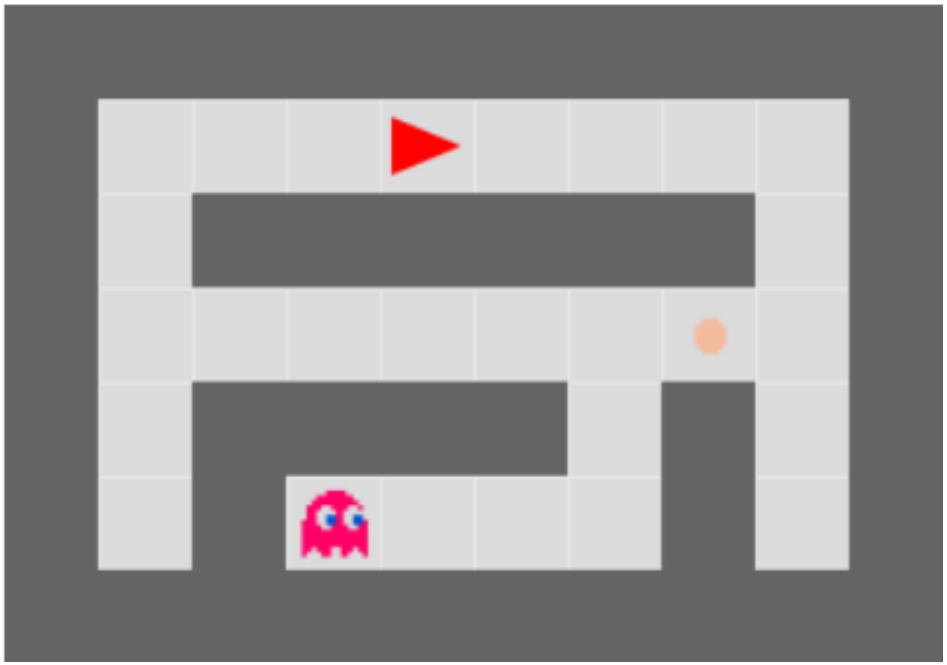


Figure: `MiniPacman` game.

Informally,

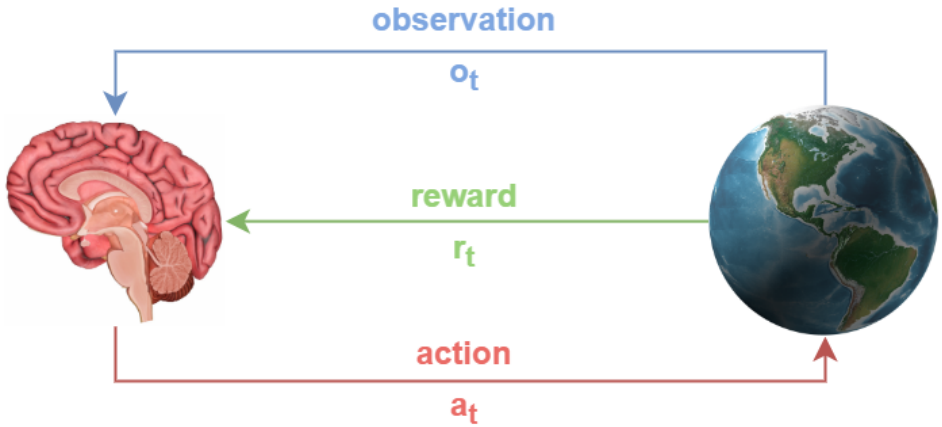
- Goal: “collect the food”.
- Safety property: “avoid the ghost”.

What is Reinforcement Learning?

What is Reinforcement Learning?
Preliminaries

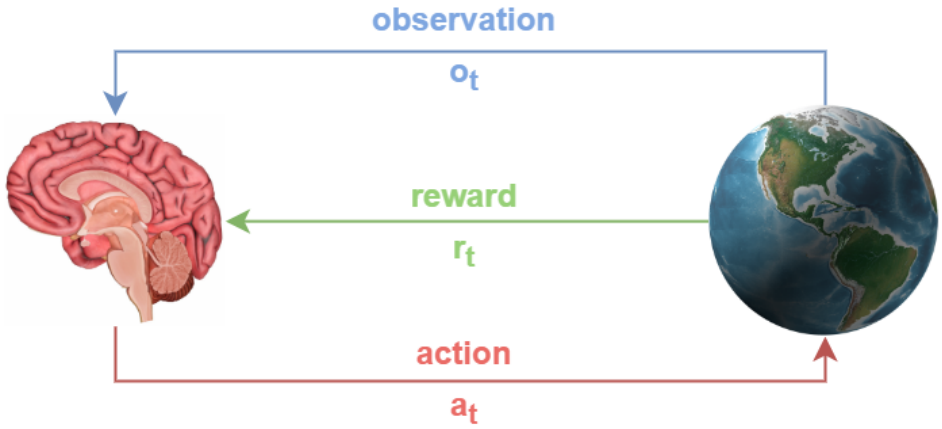
Agent-environment Interaction

- In reinforcement learning (RL) the agent explores an unknown environment to maximize accumulated rewards.



Agent-environment Interaction

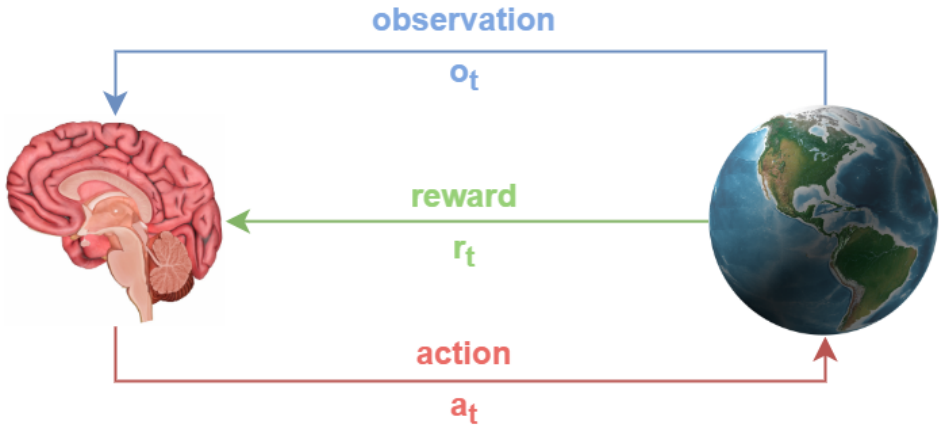
- In reinforcement learning (RL) the agent explores an unknown environment to maximize accumulated rewards.



- The agent learns to maximize their accumulated reward in a trial-and-error fashion, trying (exploring) different actions and observing outcomes.

Agent-environment Interaction

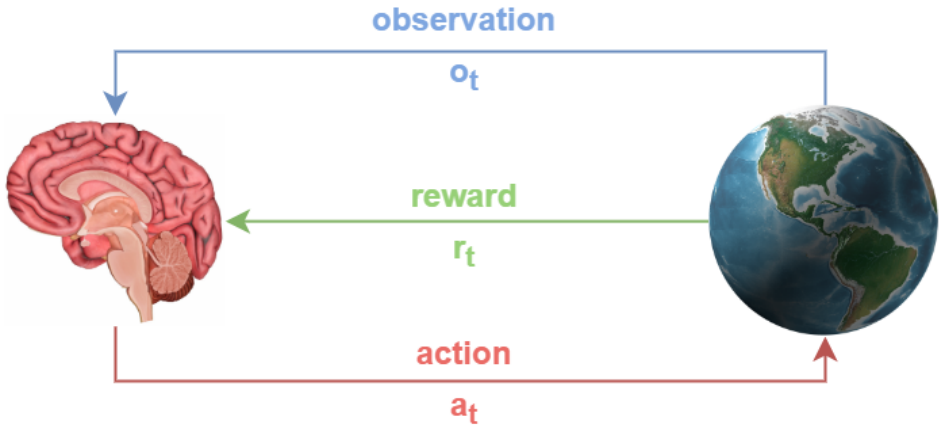
- In reinforcement learning (RL) the agent explores an unknown environment to maximize accumulated rewards.



- The agent learns to maximize their accumulated reward in a trial-and-error fashion, trying (exploring) different actions and observing outcomes.
- The more the agent interacts with the environment the more it learns and the better they can choose their actions (exploitation).

Agent-environment Interaction

- In reinforcement learning (RL) the agent explores an unknown environment to maximize accumulated rewards.



- The agent learns to maximize their accumulated reward in a trial-and-error fashion, trying (exploring) different actions and observing outcomes.
- The more the agent interacts with the environment the more it learns and the better they can choose their actions (exploitation).
- The exploration v.s exploitation trade-off is a key problem in RL. How does the agent know that it has collected enough data to starting exploiting what it knows?

Running Example (revisited)

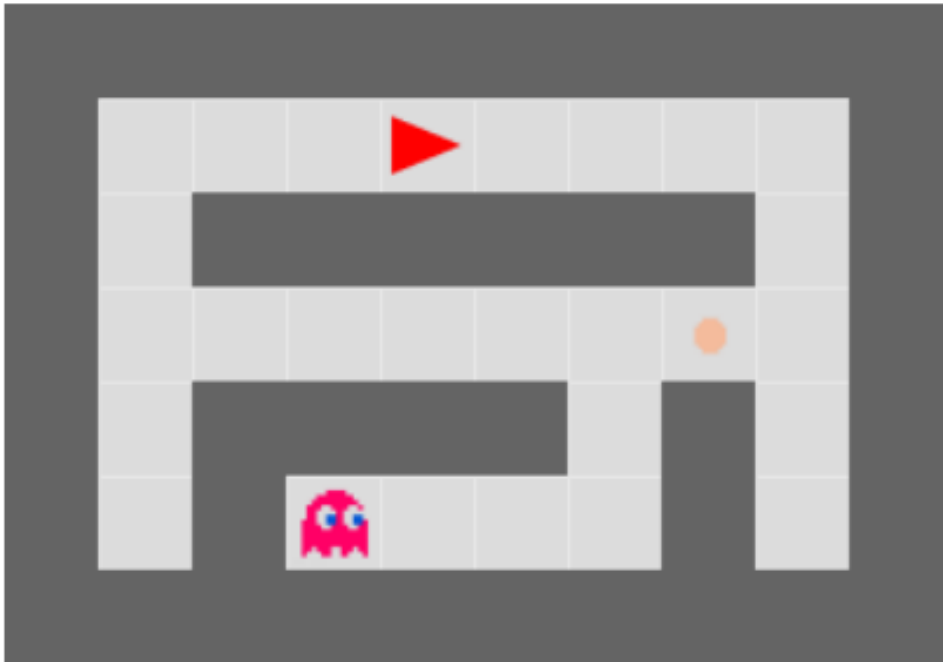


Figure: `MiniPacman` game.

Running Example (revisited)

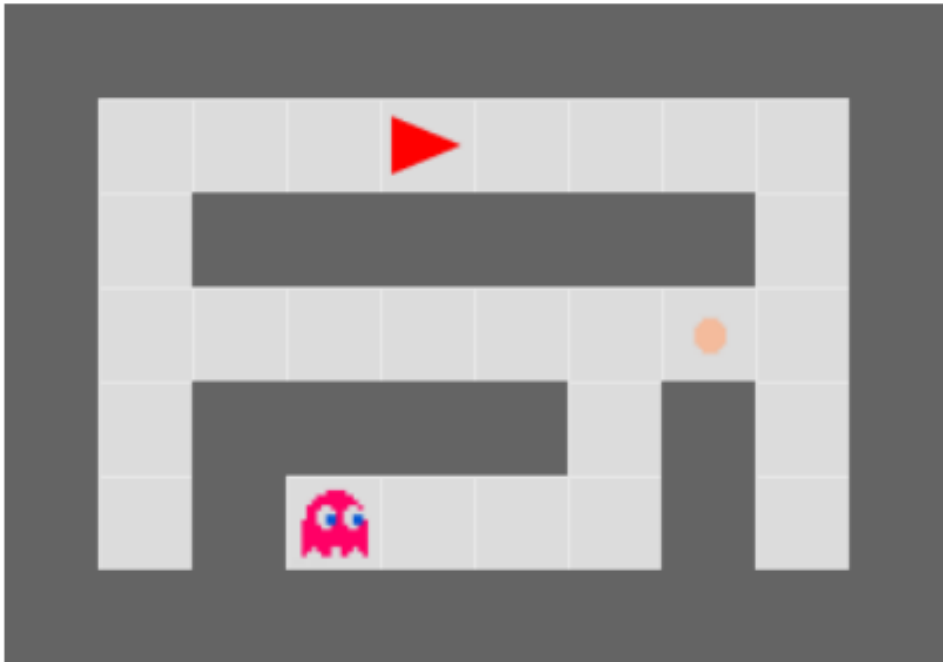


Figure: `MiniPacman` game.

Running Example (revisited)

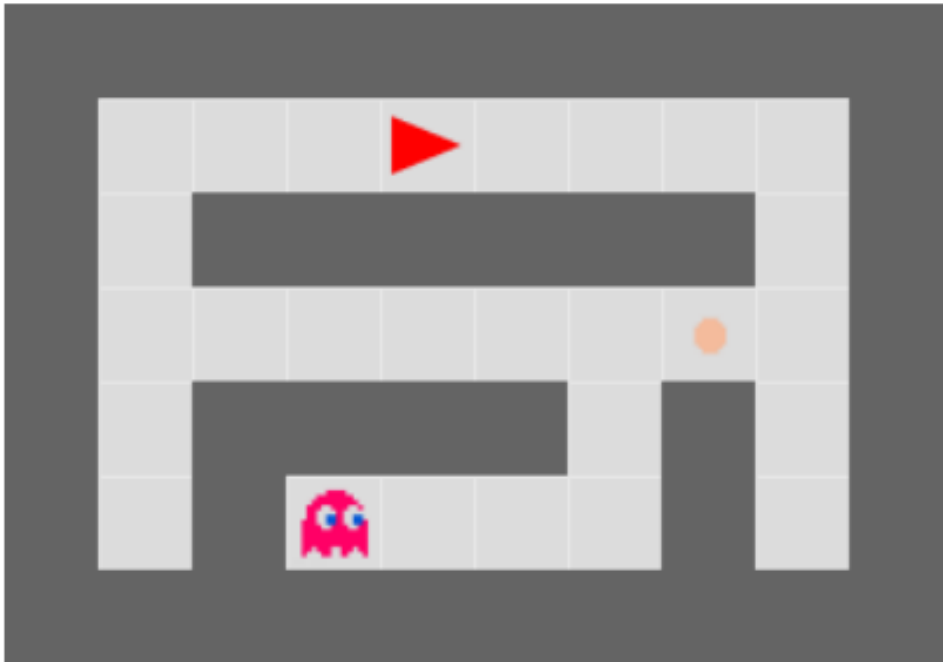


Figure: `MiniPacman` game.

- Observations: location (and direction) of the "pacman", "ghost" and "food".

Running Example (revisited)

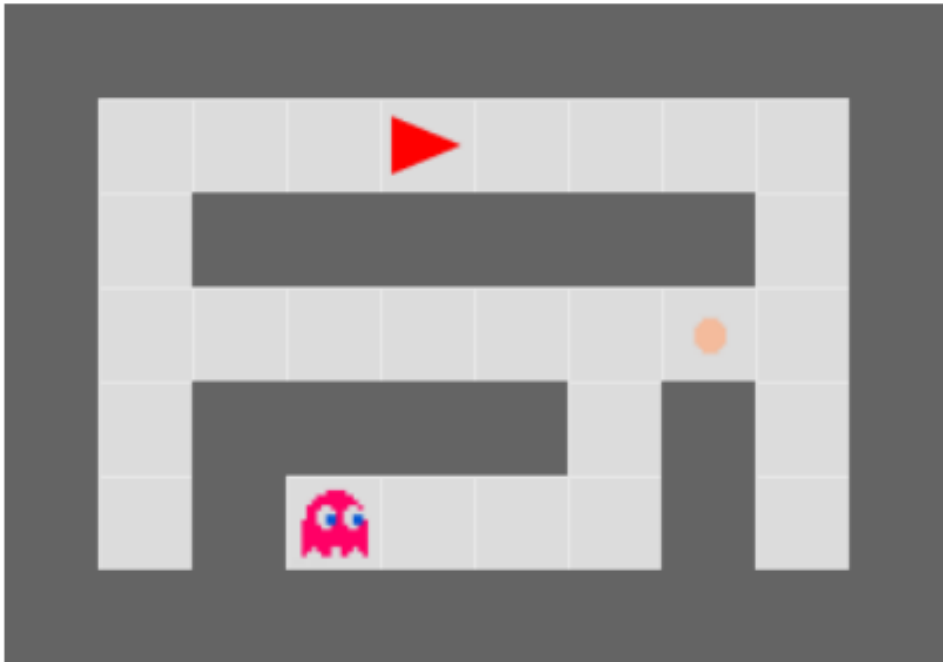


Figure: `MiniPacman` game.

- Observations: location (and direction) of the "pacman", "ghost" and "food".
- Actions: "up", "down", "left", "right", "continue".

Running Example (revisited)

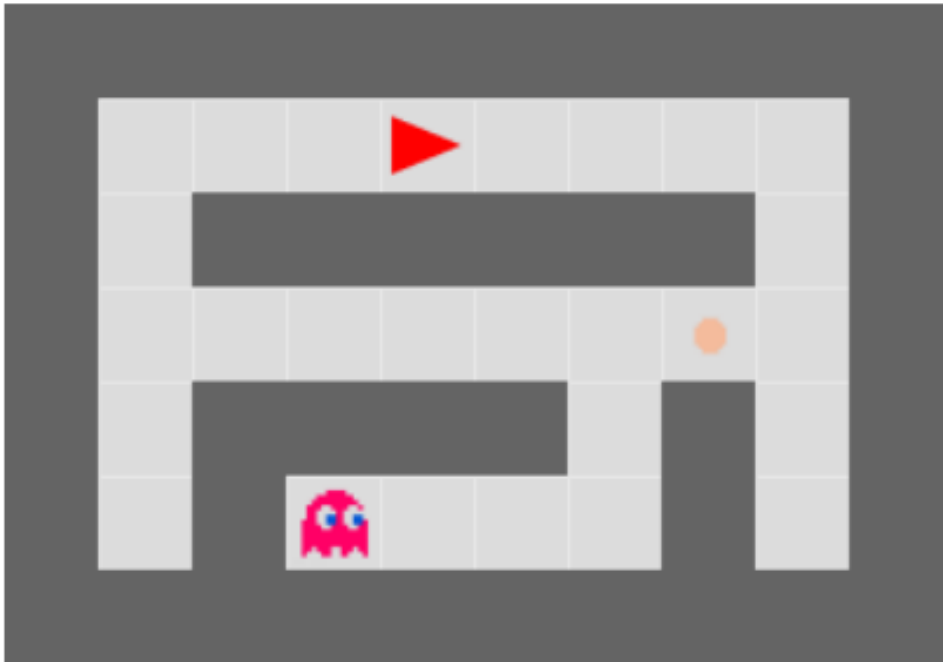


Figure: `MiniPacman` game.

- Observations: location (and direction) of the "pacman", "ghost" and "food".
- Actions: "up", "down", "left", "right", "continue".
- Reward: "+100 if the food is collected", "-1 otherwise".

Key Ideas

Modelling the environment

In RL, the environment is usually modelled as a Markov Decision Process, the probabilistic equivalent of Transition Systems.

Key Ideas

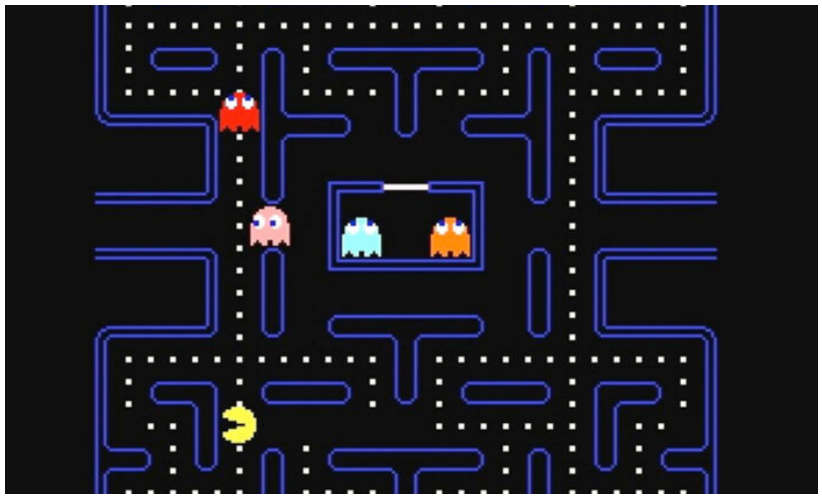
Modelling the environment

In RL, the environment is usually modelled as a Markov Decision Process, the probabilistic equivalent of Transition Systems.

Deep Reinforcement Learning

- Use *deep neural networks* to guide the agent's choices.
- Has enabled RL to tackle complex problems (e.g., Minecraft)
- Successful applications in games (e.g., AlphaGo), robotics, finance, ...
- One of the key components of LLMs (e.g., DeepSeek)

Reinforcement Learning in the Wild



Pac-Man

AlphaGo vs. Lee Sedol (Go)

Dactyl solving a Rubik's Cube

OpenAI Five (Dota 2)

What is Reinforcement Learning?
Markov Decision Processes

Markov Decision Processes: A First Example

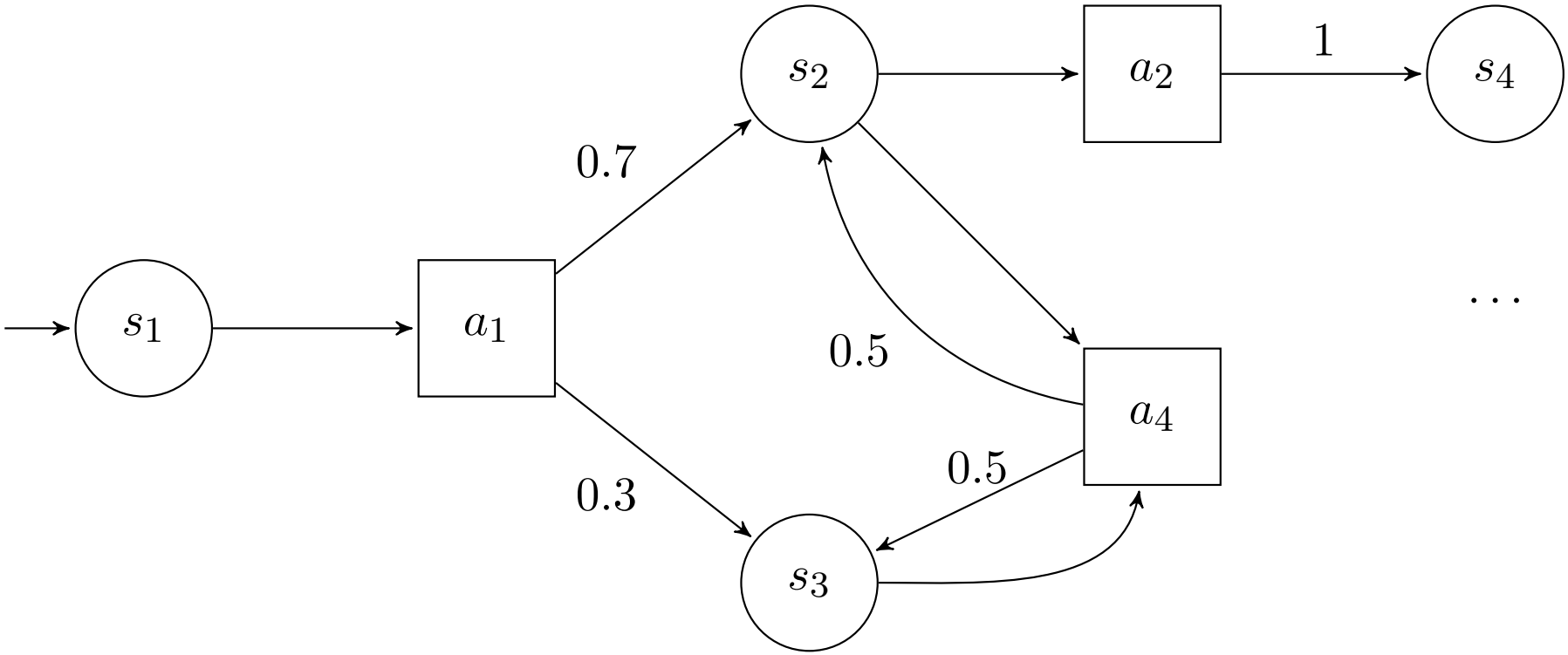


Figure: Example MDP

Markov Decision Processes: Formal Definition

Definition 1 (Markov Decision Process (MDP))

A **Markov Decision Process** is a tuple (S, s_0, A, P) where:

- S is a set of states,
- s_{init} is an initial state,
- $A(s)$ is a non-empty set of actions available from state s,
- P is an stochastic transition function: $P(\cdot \mid s, a)$ is a probability measure over next states in S from (s, a),

Markov Decision Processes: Formal Definition

Definition 1 (Markov Decision Process (MDP))

A **Markov Decision Process** is a tuple (S, s_0, A, P) where:

- S is a set of states,
- s_{init} is an initial state,
- $A(s)$ is a non-empty set of actions available from state s,
- P is an stochastic transition function: $P(\cdot \mid s, a)$ is a probability measure over next states in S from (s, a),

Why MDPs?

Markov Decision Processes: Formal Definition

Definition 1 (Markov Decision Process (MDP))

A **Markov Decision Process** is a tuple (S, s_0, A, P) where:

- S is a set of states,
- s_{init} is an initial state,
- $A(s)$ is a non-empty set of actions available from state s,
- P is an stochastic transition function: $P(\cdot \mid s, a)$ is a probability measure over next states in S from (s, a),

Why MDPs?

They are *versatile*, modelling probabilistic or deterministic sequential decision making problems, e.g., games, robotics, financial markets, LLMs.

Markov Decision Processes: Formal Definition

Definition 1 (Markov Decision Process (MDP))

A **Markov Decision Process** is a tuple (S, s_0, A, P) where:

- S is a set of states,
- s_{init} is an initial state,
- $A(s)$ is a non-empty set of actions available from state s,
- P is an stochastic transition function: $P(\cdot \mid s, a)$ is a probability measure over next states in S from (s, a),

Why MDPs?

They are *versatile*, modelling probabilistic or deterministic sequential decision making problems, e.g., games, robotics, financial markets, LLMs.

They also have the convenient **Markov property**: the next state transition only depends on the current state and action $\rightarrow$ we don't need to model history!

Markov Decision Processes: Paths

Definition 2 (Paths)

A *finite path (resp. infinite)* in an MDP is a finite (resp. infinite) sequence $\rho = s_0 a_0 s_1 a_1 \ldots$ such that for all i,

- $a_i \in A(s_i)$
- and s_i is in the support of $P(s, a)$.

We let $\text{FinPaths}(\mathcal{M})$ denote the set of all finite paths of $\mathcal{M}$.

Markov Decision Processes: Paths

Definition 2 (Paths)

A *finite path (resp. infinite)* in an MDP is a finite (resp. infinite) sequence $\rho = s_0 a_0 s_1 a_1 \ldots$ such that for all i,

- $a_i \in A(s_i)$
- and s_i is in the support of $P(s, a)$.

We let $\text{FinPaths}(\mathcal{M})$ denote the set of all finite paths of $\mathcal{M}$.

Paths: Examples

In the MDP from Example 1,

- examples of finite paths: $\rho_1 = s_1 a_1 s_3 a_4 s_2$, $s_1 a_1 s_2 a_4 s_3$
- examples of infinite paths: $\rho_2 = s_1 a_1 (s_3 a_4)^\omega$, $\rho_3 = s_1 a_1 (s_3 a_4 s_2 a_4)^\omega$

Policies

Definition 3 (Policy)

A **policy** π in an MDP $\mathcal{M} = (S, s_{init}, A, P)$ maps every finite path $\rho = s_0, a_0, \ldots, s_n$ of $\mathcal{M}$ to a probability distribution $\pi(\rho)$ over $A(s_n)$.

- It is *deterministic* if, for any finite path ρ, $\pi(\rho)$ is a Dirac distribution (assigns probability 1 to precisely one actions), and *stochastic* otherwise.
- It is *memoryless* if, for any finite path $\rho = s_0, a_0, \ldots, s_n$, $\pi(\rho)$ only depends on s_n.

Example of a Stochastic Policy

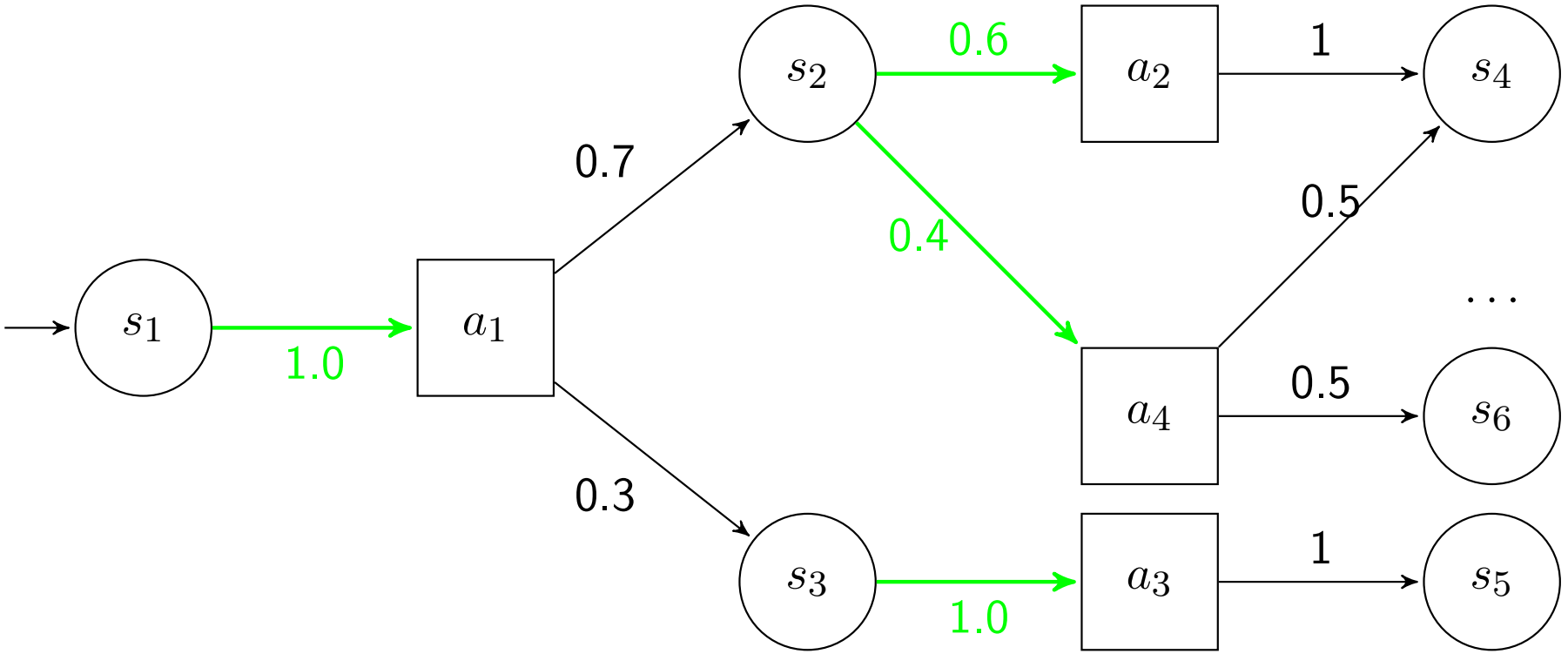


Figure: Example: Stochastic Policy

Example of a Deterministic Policy

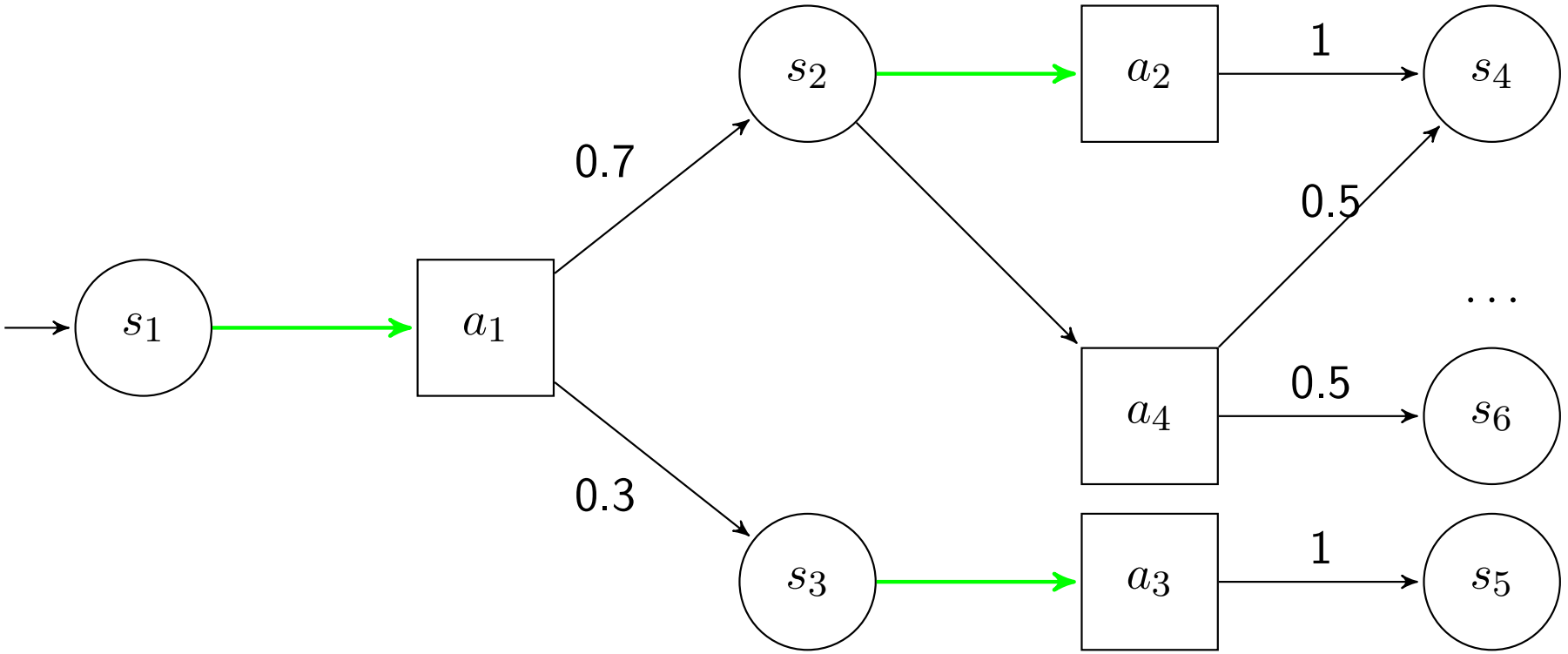


Figure: Example: Deterministic Policy

Policies and Probability Measures

Definition 4 (Induced Probability Measure)

A policy π induces a **probability measure** $\Pr^{\pi}_{\mathcal{M}}$ over infinite paths of $\mathcal{M}$

Policies and Probability Measures

Definition 4 (Induced Probability Measure)

A policy π induces a **probability measure** $\Pr^{\pi}_{\mathcal{M}}$ over infinite paths of $\mathcal{M}$

Induced Probability Measure: examples

Ex. 2 Let E is the set of all paths that begin by $s_1 a_1 s_2 a_4 s_2$.
Then, $\Pr^{\pi}_{\mathcal{M}}(E) = 0.7 \times 0.4 \times 0.5$.

Ex. 3 Let E' is the set of all paths that begin by $s_1 a_1 s_2 a_2 s_4$.
Then, $\Pr^{\pi}_{\mathcal{M}}(E) = 0.7$.

What is Reinforcement Learning?
Reward Optimal Policies

Markov Decision Processes with Rewards

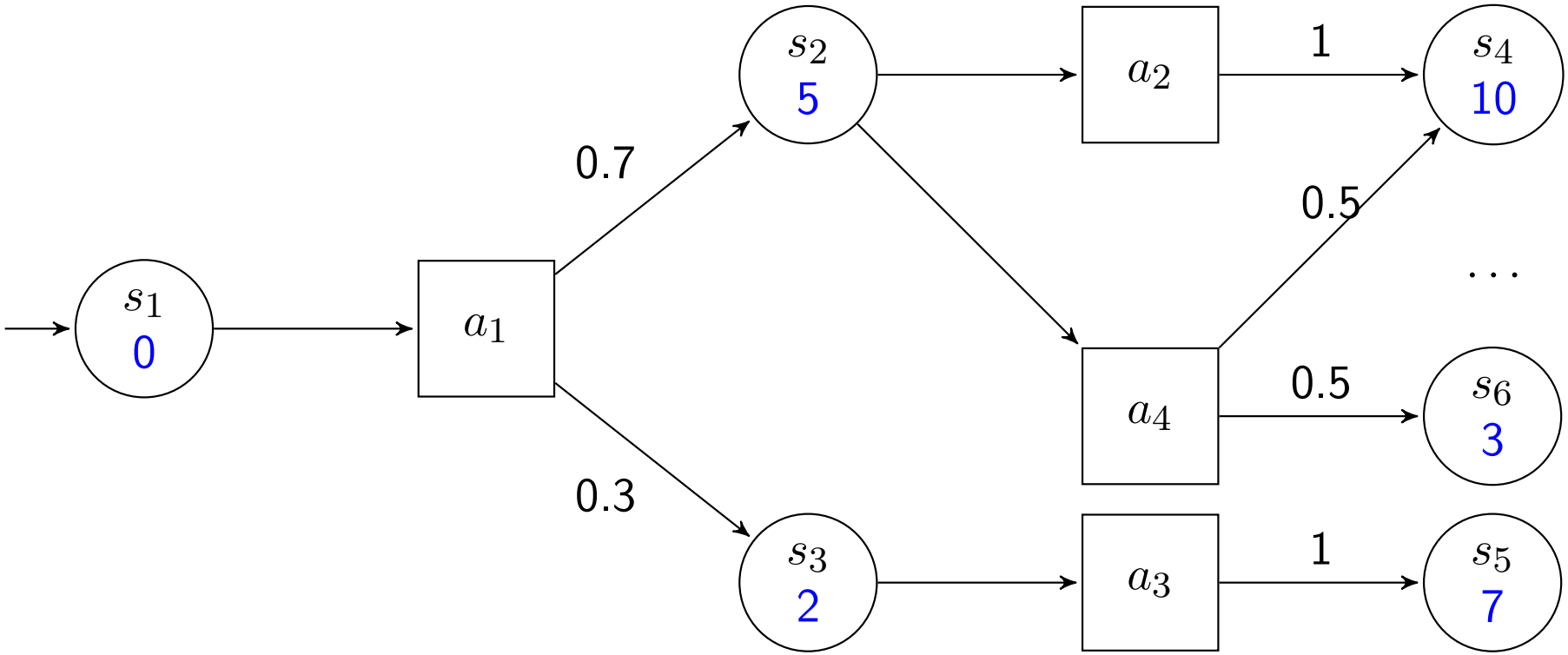
Definition 5 (MDP with Rewards)

An **MDP with rewards** is an MDP $\mathcal{M} = (S, s_0, A, P, R, \gamma)$ equipped with a reward function $R : S \times A \to \mathbb{R}$ that maps every tuple (s, a) such that $a \in A(s)$ to real-valued scalar reward in $\mathbb{R}$, and a discount factor $\gamma \in [0, 1]$ that discounts future rewards.

Remark

In reinforcement learning the environment in usually modelled as an MDP with rewards.

MDP with Rewards: Example



Reward Optimal Policy

Definition 6 (Expected Cumulative Reward)

Given an MDP $\mathcal{M}$ with rewards (and discount factor $\gamma \in [0, 1]$), the **expected cumulative (discounted) reward** of a policy π is defined as

$$\mathbb{E}_{s_t \sim P, a_t \sim \pi} \sum_{t=0} \gamma^t R(s_t, a_t)$$

Reward Optimal Policy

Definition 6 (Expected Cumulative Reward)

Given an MDP $\mathcal{M}$ with rewards (and discount factor $\gamma \in [0, 1]$), the **expected cumulative (discounted) reward** of a policy π is defined as

$$\mathbb{E}_{s_t \sim P, a_t \sim \pi} \sum_{t=0} \gamma^t R(s_t, a_t)$$

Definition 7 (Optimal Policy)

A policy is **optimal** if its expected cumulative reward is maximal amongst all available policies:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s_t \sim P, a_t \sim \pi} \sum_{t=0} \gamma^t R(s_t, a_t)$$

Reward Optimal Policy

Definition 6 (Expected Cumulative Reward)

Given an MDP $\mathcal{M}$ with rewards (and discount factor $\gamma \in [0, 1]$), the **expected cumulative (discounted) reward** of a policy π is defined as

$$\mathbb{E}_{s_t \sim P, a_t \sim \pi} \sum_{t=0} \gamma^t R(s_t, a_t)$$

Definition 7 (Optimal Policy)

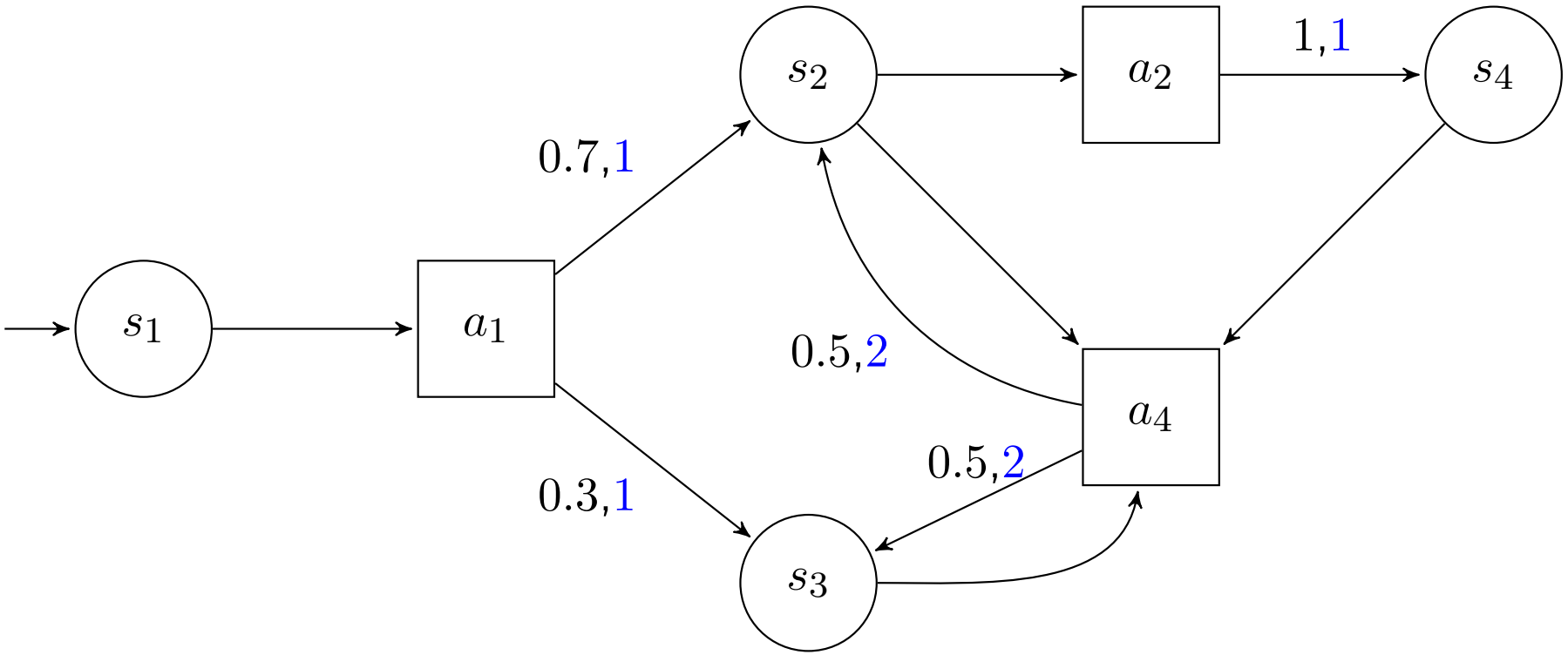
A policy is **optimal** if its expected cumulative reward is maximal amongst all available policies:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{s_t \sim P, a_t \sim \pi} \sum_{t=0} \gamma^t R(s_t, a_t)$$

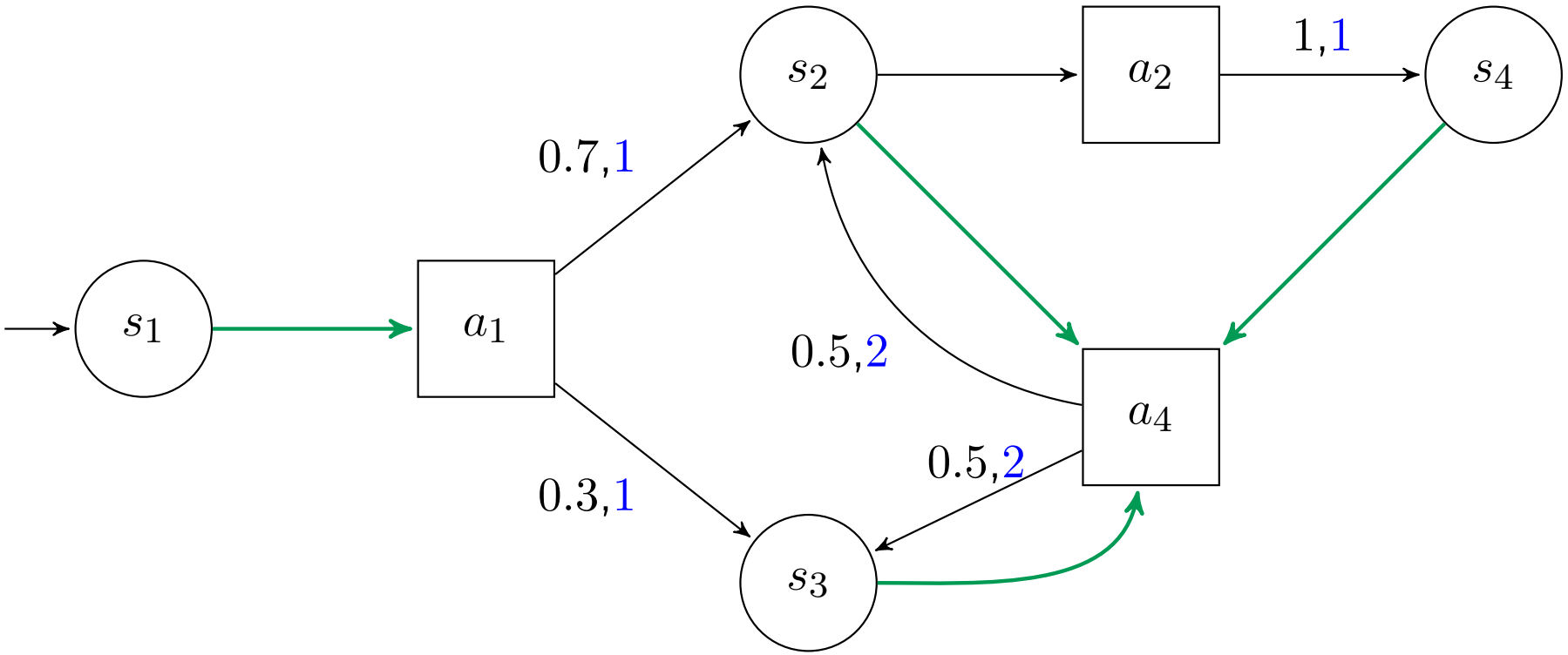
Remark

Memoryless policies are sufficient for reward optimal policies in MDPs.

Optimal Policy: Example



Optimal Policy: Example



Optimal policy π^* with expected cumulative discounted return $1 + 2(\gamma + \gamma^2 + \ldots) = 1 + 2\frac{\gamma}{1-\gamma}$.

Finding Optimal Policies: Overview

The MDP is Known

When the MDP is known, common approaches to finding the reward optimal policy include:

- Linear programming:
  - Objective: maximize expected cumulative reward.
  - Constraints: linear inequalities encoding the Bellman equations.
- Dynamic programming: value iteration or direct policy iteration.

Finding Optimal Policies: Overview

The MDP is Known

When the MDP is known, common approaches to finding the reward optimal policy include:

- Linear programming:
  - Objective: maximize expected cumulative reward.
  - Constraints: linear inequalities encoding the Bellman equations.
- Dynamic programming: value iteration or direct policy iteration.

The MDP is Unknown

In reinforcement learning the MDP is usually known. Optimal policies can still be found via more sophisticated techniques:

- Q-learning: stochastic approximation of value iteration.
- Deep reinforcement learning: deep Q-learning, policy gradient.

Standard RL Problems: Game Playing (Atari)

Atari game: Alien

- **Game Overview:** Navigate a maze, avoid the aliens, and destroy their eggs.
- **Visual Input:** Agents typically learn from raw pixel data.
- **Actions:** Joystick movements and flamethrower button.

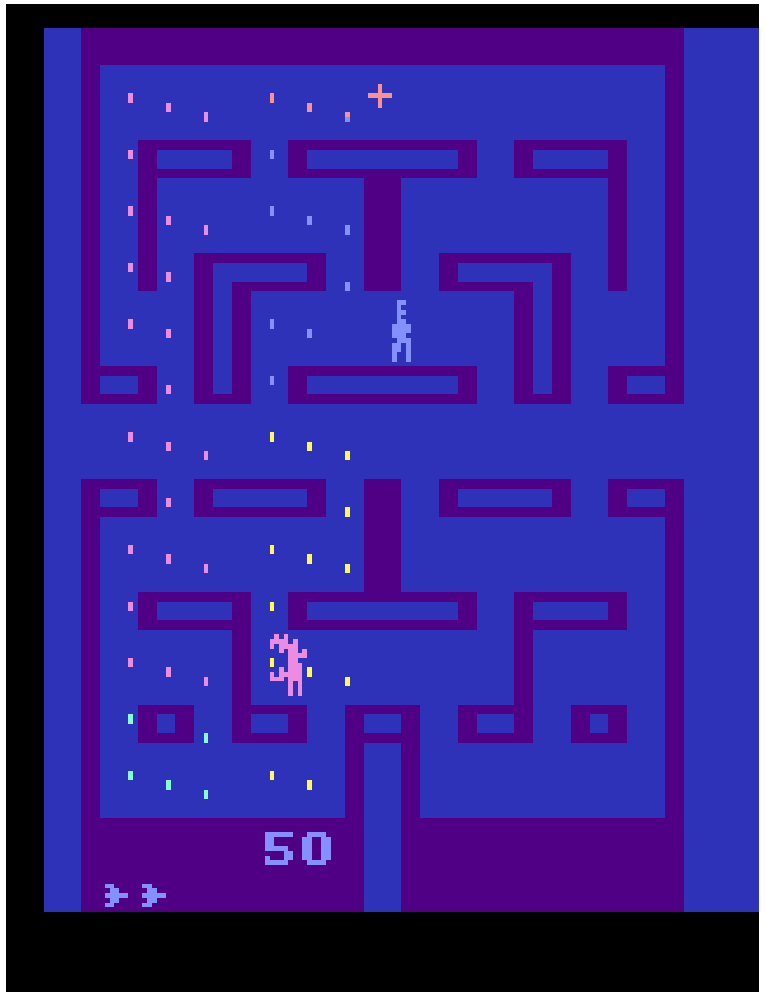


Figure: Atari game: Alien

Standard RL Problems: Continuous Control

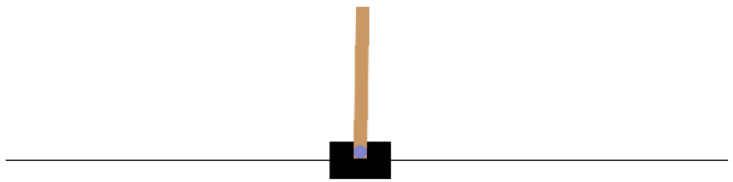


Figure: The Cart Pole environment

Cartpole

- **Actions:** moving the cart left or right.
- **Reward:** $+1$ if the pole stays up; 0 if it falls.
- **Goal:** balancing the pole so that it stays upright.

Standard RL Objective: Robotics

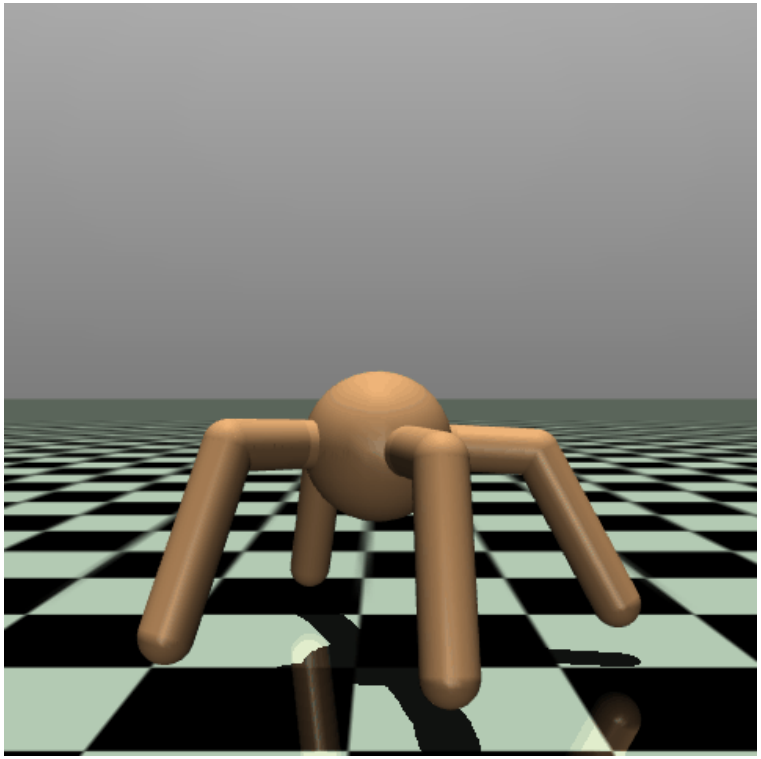


Figure: MuJoCo "Ant" model

MuJoCo: Ant Environment

- **Continuous Control:** 8 controllable joints.
- **Complex Dynamics:** Balance, forward velocity, and energy efficiency.
- **Observation Space:** Joint angles, velocities, and body position.
- **Benchmark Use:** Popular for testing policy-gradient RL algorithms.

What is Reinforcement Learning?
Constrained Reinforcement Learning

Constrained Reinforcement Learning

The most popular way of modelling the safe reinforcement learning problem is via constraints

Constrained Reinforcement Learning

The most popular way of modelling the safe reinforcement learning problem is via constraints

Definition 8 (Constrained MDP)

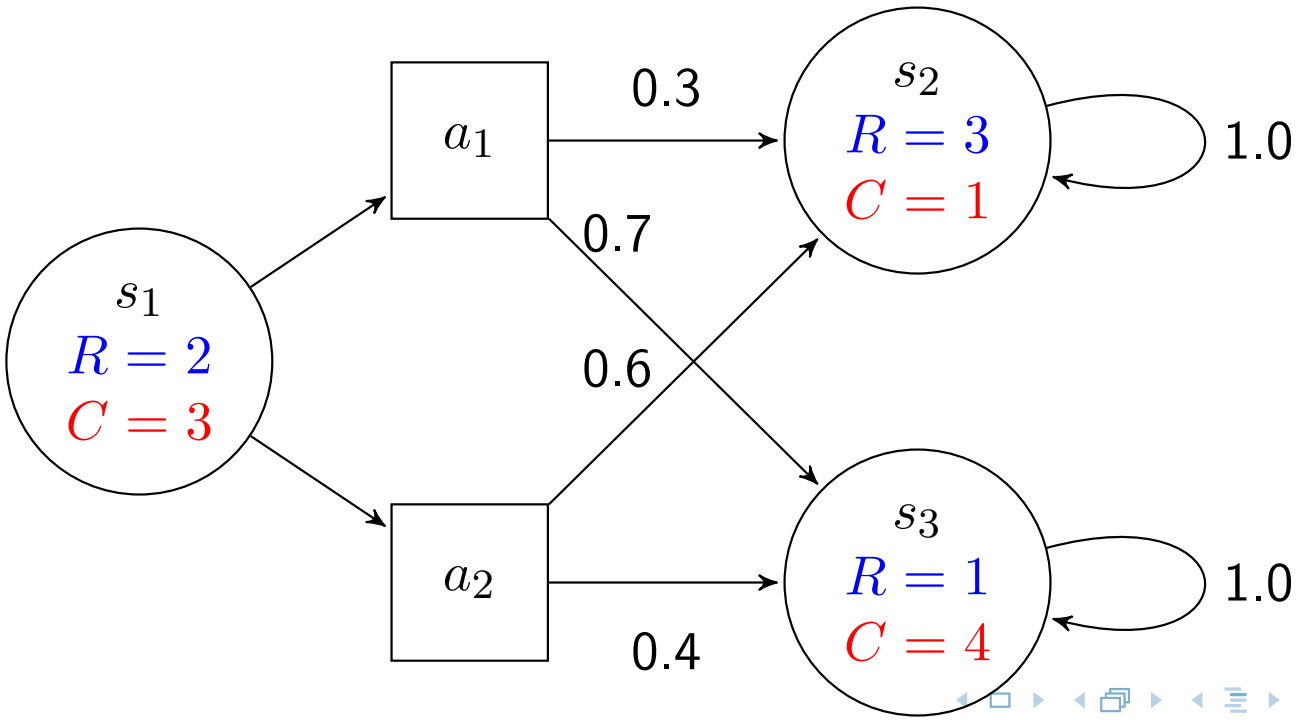
Constrained MDPs extend *MDPs with rewards* with a additional scalar **cost function** $C : S \times A \to \mathbb{R}$ and separate discount factor $\gamma_c \in [0, 1]$.

Constrained Reinforcement Learning

The most popular way of modelling the safe reinforcement learning problem is via constraints

Definition 8 (Constrained MDP)

Constrained MDPs extend *MDPs with rewards* with a additional scalar **cost function** $C : S \times A \to \mathbb{R}$ and separate discount factor $\gamma_c \in [0, 1]$.



Constrained RL: Problem Formulation with CMDPs

Problem Statement

Input: CMDP $\mathcal{M}$ (with costs C and disciunt γ_c), safety threshold d,
Question: Find a policy π^* such that

- π^* is optimal amongst all policies π whose expected cumulative discounted cost is $\leq d$:

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t} \gamma^t R(s_t, a_t)\right] \quad \text{subject to} \quad \mathbb{E}_{\pi}\left[\sum_{t} \gamma_c^t C(s_t, a_t)\right] \leq d$$

Typical Methods to Solve CMDPs

- **Linear Programming:** gives exact solutions, but becomes intractable between $\sim$ 10k to 100k states.

Typical Methods to Solve CMDPs

- **Linear Programming:** gives exact solutions, but becomes intractable between $\sim$ 10k to 100k states.
- **Primal methods:** usually actor-critic frameworks where the actor also uses the expected cost gradient to enforce constraint satisfaction when needed (usually works for $\gamma_c < 1$ or finite horizon cost).

Typical Methods to Solve CMDPs

- **Linear Programming:** gives exact solutions, but becomes intractable between $\sim$ 10k to 100k states.
- **Primal methods:** usually actor-critic frameworks where the actor also uses the expected cost gradient to enforce constraint satisfaction when needed (usually works for $\gamma_c < 1$ or finite horizon cost).
- **Primal-dual methods:** reducing to an unconstrained problem with an additional dual variable λ representing a trade-off between reward and constraint (usually works for $\gamma = \gamma_c < 1$, in practice also finite horizon cost)

Primal-dual methods

Key Idea: Lagrangian Relaxation

This constrained optimization problem can be reformulated using the following Lagrangian objective:

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] - \lambda \left(\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \right] - d \right)$$

Primal-dual methods

Key Idea: Lagrangian Relaxation

This constrained optimization problem can be reformulated using the following Lagrangian objective:

$$\mathcal{L}(\pi, \lambda) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right] - \lambda \left(\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t) \right] - d \right)$$

Optimization Procedure

1. **Primal Update:** Update policy π to maximize the Lagrangian $\mathcal{L}(\pi, \lambda)$ for a fixed λ with a step of your favourite RL algorithm.
2. **Dual Update:** Adjust λ to penalize constraint violations:

$$\lambda \leftarrow \max(0, \lambda + \eta_\lambda \cdot (\mathbb{E}_\pi[C] - p)),$$

where η_λ is the learning rate for λ.

Primal v.s. Primal-dual (pros and cons)

Primal Methods

- **Pros:** Reliable performance, high constraint satisfaction rate empirically.

Primal-dual Methods

- **Pros:** Simple and flexible, i.e.m can be paired with any RL algorithm easily (e.g., off-policy, model-based).
- **Cons:**
  - Brittle and slow convergence: dual variable must converge.
  - Very sensitive to hyperparameters, e.g., like dual variable initialization or learning rate (heavy per-environment tuning is required).
  - Often high constraint violation during training with parameter updates fluctuating between the feasible and infeasible region.

Primal v.s. Primal-dual (pros and cons)

Primal Methods

- **Pros:** Reliable performance, high constraint satisfaction rate empirically.
- **Cons:** On-policy, convergence and policy improvement relies on non-trivial assumptions, often more challenging to implement and computationally heavier.

Primal v.s. Primal-dual (pros and cons)

Primal Methods

- **Pros:** Reliable performance, high constraint satisfaction rate empirically.
- **Cons:** On-policy, convergence and policy improvement relies on non-trivial assumptions, often more challenging to implement and computationally heavier.

Primal-dual Methods

- **Pros:** Simple and flexible, i.e.m can be paired with any RL algorithm easily (e.g., off-policy, model-based).

Primal v.s. Primal-dual (pros and cons)

Primal Methods

- **Pros:** Reliable performance, high constraint satisfaction rate empirically.
- **Cons:** On-policy, convergence and policy improvement relies on non-trivial assumptions, often more challenging to implement and computationally heavier.

Primal-dual Methods

- **Pros:** Simple and flexible, i.e.m can be paired with any RL algorithm easily (e.g., off-policy, model-based).
- **Cons:**
  - Brittle and slow convergence: dual variable must converge.
  - Very sensitive to hyperparameters, e.g., like dual variable initialization or learning rate (heavy per-environment tuning is required).
  - Often high constraint violation during training with parameter updates fluctuating between the feasible and infeasible region.

Overall Limitations: CMDPs

Why CMDP Safety is not the Whole Story

- CMDPs are a useful and flexible framework, but for safety they often express "badness" as accumulated scalar cost. This can obscure the semantics of requirements such as "never collide".

Overall Limitations: CMDPs

Why CMDP Safety is not the Whole Story

- CMDPs are a useful and flexible framework, but for safety they often express "badness" as accumulated scalar cost. This can obscure the semantics of requirements such as "never collide".
- Discounted safety costs also do not meaningfully constrain limiting behaviour: a policy can push violations far into the future and reduce discounted cost without satisfying an invariant.

Overall Limitations: CMDPs

Why CMDP Safety is not the Whole Story

- CMDPs are a useful and flexible framework, but for safety they often express "badness" as accumulated scalar cost. This can obscure the semantics of requirements such as "never collide".
- Discounted safety costs also do not meaningfully constrain limiting behaviour: a policy can push violations far into the future and reduce discounted cost without satisfying an invariant.
- Expected-cost constraints are also soft in the sense that they control an expectation, not every trajectory.

Where do we go from here?

Shielding: filters actions proposed by the agent before they reach the environment, guaranteeing safety-by-construction.

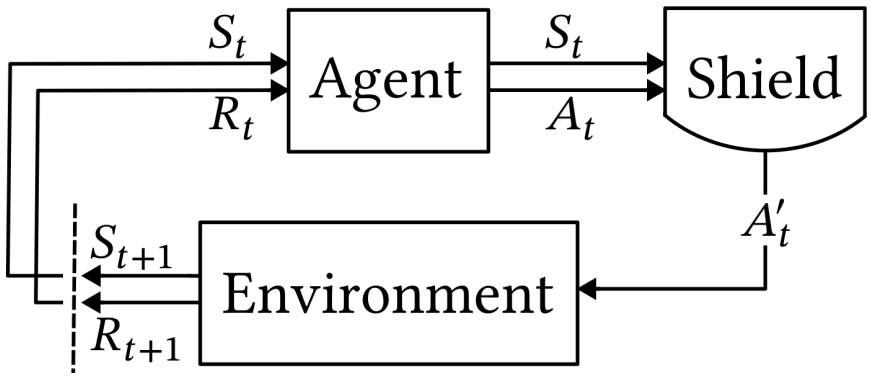


Figure: Post-posed shielding, replaces or modifies the agent's actions A_t with safe alternative A'_t if necessary.

Shielding: Overview

What Shielding Buys Us

- The guarantee strength depends on the exact setting and assumptions.
  - Known exact safety dynamics: hard (almost sure) or probabilistic guarantees.
  - Conservative safety abstraction: worst-case and adversarial safety guarantees.
  - Known transition probabilities: can give concrete probabilistic guarantees.
  - Learned model/transition probabilities: give high-confidence or empirically calibrated safety guarantees.

Shielding: Overview

What Shielding Buys Us

- The guarantee strength depends on the exact setting and assumptions.
  - Known exact safety dynamics: hard (almost sure) or probabilistic guarantees.
  - Conservative safety abstraction: worst-case and adversarial safety guarantees.
  - Known transition probabilities: can give concrete probabilistic guarantees.
  - Learned model/transition probabilities: give high-confidence or empirically calibrated safety guarantees.
- Compared with Lagrange/CMDP methods, shielding gives clearer semantics and can enforce hard safety rather than merely penalizing violations.

A Quick Primer on Linear Temporal Logic

Specifying Tasks via LTL: Motivating Gridworld Example

A robot operating in a 2D gridworld must complete some task

A robot must, with maximal probability:

1. Collect at least 3 items.
2. Then press a special button (once).
3. *After* pressing the button, avoid a dangerous zone *forever*.

Specifying Tasks via LTL: Motivating Gridworld Example

A robot operating in a 2D gridworld must complete some task

A robot must, with maximal probability:

1. Collect at least 3 items.
2. Then press a special button (once).
3. *After* pressing the button, avoid a dangerous zone *forever*.

Question

Can we provide the robot with an LTL formula for this task ?

Specifying Tasks via LTL: Motivating Gridworld Example

A robot operating in a 2D gridworld must complete some task

A robot must, with maximal probability:

1. Collect at least 3 items.
2. Then press a special button (once).
3. *After* pressing the button, avoid a dangerous zone *forever*.

Question

Can we provide the robot with an LTL formula for this task ?

Answer

Yes but how do we then design a reward function for so that the reward optimal policy completes this task with maximal probability

Specifying Tasks via LTL: Motivating Gridworld Example

A robot operating in a 2D gridworld must complete some task

A robot must, with maximal probability:

1. Collect at least 3 items.
2. Then press a special button (once).
3. *After* pressing the button, avoid a dangerous zone *forever*.

Question

Can we provide the robot with an LTL formula for this task ?

Answer

Yes but how do we then design a reward function for so that the reward optimal policy completes this task with maximal probability

Remark

We also somehow need to reason about time. Has the robot already collected the 3 items?

Linear Temporal Logic (LTL): Syntax

Temporal Operators

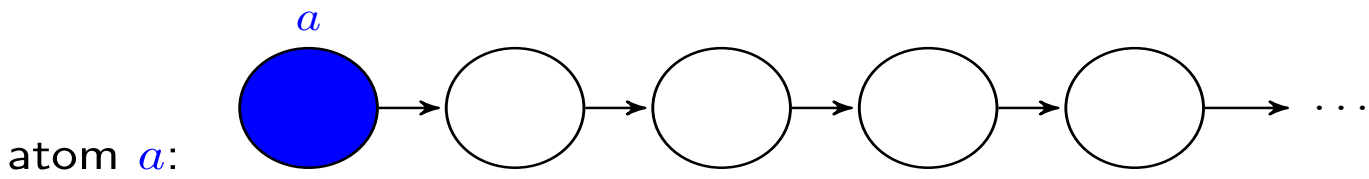
- $\mathbf{G}\varphi$ (Globally): φ will *always* be true in the future.
- $\mathbf{F}\varphi$ (Eventually): φ will *eventually* be true.
- $\mathbf{X}\varphi$ (Next): φ is true *at the next timestep*.
- $\varphi\mathbf{U}\psi$ (Until): ψ will *eventually* become true, and φ will be true *until that happens*.

Definition 9 (LTL Syntax)

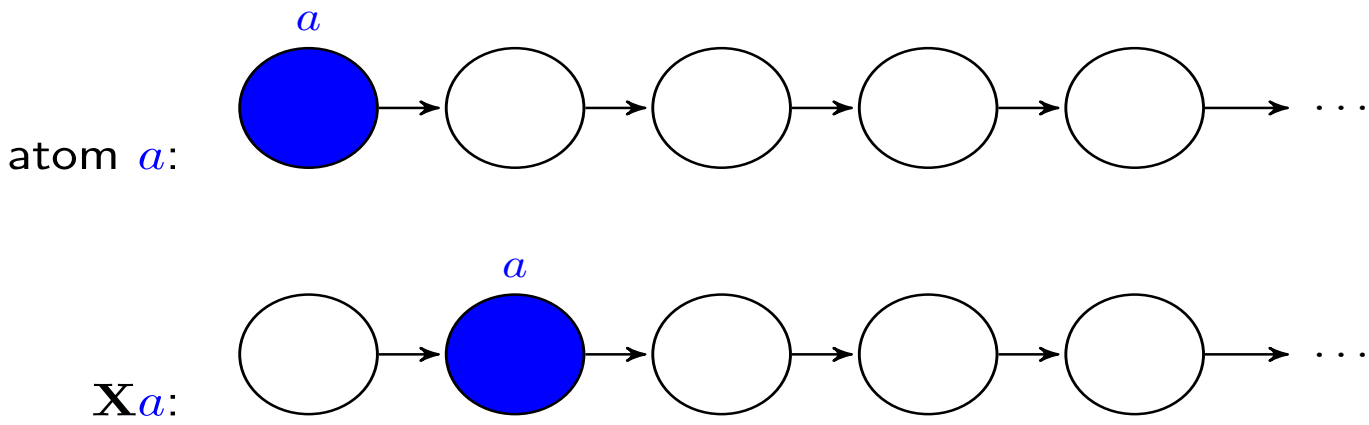
- Atomic propositions: $a, b, c, \ldots$
- Boolean operators: $\neg\varphi$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \rightarrow \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$,
- Temporal operators: $\mathbf{X}\phi$, $\mathbf{F}\phi$, $\mathbf{G}\phi$, $\phi\mathbf{U}\psi$

- Examples: $\mathbf{G}(\neg b)$, $a \vee \mathbf{F}b$, $\mathbf{F}(a\mathbf{U}(\neg(\mathbf{X}b)))$
- Suitable to express invariants, (conditional) safety, reachability, reach-avoidance, maintenance, fairness, etc.

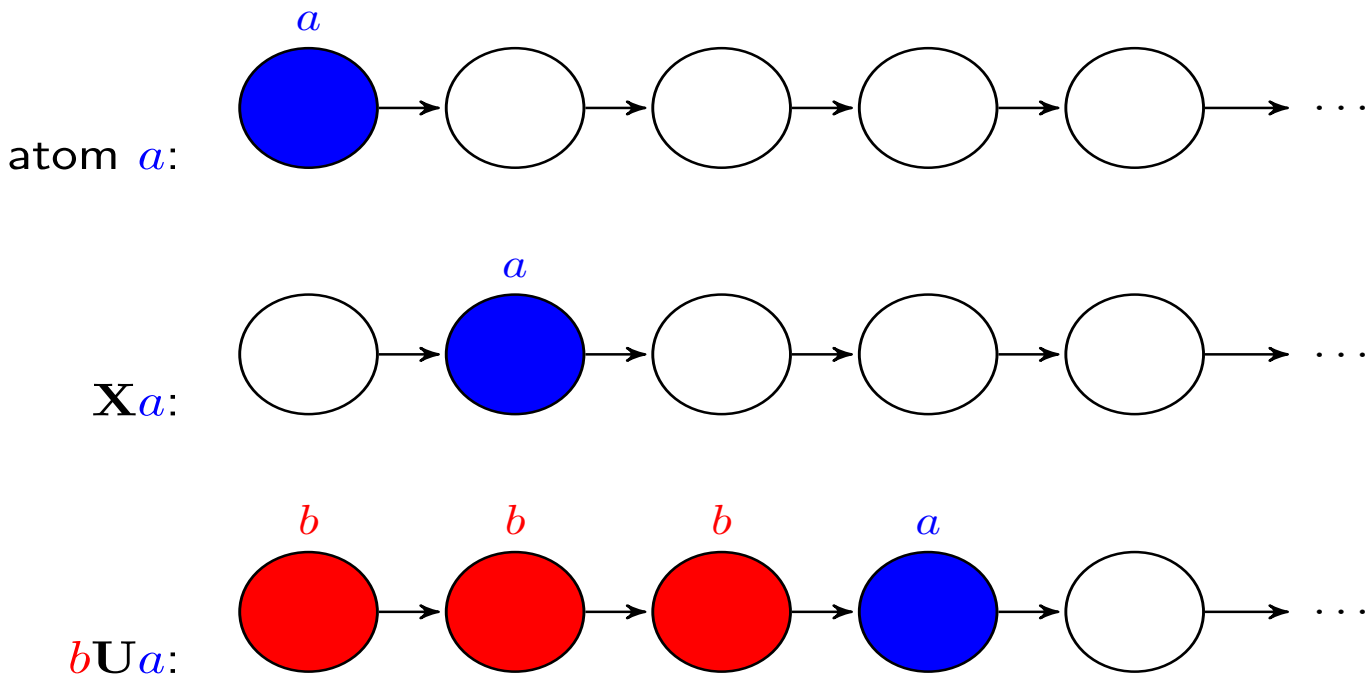
Linear Temporal Logic: semantics (intuition)



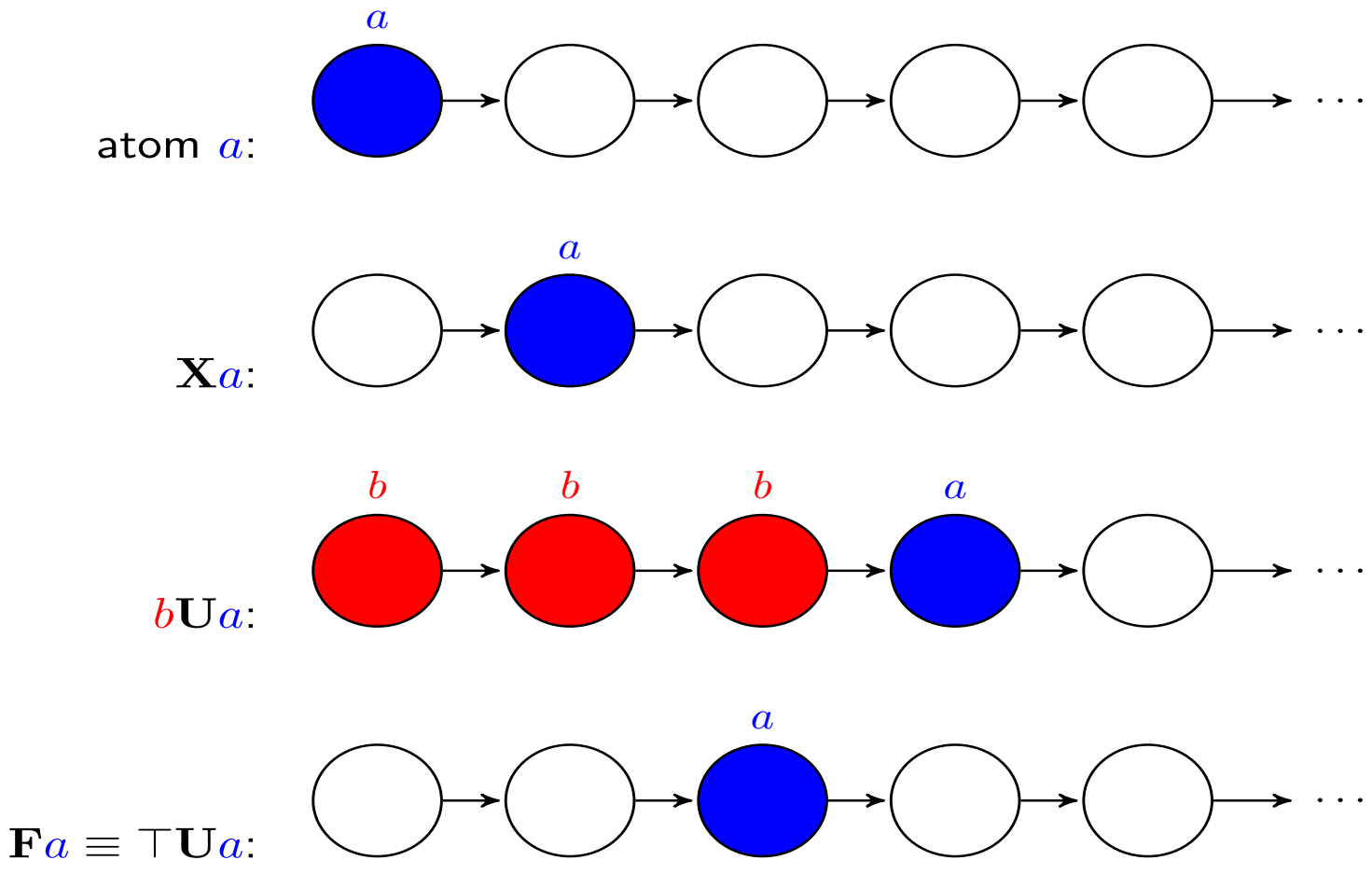
Linear Temporal Logic: semantics (intuition)



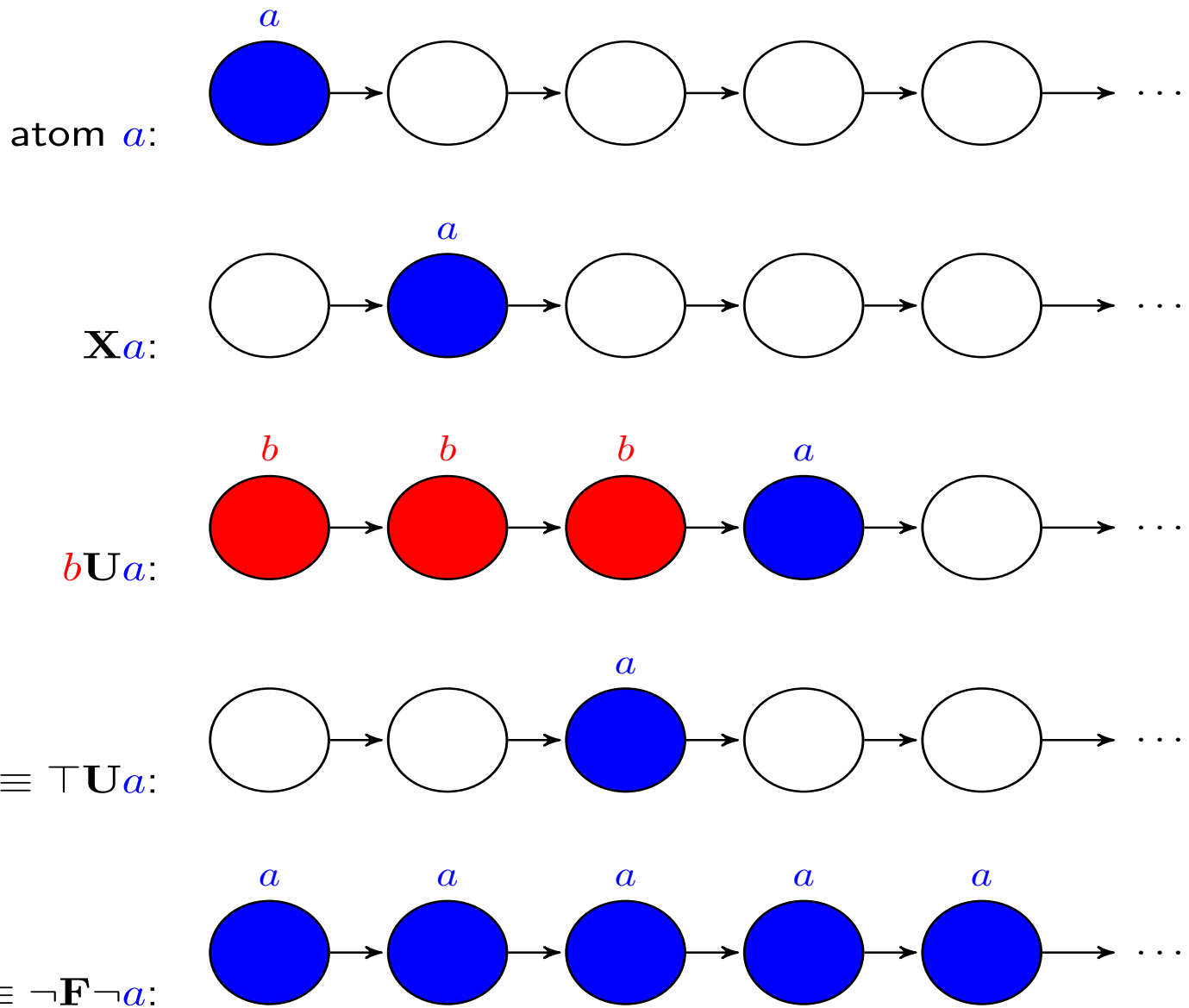
Linear Temporal Logic: semantics (intuition)



Linear Temporal Logic: semantics (intuition)



Linear Temporal Logic: semantics (intuition)



Linear Temporal Logic: Semantics (Infinite Words)

Definition 10 (LTL Semantics)

For any word $w \in (2^{AP})^\omega$, we define the satisfaction relation $w \models \varphi$ as follows:

$$
\begin{aligned}
w \models a \quad &\text{iff} \quad a \in w[0] \\
w \models \neg\varphi \quad &\text{iff} \quad w \not\models \varphi. \\
w \models \varphi_1 \vee \varphi_2 \quad &\text{iff} \quad w \models \varphi_1 \text{ or } w \models \varphi_2. \\
w \models X\varphi \quad &\text{iff} \quad w[1, +\infty] \models \varphi. \\
w \models G\varphi \quad &\text{iff} \quad \forall j, w[j, +\infty] \models \varphi. \\
w \models F\varphi \quad &\text{iff} \quad \exists j, w[j, +\infty] \models \varphi. \\
w \models \varphi_1 U \varphi_2 \quad &\text{iff} \quad \exists j, \begin{cases} w[j, +\infty] \models \varphi_2 \\ \text{and } \forall k < j, w[k, +\infty] \models \varphi_1. \end{cases}
\end{aligned}
$$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$
4. $\emptyset^\omega \models \mathbf{G}\neg b$
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$
4. $\emptyset^\omega \models \mathbf{G}\neg b$
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$ True
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$
4. $\emptyset^\omega \models \mathbf{G}\neg b$
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$ True
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$ True
4. $\emptyset^\omega \models \mathbf{G}\neg b$
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$ True
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$ True
4. $\emptyset^\omega \models \mathbf{G}\neg b$ True
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$ True
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$ True
4. $\emptyset^\omega \models \mathbf{G}\neg b$ True
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$ True
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$ True
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$ True
4. $\emptyset^\omega \models \mathbf{G}\neg b$ True
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$ True
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$ False
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$ True
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$ True
4. $\emptyset^\omega \models \mathbf{G}\neg b$ True
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$ True
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$ False
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$ False
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$

Semantics of Linear Temporal Logic: Examples

Are the following true?

1. $(\{a\}\{a,b\})^\omega \models \mathbf{G}b$ False
2. $(\{a\}\{a,b\})^\omega \models \mathbf{G}a$ True
3. $(\{a\}\{a,b\})^\omega \models \mathbf{F}b$ True
4. $\emptyset^\omega \models \mathbf{G}\neg b$ True
5. $(\{b\}\emptyset)^\omega \models \mathbf{GF}b$ True
6. $(\{b\}\emptyset)^\omega \models \mathbf{FG}b$ False
7. $\{a\}\emptyset\{b\}^\omega \models a\mathbf{U}b$ False
8. $\{a\}\{a\}\{b\}^\omega \models a\mathbf{U}b$ True

A Quick Primer on Linear Temporal Logic
Reinforcement Learning with Linear Temporal Logic

Task Specification: Gridworld Example (Revisited)

Gridworld

A robot must, with maximal probability:

1. Collect at least 3 items.
2. Then press a special button (once).
3. *After* pressing the button, avoid a dangerous zone *forever*.

$$(\neg \texttt{Button\_Pushed})\mathbf{U}(\texttt{Items\_Collected} \geq 3)$$
$$\wedge \mathbf{F}(\texttt{Button\_Pushed})$$
$$\wedge \mathbf{G}\Big[\texttt{Button\_Pushed} \rightarrow$$
$$\mathbf{XG}(\neg \texttt{Dangerous\_Zone} \wedge \neg \texttt{Button\_Pushed})\Big]$$

Reinforcement Learning with LTL Objectives

Reinforcement Learning with LTL Objectives

The LTL formula specifies the full task objective that the agent should satisfy.

$$\max_{\pi} \Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi)$$

for a given LTL formula φ

- **Important:** memoryless policies are not sufficient here in general!
- **Solution:** the Büchi automata (or similar) for φ can be synchronized with the MDP to provide the minimal amount of history dependency for probability optimal policies.

LTL Objectives: Running Example (Revisited)

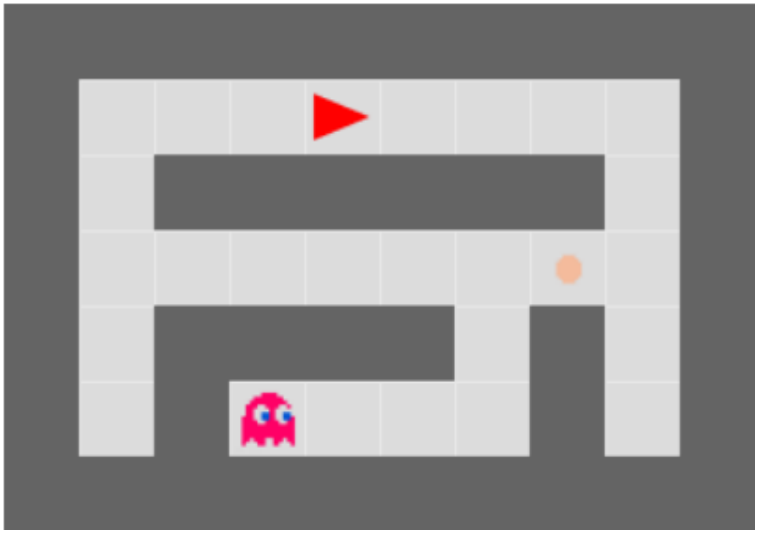


Figure: `MiniPacman` game.

LTL Task Specification

$$\varphi = \mathbf{G}(\neg \texttt{ghost}) \wedge \mathbf{F}(\texttt{food})$$

Labelled Markov Decision Processes

Definition 11 (Labelled MDP)

A **state-labelled Markov Decision Process** is an MDP $\mathcal{M}$ equipped with a labelling function $L : S \mapsto 2^{AP}$

Labelled Markov Decision Processes

Definition 11 (Labelled MDP)

A **state-labelled Markov Decision Process** is an MDP $\mathcal{M}$ equipped with a labelling function $L : S \mapsto 2^{AP}$

Running example: `MiniPacman` game

Labelled Markov Decision Processes

Definition 11 (Labelled MDP)

A **state-labelled Markov Decision Process** is an MDP $\mathcal{M}$ equipped with a labelling function $L : S \mapsto 2^{AP}$

Running example: `MiniPacman` game

- $AP = \{\texttt{ghost}, \texttt{food}\}$.

Labelled Markov Decision Processes

Definition 11 (Labelled MDP)

A **state-labelled Markov Decision Process** is an MDP $\mathcal{M}$ equipped with a labelling function $L : S \mapsto 2^{AP}$

Running example: `MiniPacman` game

- $AP = \{\texttt{ghost}, \texttt{food}\}$.
- $2^{AP} = \{\emptyset, \{\texttt{ghost}\}, \{\texttt{food}\}, \{\texttt{ghost}, \texttt{food}\}\}$ (powerset).

Labelled Markov Decision Processes

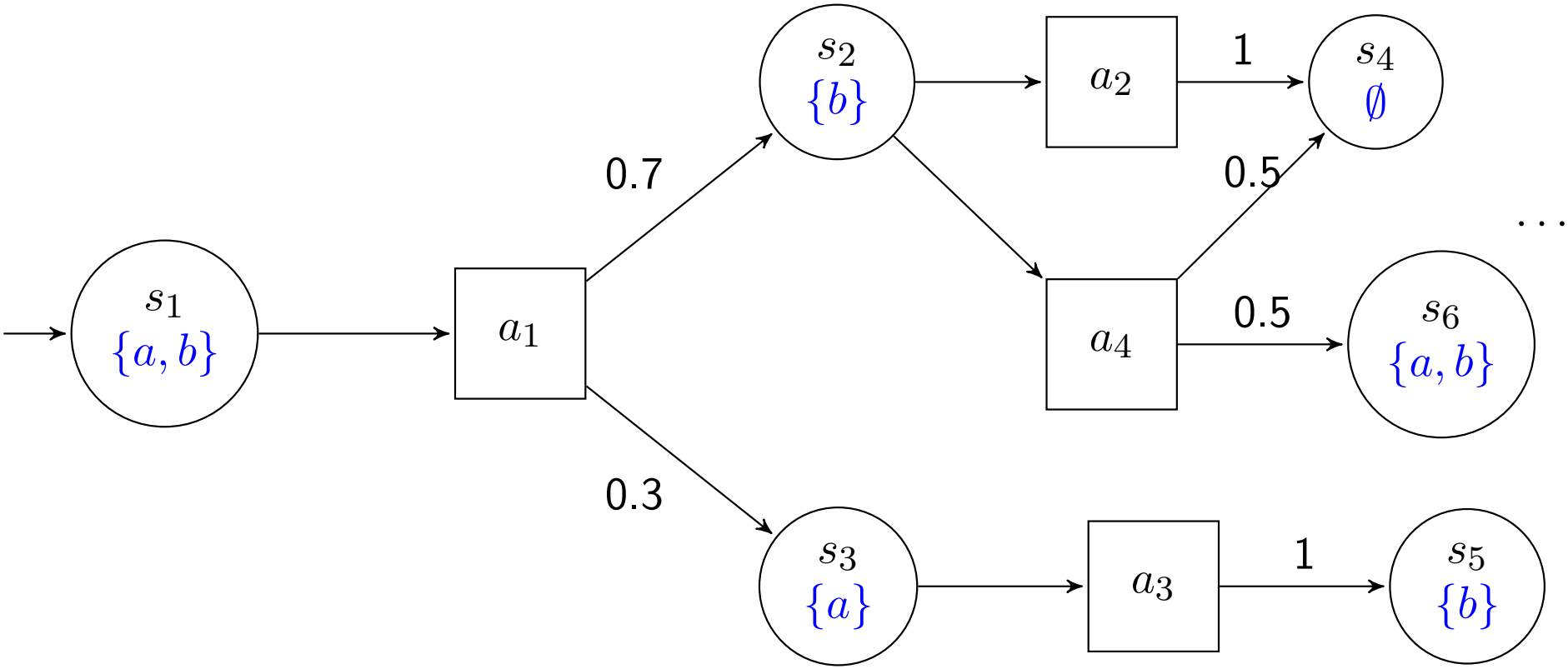
Definition 11 (Labelled MDP)

A **state-labelled Markov Decision Process** is an MDP $\mathcal{M}$ equipped with a labelling function $L : S \mapsto 2^{AP}$

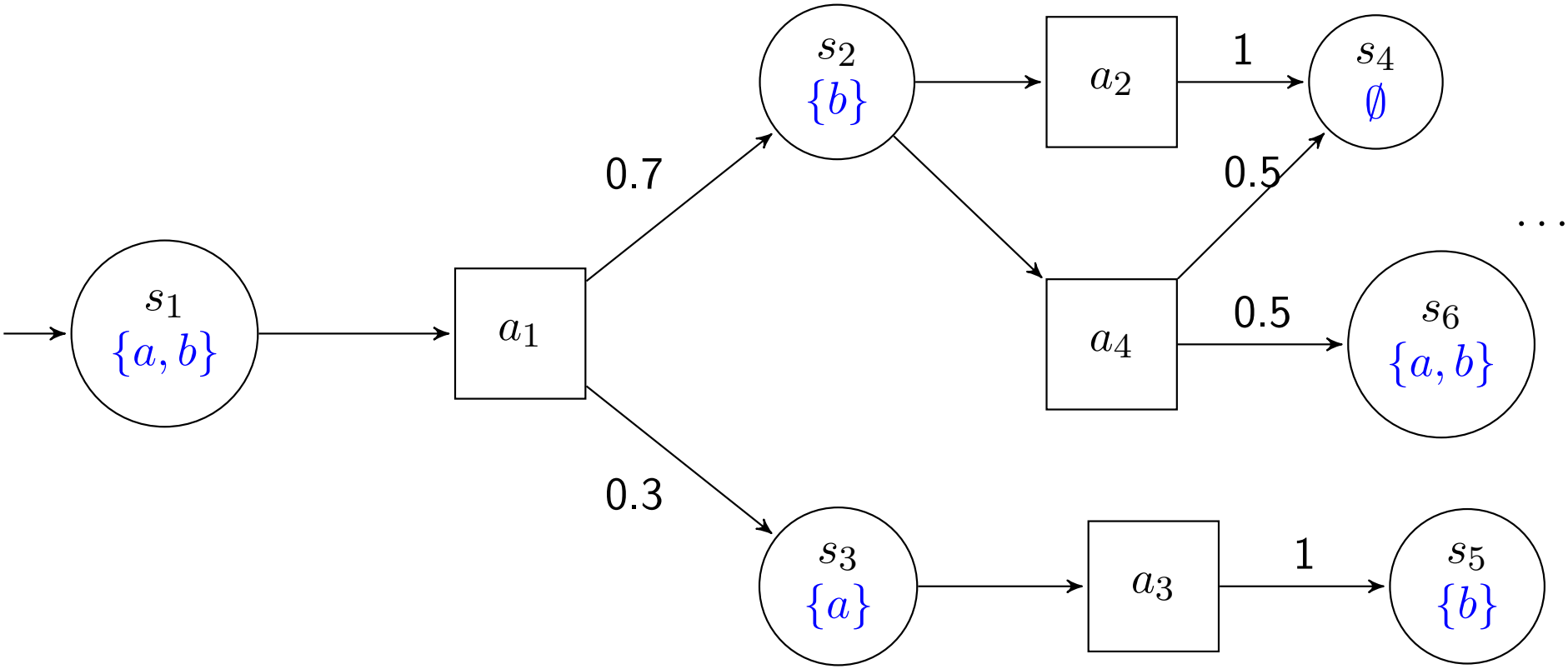
Running example: `MiniPacman` game

- $AP = \{\texttt{ghost}, \texttt{food}\}$.
- $2^{AP} = \{\emptyset, \{\texttt{ghost}\}, \{\texttt{food}\}, \{\texttt{ghost}, \texttt{food}\}\}$ (powerset).
- $\texttt{ghost} \in L(s)$ if "ghost and pacman have the same position".
- $\texttt{food} \in L(s)$ if "pacman picks up the food".

Labelled MDP: Example

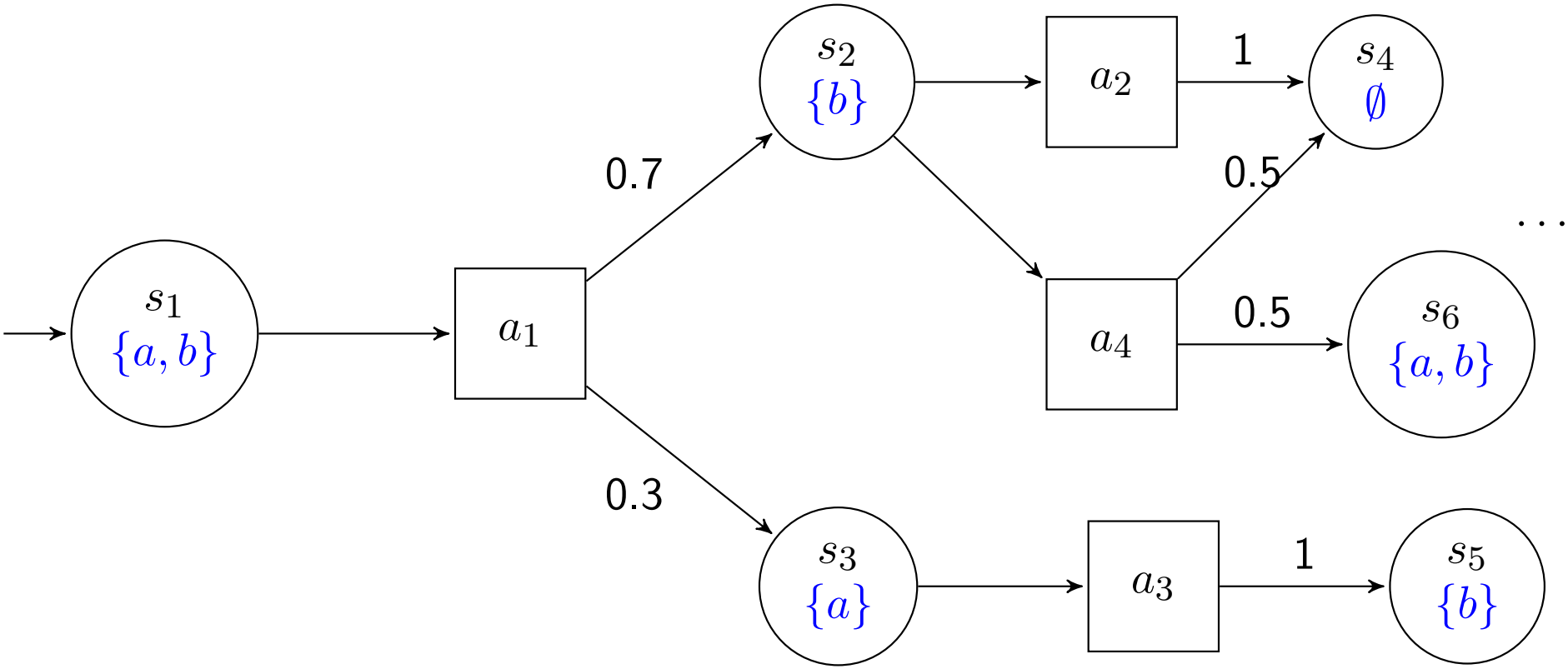


Labelled MDP: Example

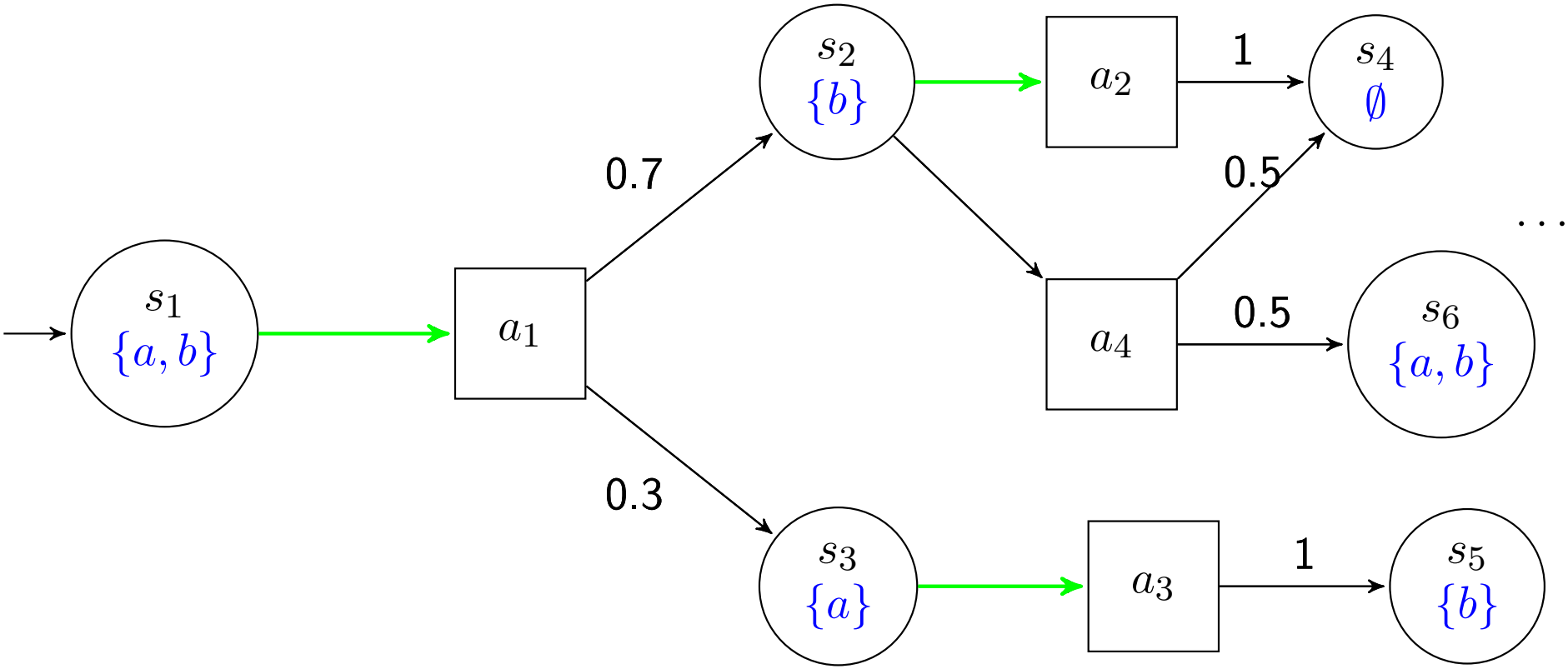


What is the maximum probability that a memoryless policy can satisfy $\varphi = \mathbf{G}(a \rightarrow \mathbf{X}(b))$.

Probability Optimal Policy



Probability Optimal Policy



The probability optimal policy π^* for $\varphi = \mathbf{G}(a \rightarrow \mathbf{X}(b))$, with $\Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi) = 0.7$.

Demo: Running Example

MASA-Safe-RL/tutorial/01_minipacman_labelled_mdp_demo.ipynb

RL with LTL: Quick Overview of Problems

RL with LTL: Different Kinds of Problems

- **LTL objectives**: full LTL can express liveness, fairness, recurrence, and persistence properties. This offers practitioners a way to specify temporally rich objectives that they want their agents to satisfy.

RL with LTL: Quick Overview of Problems

RL with LTL: Different Kinds of Problems

- **LTL objectives**: full LTL can express liveness, fairness, recurrence, and persistence properties. This offers practitioners a way to specify temporally rich objectives that they want their agents to satisfy.
- **LTL constraints**: LTL can also be used to specify constraints, the problem becomes: how do I maximize some auxiliary reward signal among the set of admissible policies satisfying my LTL constraint with high or maximal probability.

RL with LTL: Quick Overview of Problems

RL with LTL: Different Kinds of Problems

- **LTL objectives**: full LTL can express liveness, fairness, recurrence, and persistence properties. This offers practitioners a way to specify temporally rich objectives that they want their agents to satisfy.
- **LTL constraints**: LTL can also be used to specify constraints, the problem becomes: how do I maximize some auxiliary reward signal among the set of admissible policies satisfying my LTL constraint with high or maximal probability.
- **LTL safety constraints**: LTL constraints are expressed as safety constraints, a strict fragment of LTL, which admits deterministic monitors and clean shielding semantics.

Why are we not centring around (discounted) LTL Objectives?

Issues with LTL Objectives

Semantic mismatch: LTL satisfaction is a property of an infinite trace, whereas standard RL optimizes a discounted scalar return. RL for discounted LTL objectives is often not sound and semantically shifts the problem away from maximizing probability of LTL satisfaction.

Why are we not centring around (discounted) LTL Objectives?

Issues with LTL Objectives

Semantic mismatch: LTL satisfaction is a property of an infinite trace, whereas standard RL optimizes a discounted scalar return. RL for discounted LTL objectives is often not sound and semantically shifts the problem away from maximizing probability of LTL satisfaction.

Automata constructions: laborious Büchi automata (or similar) constructions are technically dense and may shift the focus of the tutorial away from safe learning.

Why are we not centring around (discounted) LTL Objectives?

Issues with LTL Objectives

Semantic mismatch: LTL satisfaction is a property of an infinite trace, whereas standard RL optimizes a discounted scalar return. RL for discounted LTL objectives is often not sound and semantically shifts the problem away from maximizing probability of LTL satisfaction.

Automata constructions: laborious Büchi automata (or similar) constructions are technically dense and may shift the focus of the tutorial away from safe learning.

Reward shaping becomes fragile: reward shaping over automata constructions induced by LTL objectives can become fragile and incentivize partial completion or looping rather than full satisfaction.

Why are we not centring around (discounted) LTL Objectives?

Issues with LTL Objectives

Semantic mismatch: LTL satisfaction is a property of an infinite trace, whereas standard RL optimizes a discounted scalar return. RL for discounted LTL objectives is often not sound and semantically shifts the problem away from maximizing probability of LTL satisfaction.

Automata constructions: laborious Büchi automata (or similar) constructions are technically dense and may shift the focus of the tutorial away from safe learning.

Reward shaping becomes fragile: reward shaping over automata constructions induced by LTL objectives can become fragile and incentivize partial completion or looping rather than full satisfaction.

No reward / safety separation: without explicit reward and safety separation training can be inefficient and policy updates that improve reward may affect safety (and vice versa).

Tutorial Focus

Tutorial focus

We use LTL primarily to express **safety constraints**, then study how these constraints can be monitored and enforced via shielding while the agent optimizes an auxiliary reward.

Safety LTL and Product MDP

Safety v.s Liveness

Intuition

- **Safety:** “nothing bad ever happens”.
- **Liveness:** “something good eventually happens”.

Safety v.s Liveness

Intuition

- **Safety:** "nothing bad ever happens".
- **Liveness:** "something good eventually happens".

Definition 12 (Bad prefix)

Let $\Sigma = 2^{AP}$ and let $P \subseteq \Sigma^\omega$ be a property. Denote $w' \preceq w$ to mean "w' is a prefix of word w". Thus, a finite word $w_{\text{pref}} \in \Sigma^*$ is a **bad prefix** for P if

$$\forall w \in \Sigma^\omega : \ (w_{\text{pref}} \preceq w \Rightarrow w \notin P).$$

Safety v.s Liveness

Intuition

- **Safety:** "nothing bad ever happens".
- **Liveness:** "something good eventually happens".

Definition 12 (Bad prefix)

Let $\Sigma = 2^{AP}$ and let $P \subseteq \Sigma^\omega$ be a property. Denote $w' \preceq w$ to mean "w' is a prefix of word w". Thus, a finite word $w_{\text{pref}} \in \Sigma^*$ is a **bad prefix** for P if

$$\forall w \in \Sigma^\omega : \ (w_{\text{pref}} \preceq w \Rightarrow w \notin P).$$

Definition 13 (Safety property)

A property $P \subseteq \Sigma^\omega$ is a **safety property** iff for every $w \notin P$ there exists a finite prefix $w_{\text{pref}} \preceq w$ such that w_{pref} is a bad prefix for P.

Running Example (Revisited Again...)

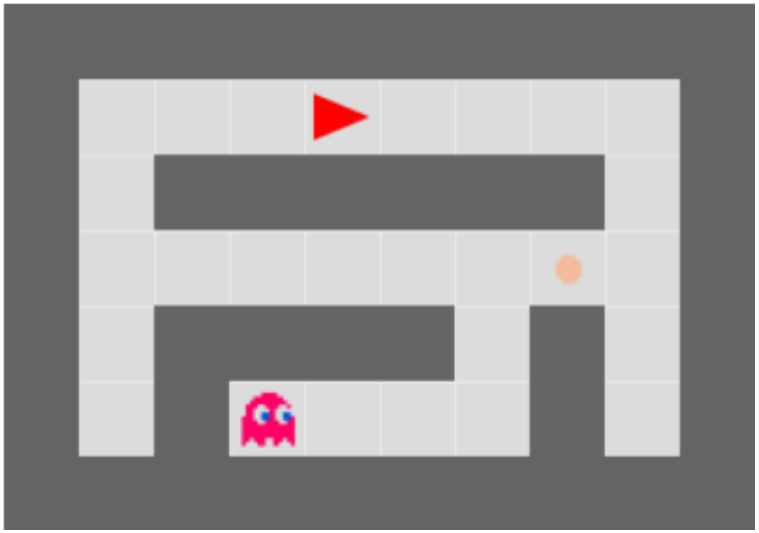


Figure: `MiniPacman` game.

LTL Task Specification

$$\varphi = \mathbf{G}(\neg \texttt{ghost}) \wedge \mathbf{F}(\texttt{food})$$

- **Safety property:** $\mathbf{G}(\neg \texttt{ghost})$.
- **Liveness property:** $\mathbf{F}(\texttt{food})$.

RL with LTL Safety Constraints: Problem Formulation

Problem Statement

Input: reward MDP $\mathcal{M}$ (with labelling function L), LTL safety specification (e.g., $\mathbf{G}\varphi$), tolerance $\epsilon \in [0, 1]$.

Output: Find a policy π^* such that,

- π^* maximizes the usual RL objectives among policies satisfying the safety specification:

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t} \gamma^t R(s_t, a_t) \right] \quad \text{subject to} \quad \Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi) \geq 1 - \epsilon$$

RL with LTL Safety Constraints: Problem Formulation

Problem Statement

Input: reward MDP $\mathcal{M}$ (with labelling function L), LTL safety specification (e.g., $\mathbf{G}\varphi$), tolerance $\epsilon \in [0, 1]$.

Output: Find a policy π^* such that,

- π^* maximizes the usual RL objectives among policies satisfying the safety specification:

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t} \gamma^t R(s_t, a_t)\right] \quad \text{subject to} \quad \Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi) \geq 1 - \epsilon$$

Special cases

- $\epsilon = 0$: hard / almost-sure safety.
- $\epsilon > 0$: probabilistic safety with a bounded risk of violation.

The Safety Fragment of LTL: Recap

Definition 14 (Safety LTL)

The safety fragment of LTL can be generated by the following grammar:

$$\varphi ::= \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

where $p \in AP$.

The Safety Fragment of LTL: Recap

Definition 14 (Safety LTL)

The safety fragment of LTL can be generated by the following grammar:

$$\varphi ::= \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

where $p \in AP$.

The usual grammar for LTL is:

$$\varphi ::= \top \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi$$

The Safety Fragment of LTL: Recap

Definition 14 (Safety LTL)

The safety fragment of LTL can be generated by the following grammar:

$$\varphi ::= \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

where $p \in AP$.

The usual grammar for LTL is:

$$\varphi ::= \top \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi$$

Why can't we negate general formula φ in the grammar for safety LTL?

The Safety Fragment of LTL: Recap

Definition 14 (Safety LTL)

The safety fragment of LTL can be generated by the following grammar:

$$\varphi ::= \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

where $p \in AP$.

The usual grammar for LTL is:

$$\varphi ::= \top \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi \mathbf{U} \varphi$$

Why can't we negate general formula φ in the grammar for safety LTL?

Because of the common identity: $\mathbf{F}\varphi = \neg\mathbf{G}\neg\varphi$. We are not allowed $\mathbf{F}$, because this generates live formula!

The Safety Fragment of LTL: Recap

Definition 14 (Safety LTL)

The safety fragment of LTL can be generated by the following grammar:

$$\varphi ::= \top \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{G}\varphi$$

where $p \in AP$.

The usual grammar for LTL is:

$$\varphi ::= \top \mid p \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi$$

Why can't we negate general formula φ in the grammar for safety LTL?

Because of the common identity: $\mathbf{F}\varphi = \neg\mathbf{G}\neg\varphi$. We are not allowed $\mathbf{F}$, because this generates live formula!

Similarly we are not allowed $\mathbf{U}$ because of the identity: $\mathbf{F}\varphi = \top\mathbf{U}\varphi$.

The Safety Fragment of LTL: Key Properties

Theorem 15 (Monitorability (informal))

*For every **safety** LTL formula φ there exists a **deterministic finite automaton (DFA)** that recognizes the set of **bad prefixes** of φ (equivalently: a DFA monitor that detects violations).*

The Safety Fragment of LTL: Key Properties

Theorem 15 (Monitorability (informal))

*For every **safety** LTL formula φ there exists a **deterministic finite automaton (DFA)** that recognizes the set of **bad prefixes** of φ (equivalently: a DFA monitor that detects violations).*

Corollary 16

*For safety constraints we do **not** need Büchi acceptance:*

- *We can work with **DFAs** (finite memory).*
- *Synchronizing an MDP with a DFA yields a product MDP where safety reduces to **reachability of a rejecting sink**.*

Deterministic Finite Automata

Definition 17 (Deterministic Finite Automaton (DFA))

A **DFA** is a tuple $\mathcal{D} = (\Sigma, Q, \delta, q_0, F)$ where:

- Σ is a finite alphabet (here: $\Sigma = 2^{AP}$),
- Q is a finite set of states,
- $\delta : Q \times \Sigma \to Q$ is a **total** transition function,
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of accepting states.

DFA: Language

Definition 18 (Extended transition function)

The transition function extends to words $\delta^* : Q \times \Sigma^* \to Q$ by the following recursive definition:

$$\delta^*(q, w) = \delta(\delta^*(q, w \setminus w\downarrow), w\downarrow).$$

DFA: Language

Definition 18 (Extended transition function)

The transition function extends to words $\delta^* : Q \times \Sigma^* \to Q$ by the following recursive definition:

$$\delta^*(q, w) = \delta(\delta^*(q, w \setminus w\downarrow), w\downarrow).$$

Definition 19 (Language of a DFA)

Let $\downarrow$ denote the last element of a finite word $w \in \Sigma^*$. The language recognized by $\mathcal{D}$ is:

$$L(\mathcal{D}) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}.$$

DFA: Language

Definition 18 (Extended transition function)

The transition function extends to words $\delta^* : Q \times \Sigma^* \to Q$ by the following recursive definition:

$$\delta^*(q, w) = \delta(\delta^*(q, w \setminus w\downarrow), w\downarrow).$$

Definition 19 (Language of a DFA)

Let $\downarrow$ denote the last element of a finite word $w \in \Sigma^*$. The language recognized by $\mathcal{D}$ is:

$$L(\mathcal{D}) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}.$$

Remark (DFA monitors for safety)

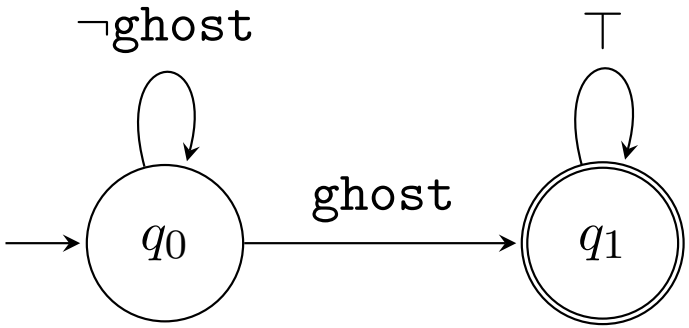
In our setting, we build a DFA for monitoring:

- **bad prefixes** of a safety LTL formula (accepting = "violation detected").

DFA: Examples

Example 1: Invariant $\mathbf{G}(\neg \texttt{ghost})$

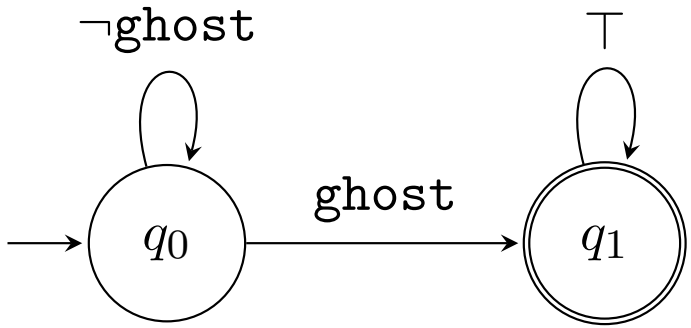
Once ghost is observed, safety is violated forever.



DFA: Examples

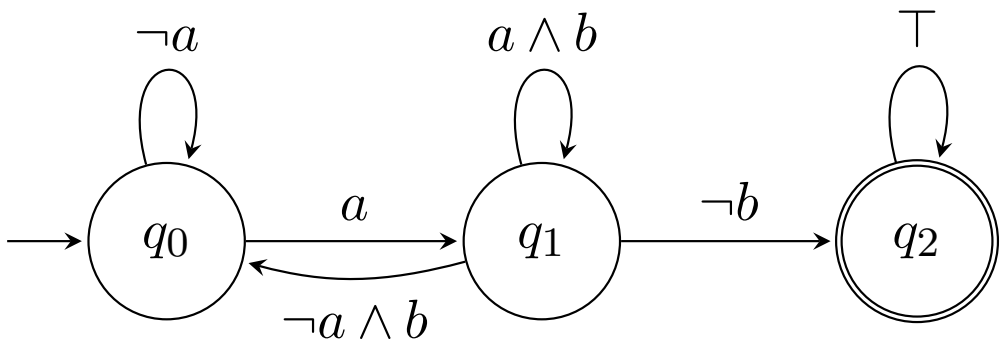
Example 1: Invariant $\mathbf{G}(\neg \texttt{ghost})$

Once ghost is observed, safety is violated forever.



Example 2: $\mathbf{G}(a \rightarrow \mathbf{X}b)$

Interpretation: whenever a holds now, then b must hold next.



Safety LTL and Product MDP
Product MDP Construction

Product MDP

Reminder: Labelled MDP

A labelled MDP is an MDP $\mathcal{M} = (S, s_0, A, P)$ equipped with $L : S \to 2^{AP}$. The induced label sequence of a path $\rho = s_0 a_0 s_1 a_1 \ldots$ is $L(\rho) = L(s_0)L(s_1)\ldots \in (2^{AP})^\omega$.

Product MDP

Reminder: Labelled MDP

A labelled MDP is an MDP $\mathcal{M} = (S, s_0, A, P)$ equipped with $L : S \to 2^{AP}$. The induced label sequence of a path $\rho = s_0 a_0 s_1 a_1 \ldots$ is $L(\rho) = L(s_0)L(s_1)\ldots \in (2^{AP})^\omega$.

Idea

To check a safety LTL constraint φ, we run a DFA monitor $\mathcal{D}$ on the observed labels. Synchronize the MDP with the DFA to obtain a new MDP over **product states**.

Product MDP: Formal Definition

Definition 20 (Product MDP (MDP $\otimes$ DFA))

Let $\mathcal{M} = (S, s_0, A, P, L)$ be a labelled MDP and let $\mathcal{D} = (\Sigma, Q, \delta, q_0, F)$ be a DFA with $\Sigma = 2^{AP}$. The **product MDP** is

$$\mathcal{M} \otimes \mathcal{D} = (S^{\otimes}, s_0^{\otimes}, A^{\otimes}, P^{\otimes}),$$

where:

- $S^{\otimes} = S \times Q$,
- $s_0^{\otimes} = (s_0, \delta(q_0, L(s_0)))$,
- $A^{\otimes}((s, q)) = A(s)$.

Product MDP: Formal Definition

Definition 20 (Product MDP (MDP $\otimes$ DFA))

Let $\mathcal{M} = (S, s_0, A, P, L)$ be a labelled MDP and let $\mathcal{D} = (\Sigma, Q, \delta, q_0, F)$ be a DFA with $\Sigma = 2^{AP}$. The **product MDP** is

$$\mathcal{M} \otimes \mathcal{D} = (S^{\otimes}, s_0^{\otimes}, A^{\otimes}, P^{\otimes}),$$

where:

- $S^{\otimes} = S \times Q$,
- $s_0^{\otimes} = (s_0, \delta(q_0, L(s_0)))$,
- $A^{\otimes}((s, q)) = A(s)$.

Remark

The product state (s, q) compactly represents:

- The current environment state s.
- The current monitor state q (i.e., the relevant safety memory).

Product MDP: Transition Function

Product Transition function: $P^{\otimes}$

Let $(s, q) \in S^{\otimes}$ and $a \in A^{\otimes}((s, q)) = A(s)$. For every $(s', q') \in S^{\otimes}$ define:

$$P^{\otimes}\Big((s', q') \mid (s, q), a\Big) = \begin{cases} P(s' \mid s, a) & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise.} \end{cases}$$

Product MDP: Transition Function

Product Transition function: $P^{\otimes}$

Let $(s, q) \in S^{\otimes}$ and $a \in A^{\otimes}((s, q)) = A(s)$. For every $(s', q') \in S^{\otimes}$ define:

$$P^{\otimes}\Big((s', q') \mid (s, q), a\Big) = \begin{cases} P(s' \mid s, a) & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise.} \end{cases}$$

Interpretation

- The environment transitions as usual with probability $P(s'|s, a)$.
- The DFA updates using the label of the **new** state:

$$q' = \delta(q, L(s')).$$

Product MDP: Transition Function

Product Transition function: $P^{\otimes}$

Let $(s, q) \in S^{\otimes}$ and $a \in A^{\otimes}((s, q)) = A(s)$. For every $(s', q') \in S^{\otimes}$ define:

$$P^{\otimes}\Big((s', q') \mid (s, q), a\Big) = \begin{cases} P(s' \mid s, a) & \text{if } q' = \delta(q, L(s')) \\ 0 & \text{otherwise.} \end{cases}$$

Interpretation

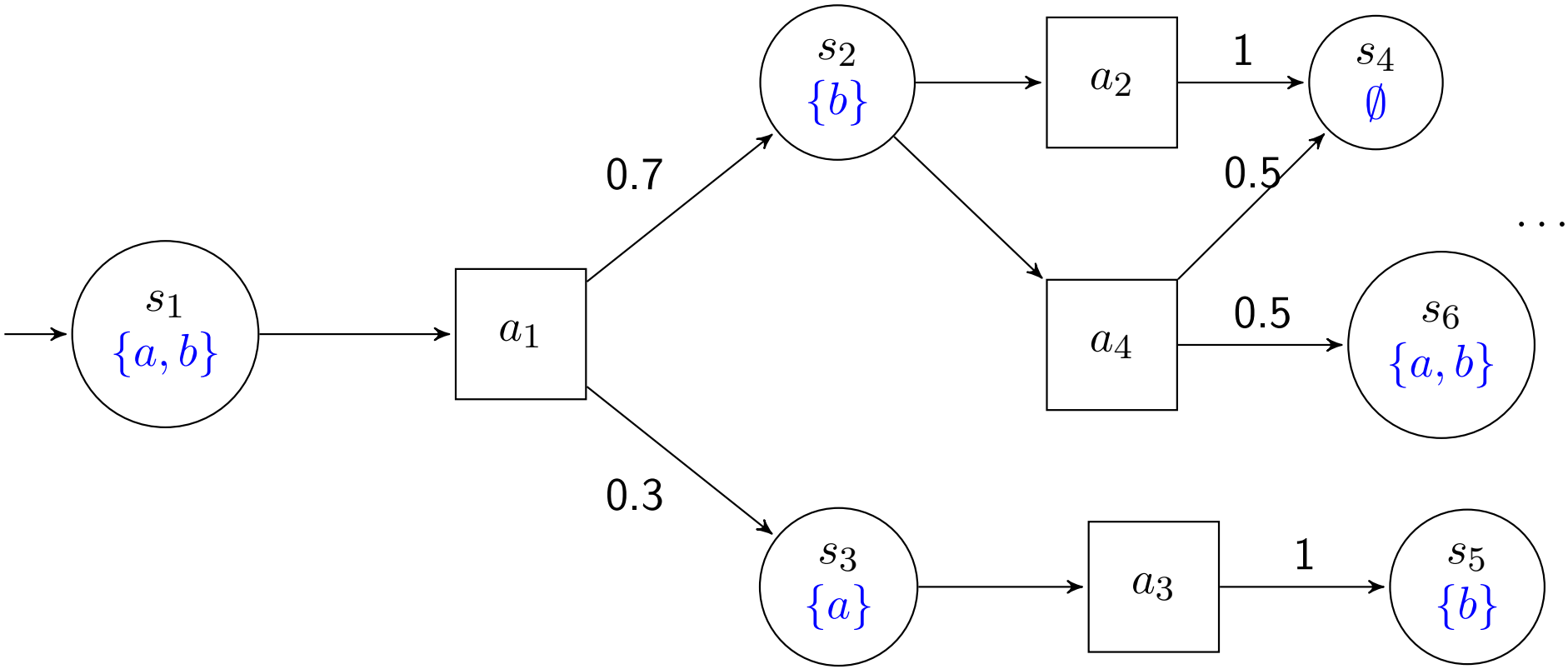
- The environment transitions as usual with probability $P(s'|s, a)$.
- The DFA updates using the label of the **new** state:

$$q' = \delta(q, L(s')).$$

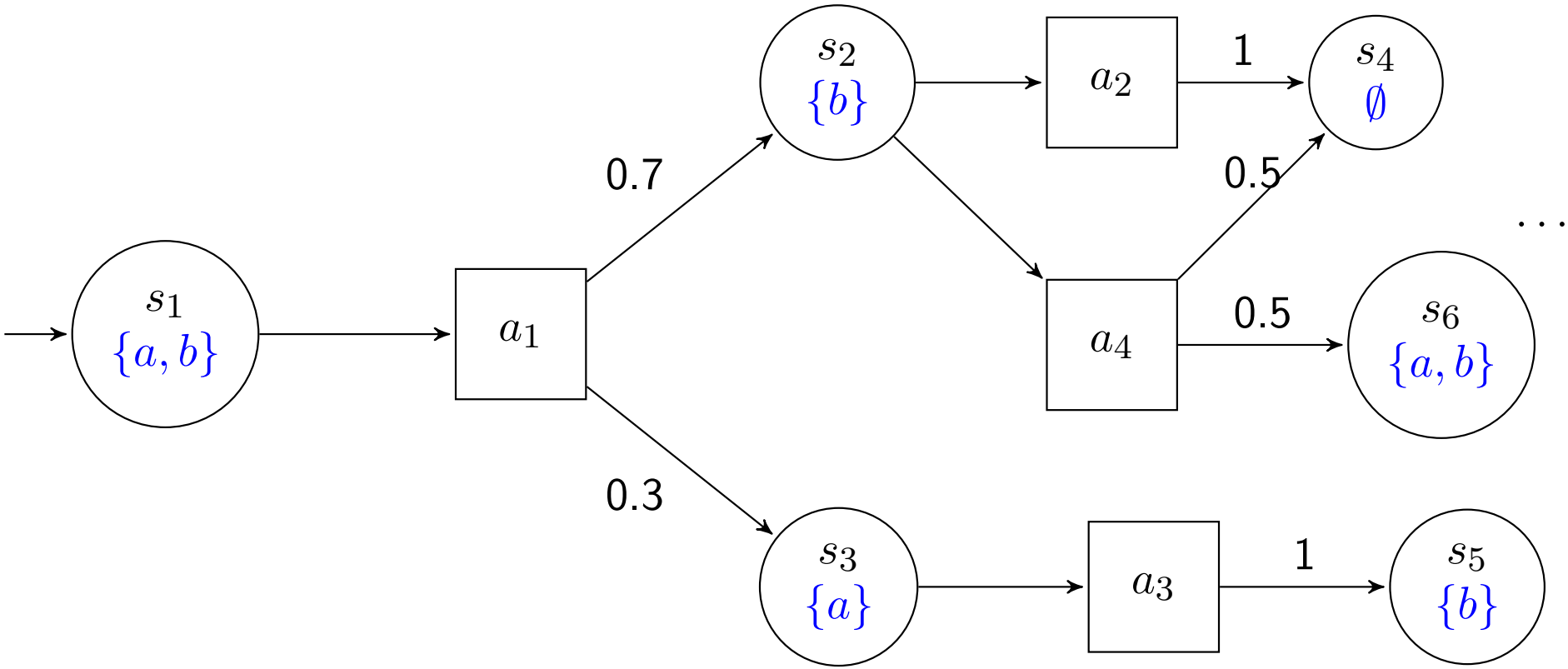
Rejecting sink

For safety monitoring the convention is that accepting DFA states are considered rejecting sinks. Once the DFA enters F it never leaves.

Product MDP: Example (Labelled MDP)



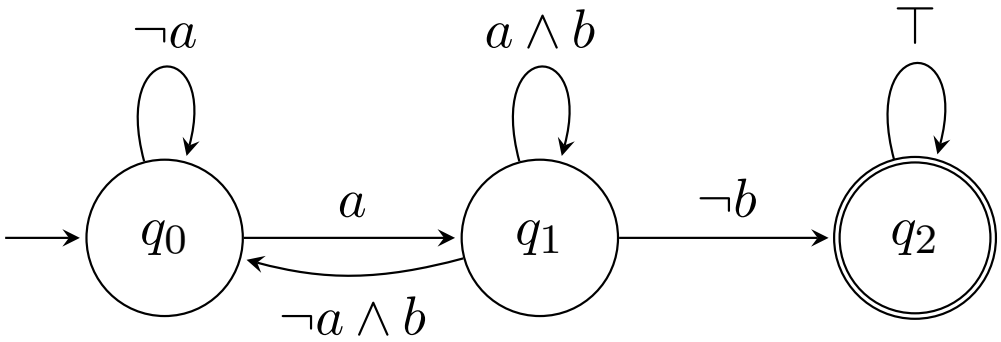
Product MDP: Example (Labelled MDP)



We're going to construct the product MDP with the safety property $\mathbf{G}(a \rightarrow \mathbf{X}(b))$.

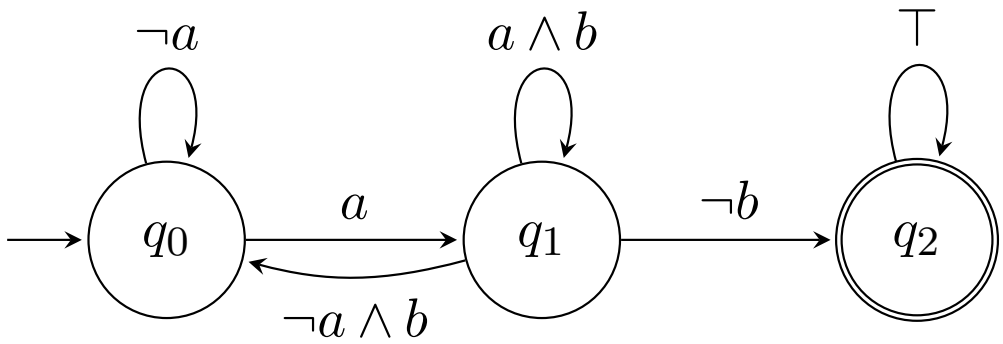
Product MDP: Example (DFA)

Recall the DFA for $\mathbf{G}(a \rightarrow \mathbf{X}(b))$:



Product MDP: Example (DFA)

Recall the DFA for $\mathbf{G}(a \to \mathbf{X}(b))$:



Labelled MDP

Let's only consider the MDP fragment on states: $\{s_1, s_2, s_3\}$:

$$s_1 \xrightarrow{a_1} \begin{cases} s_2 & \text{w.p. } 0.7 \\ s_3 & \text{w.p. } 0.3 \end{cases}$$

(ignoring all outgoing transitions of s_2, s_3).

Labelling:

$$L(s_1) = \{a, b\}, \quad L(s_2) = \{b\}, \quad L(s_3) = \{a\}.$$

Product MDP: Example (Product States)

Product state space (restricted)

The restricted product state space is:

$$S^{\otimes} = \{(s_i, q_j) \mid i \in \{1, 2, 3\},\ j \in \{0, 1, 2\}\}.$$

Initial product state:

$$s_0^{\otimes} = (s_1, \delta(q_0, L(s_1))).$$

Recall that $L(s_1) = \{a, b\}$ and $q_1 = \delta(q_0, \{a, b\})$. Thus the effective initial product state:

$$s_0^{\otimes} = (s_1, q_1).$$

Product MDP: Example (DFA Successors)

Computing DFA successors and MDP transitions

Using $L(s)$:

- $L(s_2) = \{b\} \models \neg a \wedge b$, so,

$$\delta(q_1, L(s_2)) = q_0,$$

Product MDP: Example (DFA Successors)

Computing DFA successors and MDP transitions

Using $L(s)$:

- $L(s_2) = \{b\} \models \neg a \wedge b$, so,

$$\delta(q_1, L(s_2)) = q_0,$$

- $L(s_3) = \{a\} \models a \wedge \neg b$, so,

$$\delta(q_1, L(s_3)) = q_2,$$

Thus, we obtain the following transitions:

Product MDP: Example (DFA Successors)

Computing DFA successors and MDP transitions

Using $L(s)$:

- $L(s_2) = \{b\} \models \neg a \wedge b$, so,

$$\delta(q_1, L(s_2)) = q_0,$$

- $L(s_3) = \{a\} \models a \wedge \neg b$, so,

$$\delta(q_1, L(s_3)) = q_2,$$

Thus, we obtain the following transitions:

- From (s_1, q_1) under a_1:

$$(s_1, q_1) \xrightarrow{a_1} \begin{cases} (s_2, q_0) & \text{w.p. } 0.7 \quad (\delta(q_1, L(s_2)) = q_0) \\ (s_3, q_2) & \text{w.p. } 0.3 \quad (\delta(q_1, L(s_3)) = q_2) \end{cases}$$

Product MDP: Example (DFA Successors)

Computing DFA successors and MDP transitions

Using $L(s)$:

- $L(s_2) = \{b\} \models \neg a \wedge b$, so,

$$\delta(q_1, L(s_2)) = q_0,$$

- $L(s_3) = \{a\} \models a \wedge \neg b$, so,

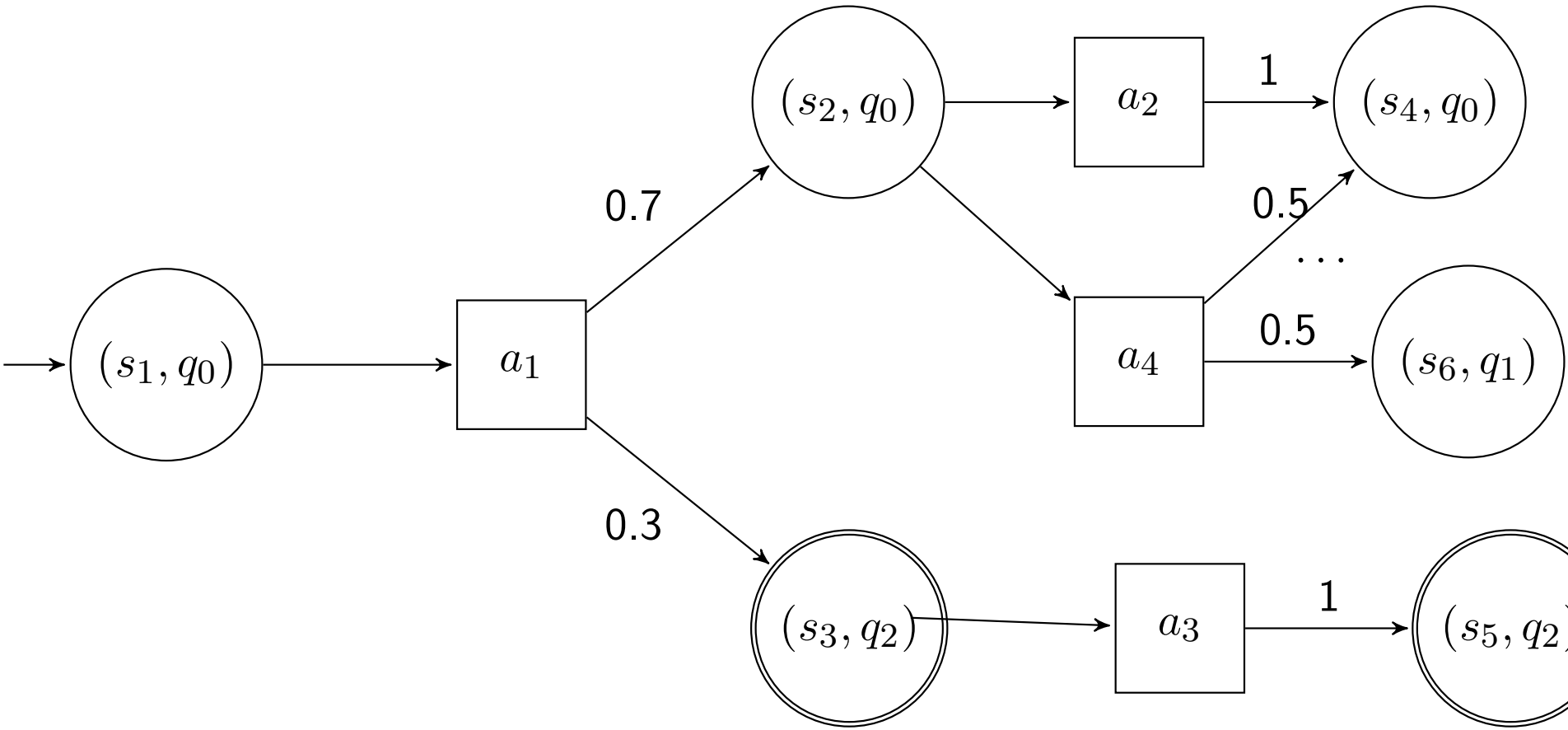
$$\delta(q_1, L(s_3)) = q_2,$$

Thus, we obtain the following transitions:

- From (s_1, q_1) under a_1:

$$(s_1, q_1) \xrightarrow{a_1} \begin{cases} (s_2, q_0) & \text{w.p. } 0.7 \quad (\delta(q_1, L(s_2)) = q_0) \\ (s_3, q_2) & \text{w.p. } 0.3 \quad (\delta(q_1, L(s_3)) = q_2) \end{cases}$$

Product MDP: Example



Product MDP: Key Theorem (Setup)

Setup

Let φ be a safety LTL formula, and let $\mathcal{D}$ be a DFA monitor with a rejecting sink states F such that:

- F is reached **iff** a bad prefix of φ has occurred.

Define the set of **bad product states**,

$$F^{\otimes} := S \times F \subseteq S^{\otimes}.$$

and **product labelling function** $L^{\otimes}$ be such that,

$$\texttt{accept} \in L^{\otimes}(s^{\otimes}) \text{ iff } s^{\otimes} \in F^{\otimes}$$

Product MDP: Key Theorem (Setup)

Setup

Let φ be a safety LTL formula, and let $\mathcal{D}$ be a DFA monitor with a rejecting sink states F such that:

- F is reached **iff** a bad prefix of φ has occurred.

Define the set of **bad product states**,

$$F^{\otimes} := S \times F \subseteq S^{\otimes}.$$

and **product labelling function** $L^{\otimes}$ be such that,

$$\texttt{accept} \in L^{\otimes}(s^{\otimes}) \text{ iff } s^{\otimes} \in F^{\otimes}$$

Corollary 21 (Unique Path)

For every finite or infinite path $\rho = s_0 a_0 s_1 a_1 \ldots$ in the MDP $\mathcal{M}$ there exists a unique path $\rho^{\times} = (s_0, q_0) a_0 (s_1, q_1) a_1 \ldots$ in the product MDP $\mathcal{M} \otimes \mathcal{D}$.

Product MDP: Key Theorem

Theorem 22 (Safety Probability as Reachability)

For every policy π in $\mathcal{M}$ (and its corresponding policy $\pi^{\otimes}$ in $\mathcal{M} \otimes \mathcal{D}$),

$$\Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi) \;=\; 1 \;-\; \Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}\Big(\rho^{\times} \models \mathbf{F}\,\mathit{accept}\Big)$$

Product MDP: Key Theorem

Theorem 22 (Safety Probability as Reachability)

For every policy π in $\mathcal{M}$ (and its corresponding policy $\pi^{\otimes}$ in $\mathcal{M} \otimes \mathcal{D}$),

$$\Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi) \;=\; 1 \;-\; \Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}\Big(\rho^{\times} \models \mathbf{F}\,\mathit{accept}\Big)$$

Takeaway: assessing the safety probability of a policy π reduces to a reachability analysis in the product MDP.

Product MDP: Memoryless Policies

Recall (memoryless policies)

A policy π is memoryless if for any finite path $\rho = s_0 a_0 \dots s_n$, $\pi(\rho)$ only depends on s_n, thus $\pi(\cdot \mid s)$ is a probability measure over actions A dependent on the current state s only.

Product MDP: Memoryless Policies

Recall (memoryless policies)

A policy π is memoryless if for any finite path $\rho = s_0 a_0 \dots s_n$, $\pi(\rho)$ only depends on s_n, thus $\pi(\cdot \mid s)$ is a probability measure over actions A dependent on the current state s only.

Corollary 23 (Memoryless optimality in the product)

*Let $\mathcal{M}$ be a finite labelled MDP and let $\mathcal{D}$ be a DFA monitor for a safety LTL formula φ. Then there exists a **memoryless** policy $\pi^{\otimes *}$ on the product MDP $\mathcal{M} \otimes \mathcal{D}$ that is **probability optimal** for safety.*

Product MDP: Memoryless Policies

Recall (memoryless policies)

A policy π is memoryless if for any finite path $\rho = s_0 a_0 \dots s_n$, $\pi(\rho)$ only depends on s_n, thus $\pi(\cdot \mid s)$ is a probability measure over actions A dependent on the current state s only.

Corollary 23 (Memoryless optimality in the product)

*Let $\mathcal{M}$ be a finite labelled MDP and let $\mathcal{D}$ be a DFA monitor for a safety LTL formula φ. Then there exists a **memoryless** policy $\pi^{\otimes *}$ on the product MDP $\mathcal{M} \otimes \mathcal{D}$ that is **probability optimal** for safety.*

Why this is important?

In the original MDP $\mathcal{M}$, memoryless policies may be insufficient but the DFA state q stores *exactly the minimal history information* needed to detect (or avoid) a bad prefix.

Demo: MASA-Safe-RL

MASA-Safe-RL/tutorial/02_colour_bomb_product_mdp_demo.ipynb

Safety LTL and Product MDP
Reachability and Safe Sets

Primer on Markov Chains

From MDP to Markov chain

Fix a (memoryless) policy π in an MDP $\mathcal{M} = (S, s_0, A, P)$. Then π induces a **Markov chain** $\mathcal{M}^\pi$ on S with transition probabilities:

$$P^\pi(s' \mid s) := \sum_{a \in A(s)} \pi(a \mid s)\, P(s' \mid s, a).$$

Primer on Markov Chains

From MDP to Markov chain

Fix a (memoryless) policy π in an MDP $\mathcal{M} = (S, s_0, A, P)$. Then π induces a **Markov chain** $\mathcal{M}^\pi$ on S with transition probabilities:

$$P^\pi(s' \mid s) := \sum_{a \in A(s)} \pi(a \mid s)\, P(s' \mid s, a).$$

Definition 24 (Transient state)

A state s is **transient** in $\mathcal{M}^\pi$ if, starting from s, the probability of returning to s infinitely often is 0. Equivalently, the expected number of visits to s is finite.

Primer on Markov Chains

From MDP to Markov chain

Fix a (memoryless) policy π in an MDP $\mathcal{M} = (S, s_0, A, P)$. Then π induces a **Markov chain** $\mathcal{M}^\pi$ on S with transition probabilities:

$$P^\pi(s' \mid s) := \sum_{a \in A(s)} \pi(a \mid s)\, P(s' \mid s, a).$$

Definition 24 (Transient state)

A state s is **transient** in $\mathcal{M}^\pi$ if, starting from s, the probability of returning to s infinitely often is 0. Equivalently, the expected number of visits to s is finite.

Definition 25 (Recurrent state)

A state s is **recurrent** in $\mathcal{M}^\pi$ if, starting from s, the probability of returning to s infinitely often is 1.

Primer on Markov Chains: Recurrent Classes

Definition 26 (Recurrent class)

A **recurrent class** (also called a **bottom strongly connected component**) is a non-empty set $C \subseteq S$ such that:

- the subgraph induced by C is strongly connected, and
- there are **no transitions leaving** C under P^π.

Primer on Markov Chains: Recurrent Classes

Definition 26 (Recurrent class)

A **recurrent class** (also called a **bottom strongly connected component**) is a non-empty set $C \subseteq S$ such that:

- the subgraph induced by C is strongly connected, and
- there are **no transitions leaving** C under P^π.

Key fact

For any finite Markov chain, with probability 1 a run eventually enters some recurrent class and stays there forever.

Safe End Components

Definition 27 (Safe recurrent class / safe end component)

Fix a memoryless policy $\pi^{\otimes}$. A recurrent class C of the product Markov chain $(\mathcal{M} \otimes \mathcal{D})^{\pi^{\otimes}}$ is called a **safe end component** in the product MDP $(\mathcal{M} \otimes \mathcal{D})$ if

$$C \subseteq S^{\otimes} \setminus F^{\otimes}.$$

In other words, once the chain enters C, it can never reach $F^{\otimes}$.

Safe End Components

Definition 27 (Safe recurrent class / safe end component)

Fix a memoryless policy $\pi^{\otimes}$. A recurrent class C of the product Markov chain $(\mathcal{M} \otimes \mathcal{D})^{\pi^{\otimes}}$ is called a **safe end component** in the product MDP $(\mathcal{M} \otimes \mathcal{D})$ if

$$C \subseteq S^{\otimes} \setminus F^{\otimes}.$$

In other words, once the chain enters C, it can never reach $F^{\otimes}$.

Theorem 28 (Existence of a Safe End Component)

*Assume there exists a **memoryless** policy $\pi^{\otimes}$ such that,*

$$\Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}\left(\rho^{\times} \models \neg \mathbf{F}\, \mathit{accept}\right) > 0.$$

*Then there exists at least one **safe end component**, defined by the corresponding recurrent class C in the induced Markov chain $(\mathcal{M} \otimes \mathcal{D})^{\pi^{\otimes}}$.*

Almost Sure Safe Set

Definition 29 (Almost sure safe set)

The **almost sure safe set** is the set of product states from which there exists a policy that avoids $F^{\otimes}$ with probability 1:

$$\text{AS} := \{s \in S^{\otimes} \mid \exists \pi^{\otimes}\, s.t.\, \Pr^{\pi^{\otimes}}_{\mathcal{M}^{\otimes}\mathcal{D}}(\rho^{\times} \models \neg \mathbf{F}\, \texttt{accept} \mid s) = 1\}.$$

Almost Sure Safe Set

Definition 29 (Almost sure safe set)

The **almost sure safe set** is the set of product states from which there exists a policy that avoids $F^{\otimes}$ with probability 1:

$$\text{AS} := \{s \in S^{\otimes} \mid \exists \pi^{\otimes} \, s.t. \, \Pr^{\pi^{\otimes}}_{\mathcal{M}^{\otimes}\mathcal{D}}(\rho^{\times} \models \neg \mathbf{F}\, \texttt{accept} \mid s) = 1\}.$$

Intuition

- From $s \in \text{AS}$ we can choose actions so that we eventually reach a **safe end component** with probability 1.
- Outside AS, every policy has some non-zero probability of eventually reaching $F^{\otimes}$.

Almost Sure Safe Set

Definition 29 (Almost sure safe set)

The **almost sure safe set** is the set of product states from which there exists a policy that avoids $F^{\otimes}$ with probability 1:

$$\text{AS} := \{s \in S^{\otimes} \mid \exists \pi^{\otimes} \, s.t. \, \Pr^{\pi^{\otimes}}_{\mathcal{M} \otimes \mathcal{D}}(\rho^{\times} \models \neg \mathbf{F} \, \texttt{accept} \mid s) = 1\}.$$

Intuition

- From $s \in \text{AS}$ we can choose actions so that we eventually reach a **safe end component** with probability 1.
- Outside AS, every policy has some non-zero probability of eventually reaching $F^{\otimes}$.

Remark (Winning Set)

The **winning set** W is closely related to AS is often characterized as the set of state and action pairs the keep the product MDP in the **almost sure safe set**.

ϵ–Safe Set

Motivation

Almost-sure safety can be too strong in practice (unknown dynamics, function approximation, partial observability). Instead we may require safety **with high probability**.

ϵ–Safe Set

Motivation

Almost-sure safety can be too strong in practice (unknown dynamics, function approximation, partial observability). Instead we may require safety **with high probability**.

Definition 30 (ϵ–safe set)

For $\epsilon \in (0, 1)$, define the ϵ–**safe set** as

$$\mathsf{S}_\epsilon := \left\{ s \in S^{\otimes} \;\middle|\; \exists \pi^{\otimes}\, s.t.\, \Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}(\rho^{\times} \models \neg \mathbf{F}\, \mathtt{accept} \mid s) \geq 1 - \epsilon \right\}.$$

ϵ–Safe Set

Motivation

Almost-sure safety can be too strong in practice (unknown dynamics, function approximation, partial observability). Instead we may require safety **with high probability**.

Definition 30 (ϵ–safe set)

For $\epsilon \in (0, 1)$, define the ϵ–**safe set** as

$$\mathsf{S}_\epsilon := \left\{ s \in S^{\otimes} \;\middle|\; \exists \pi^{\otimes}\, s.t.\, \Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}(\rho^{\times} \models \neg \mathbf{F}\, \mathtt{accept} \mid s) \geq 1 - \epsilon \right\}.$$

Interpretation

If $s \in \mathsf{S}_\epsilon$, we can steer the system so that reaching the bad sink set occurs with probability at most ϵ. the ϵ-**winning set** W_ϵ is related to S_ϵ in a similar way as before.

Computing Safe Sets: Overview

Goal

Given the product MDP $\mathcal{M} \otimes \mathcal{D}$, compute:

- the **almost sure safe set** AS,
- the ϵ**–safe sets** S_ϵ.

Computing Safe Sets: Overview

Goal

Given the product MDP $\mathcal{M} \otimes \mathcal{D}$, compute:

- the **almost sure safe set** AS,
- the ϵ**–safe sets** S_ϵ.

Almost sure safe set AS (qualitative)

- Remove the bad states $F^\otimes$ from the graph.

Computing Safe Sets: Overview

Goal

Given the product MDP $\mathcal{M} \otimes \mathcal{D}$, compute:

- the **almost sure safe set** AS,
- the ϵ**–safe sets** S_ϵ.

Almost sure safe set AS (qualitative)

- Remove the bad states $F^{\otimes}$ from the graph.
- Iteratively remove states that **cannot stay inside** the remaining set with any action:

$$s \text{ is removed if } \forall a \in A(s),\ \mathsf{supp}(P(\cdot \mid s, a)) \not\subseteq W.$$

Computing Safe Sets: Overview

Goal

Given the product MDP $\mathcal{M} \otimes \mathcal{D}$, compute:

- the **almost sure safe set** AS,
- the ϵ**–safe sets** S_ϵ.

Almost sure safe set AS (qualitative)

- Remove the bad states $F^{\otimes}$ from the graph.
- Iteratively remove states that **cannot stay inside** the remaining set with any action:

$$s \text{ is removed if } \forall a \in A(s),\ \mathsf{supp}(P(\cdot \mid s, a)) \not\subseteq W.$$

- The fixpoint W is the set of states from which we can enforce safety with probability 1.

Computing Safe Sets: Advanced Ideas

ϵ–Safe Set (known transition probabilities)

The ϵ–Safe Set S_ϵ and ϵ-**winning set** W_ϵ can be computed exactly via interval-bound value iteration or linear programming (similar to computing reward optimal policies).

Computing Safe Sets: Advanced Ideas

ϵ–Safe Set (known transition probabilities)

The ϵ–Safe Set S_ϵ and ϵ-**winning set** W_ϵ can be computed exactly via interval-bound value iteration or linear programming (similar to computing reward optimal policies).

ϵ–Safe Set (unknown transition probabilities)

When the transition probabilities are unknown we can interval or robust MDPs can estimate the ϵ–Safe Set via worst case reachability analysis.

Computing Safe Sets: Advanced Ideas

ϵ–Safe Set (known transition probabilities)

The ϵ–Safe Set S_ϵ and ϵ-**winning set** W_ϵ can be computed exactly via interval-bound value iteration or linear programming (similar to computing reward optimal policies).

ϵ–Safe Set (unknown transition probabilities)

When the transition probabilities are unknown we can interval or robust MDPs can estimate the ϵ–Safe Set via worst case reachability analysis.

Safety Abstractions

Often many environments have additional features that do not need to be modelled for safety analysis, and many states are bi-similar with respect to the labelling function and safety dynamics. Rather we can build a safety abstraction of the MDP $\text{Abs}(\mathcal{M})$ with far fewer states and conduct reachability analysis/safe set construction on the abstraction for efficiently.

Safety-relevant abstraction: Example

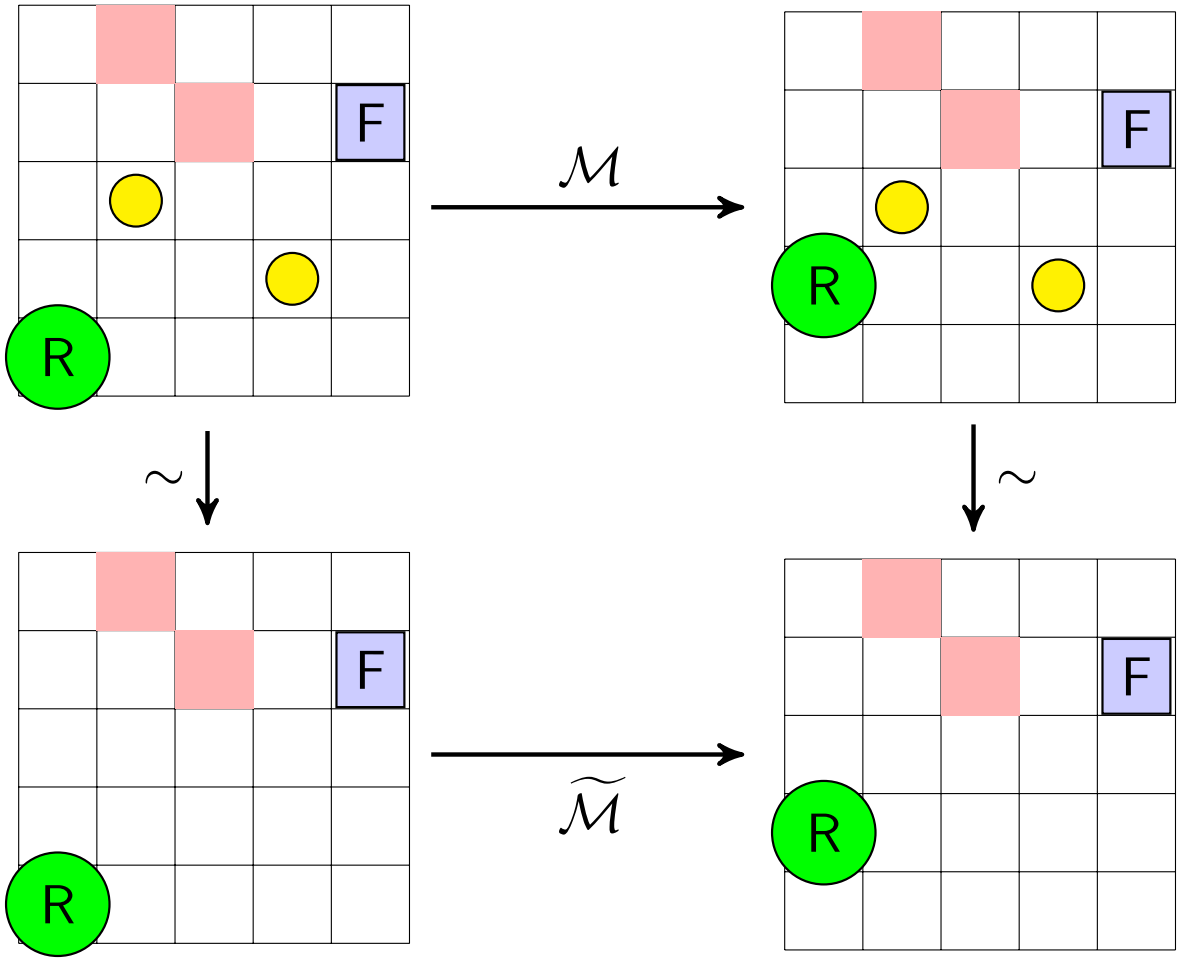


Figure: Illustration of the safety-relevant abstraction: $\text{Abs}(\mathcal{M})$ is the "same" MDP as $\mathcal{M}$ but without coins.

Demo: PacmanWithCoins

MASA-Safe-RL/tutorial/03_pacman_coins_safety_abstraction_demo.ipynb

Safety Compliance via Shielding

Safety Compliance via Shielding
Shielding Methodologies: Absolute Safety

Shielding Methodologies: Absolute Safety

Assumption (absolute safety is feasible)

We assume the initial state satisfies **almost sure** safety: $s_0^{\otimes} \in \text{AS}$.

Shielding Methodologies: Absolute Safety

Assumption (absolute safety is feasible)

We assume the initial state satisfies **almost sure** safety: $s_0^{\otimes} \in \text{AS}$.

Definition 31 (Winning (safe) action set)

Let AS be the almost sure safe set for the product MDP $\mathcal{M} \otimes \mathcal{D}$. For each product state $s^{\otimes} \in \text{AS}$ define the **winning set of actions**:

$$W(s^{\otimes}) := \left\{ a \in A(s^{\otimes}) \;\middle|\; \text{supp}\Big(P^{\otimes}(\cdot \mid s^{\otimes}, a)\Big) \subseteq \text{AS} \right\}.$$

Shielding Methodologies: Absolute Safety

Assumption (absolute safety is feasible)

We assume the initial state satisfies **almost sure** safety: $s_0^{\otimes} \in \mathsf{AS}$.

Definition 31 (Winning (safe) action set)

Let AS be the almost sure safe set for the product MDP $\mathcal{M} \otimes \mathcal{D}$. For each product state $s^{\otimes} \in \mathsf{AS}$ define the **winning set of actions**:

$$W(s^{\otimes}) := \left\{ a \in A(s^{\otimes}) \;\middle|\; \mathsf{supp}\Big(P^{\otimes}(\cdot \mid s^{\otimes}, a)\Big) \subseteq \mathsf{AS} \right\}.$$

Correct-by-construction shield

Construct a **shielded MDP** $\mathsf{Sh}(\mathcal{M})$ by restricting actions:

$$A^{\mathsf{Sh}}(s^{\otimes}) := W(s^{\otimes}) \qquad \text{for all } s^{\otimes} \in \mathsf{AS}.$$

Then *every* policy in $\mathsf{Sh}(\mathcal{M})$ is **absolutely safe**.

Absolute Safety: Algorithm

Algorithm 1 Shielding (Absolute Safety)

1: **Input:** An MDP $\mathcal{M}$, reward function R, labelling function L and safety formula φ.
2: Compute the set of safe actions A^{Sh}
3: Construct the shield $\mathsf{Sh}(\mathcal{M})$ by removing $A \setminus A^{\mathsf{Sh}}$ from $\mathcal{M}$.
4: Learn a memoryless policy $\pi^\star$ on $\mathsf{Sh}(\mathcal{M})$ with any RL algorithm.
5: **Return** $\pi^\star$.

Absolute Safety: Algorithm

Algorithm 2 Shielding (Absolute Safety)

1: **Input:** An MDP $\mathcal{M}$, reward function R, labelling function L and safety formula φ.
2: Compute the set of safe actions A^{Sh}
3: Construct the shield $\mathsf{Sh}(\mathcal{M})$ by removing $A \setminus A^{\mathsf{Sh}}$ from $\mathcal{M}$.
4: Learn a memoryless policy $\pi^\star$ on $\mathsf{Sh}(\mathcal{M})$ with any RL algorithm.
5: **Return** $\pi^\star$.

In practice:

- We replace actions proposed by the agent **on-the-fly**, if the agent proposes $a \notin A^{\mathsf{Sh}}$ then replace it with an action $a \in A^{\mathsf{Sh}}$ randomly or by a partial ordering on A.
- Guarantees **minimal interference**: we can obtain the optimal policy under absolute safety constraints.

Safety Compliance via Shielding
Shielding Methodologies: Probabilistic Safety

Shielding Methodologies: Probabilistic Safety

Assumption (probabilistic safety is feasible)

We assume the initial state satisfies safety **with high probability**: $s_0^{\otimes} \in \mathsf{S}_\epsilon$.

Shielding Methodologies: Probabilistic Safety

Assumption (probabilistic safety is feasible)

We assume the initial state satisfies safety **with high probability**: $s_0^{\otimes} \in \mathsf{S}_\epsilon$.

Goal

Learn an optimal policy such that the probability of ever violating the safety property is no more than ϵ. Define the set of safe policies,

$$\Pi^\epsilon := \{\pi \mid \Pr^\pi_{\mathcal{M}}(\rho \models \varphi) \geq 1 - \epsilon\}$$

for a given LTL safety property φ. Then let the full objective be,

$$\max_{\pi \in \Pi^\epsilon} \mathbb{E}_{s_t \sim P, a_t \sim \pi} \sum_{t=0} \gamma^t R(s_t, a_t)$$

Probabilistic Safety: Definitions

Quantitative "risk-to-failure" value

Let $\beta^*(s^{\otimes})$ denote the **minimal** probability of ever reaching the bad set (optimal avoidance):

$$\beta^*(s^{\otimes}) := \min_{\pi^{\otimes}} \Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}(\rho^{\times} \models \mathbf{F}\,\texttt{accept} \mid s^{\otimes}).$$

Probabilistic Safety: Definitions

Quantitative "risk-to-failure" value

Let $\beta^*(s^{\otimes})$ denote the **minimal** probability of ever reaching the bad set (optimal avoidance):

$$\beta^*(s^{\otimes}) := \min_{\pi^{\otimes}} \Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}(\rho^{\times} \models \mathbf{F}\,\texttt{accept} \mid s^{\otimes}).$$

Definition 32 (ϵ-winning action set)

Given a current **risk budget** ϵ_t, define:

$$\mathsf{W}_{\epsilon_t}(s^{\otimes}) := \left\{ a \in A(s^{\otimes}) \;\middle|\; \sum_{s'^{\otimes}} P^{\otimes}(s'^{\otimes} \mid s^{\otimes}, a)\, \beta^*(s'^{\otimes}) \leq \epsilon_t \right\}.$$

Risk Budgeting: Naïve Approach

Naïve risk budgeting (decreasing ϵ_t)

To obtain an end-to-end bound across time, we must control the **accumulated** risk. A simple scheme:

- choose a decreasing budget sequence $\epsilon_0 \geq \epsilon_1 \geq \epsilon_2 \geq \cdots$,

Risk Budgeting: Naïve Approach

Naïve risk budgeting (decreasing ϵ_t)

To obtain an end-to-end bound across time, we must control the **accumulated** risk. A simple scheme:

- choose a decreasing budget sequence $\epsilon_0 \geq \epsilon_1 \geq \epsilon_2 \geq \cdots$,
- at time t, permit only $a \in \mathsf{W}_{\epsilon_t}(s_t^{\otimes})$,

Risk Budgeting: Naïve Approach

Naïve risk budgeting (decreasing ϵ_t)

To obtain an end-to-end bound across time, we must control the **accumulated** risk. A simple scheme:

- choose a decreasing budget sequence $\epsilon_0 \geq \epsilon_1 \geq \epsilon_2 \geq \cdots$,
- at time t, permit only $a \in \mathsf{W}_{\epsilon_t}(s_t^{\otimes})$,
- if the agent proposes $a \notin \mathsf{W}_{\epsilon_t}(s_t^{\otimes})$, override with $\tilde{a} \in \mathsf{W}_{\epsilon_t}(s_t^{\otimes})$.
- Safety with probability $1 - \epsilon$ is guaranteed if $\sum_{t=0}^{\infty} \epsilon_t \leq \epsilon$ (union bound).

Risk Budgeting: Less Naïve (Adaptive)

Adaptive risk budgeting

Rather than fixing a decreasing sequence $\epsilon_0 \geq \epsilon_1 \geq \epsilon_2 \geq \cdots$ in advance, we keep track of the remaining risk budget β.

- Initialize $\beta_0 = \epsilon$.
- At timestep t the tail risk of the proposed action is given by,

$$\beta(s_t^{\otimes}, a) := \sum_{s'^{\otimes}} P^{\otimes}(s'^{\otimes} \mid s_t^{\otimes}, a_t)\, \beta^*(s'^{\otimes})$$

- Permit action a only if $\beta(s_t^{\otimes}, a) \leq \beta_t$; otherwise pick the safest possible action $\tilde{a} = \arg\min_{a'} \beta(s_t^{\otimes}, a')$.
- After executing the (safe) action update the risk budget with the update rule,

$$\beta_{t+1} \leftarrow \beta^*(s_{t+1}^{\otimes}) + \max\{\beta_t - \beta(s_t^{\otimes}, \tilde{a}_t), 0\}$$

- or $\beta \leftarrow 0$ if $s_{t+1}^{\otimes} \in F^{\otimes}$

Risk Budgeting: Optimality-preserving Approach

Optimality-preserving approach: augmented MDP with risk allocation

A more principled approach is to augment the state with remaining risk budget β:

$$\hat{s}_t = (s_t^{\otimes}, \beta_t), \qquad \beta_t \in [0, 1].$$

The agent selects **both** an action and a risk allocation (over successor states):

$$\hat{a}_t = (a_t, \alpha_1, \ldots \alpha_k) \quad \text{with } \alpha(s^{\otimes}) \geq \beta^*(s^{\otimes}).$$

Where k is the number of successor states. The shield enforces that the chosen action $(a_t, \alpha_1, \ldots \alpha_k)$ is feasible, i.e.,

$$\sum_{s'^{\otimes}} P^{\otimes}(s'^{\otimes} \mid s_t^{\otimes}, a_t)\, \alpha(s'^{\otimes}) \leq \beta_{t+1}.$$

Motivating Example (1)

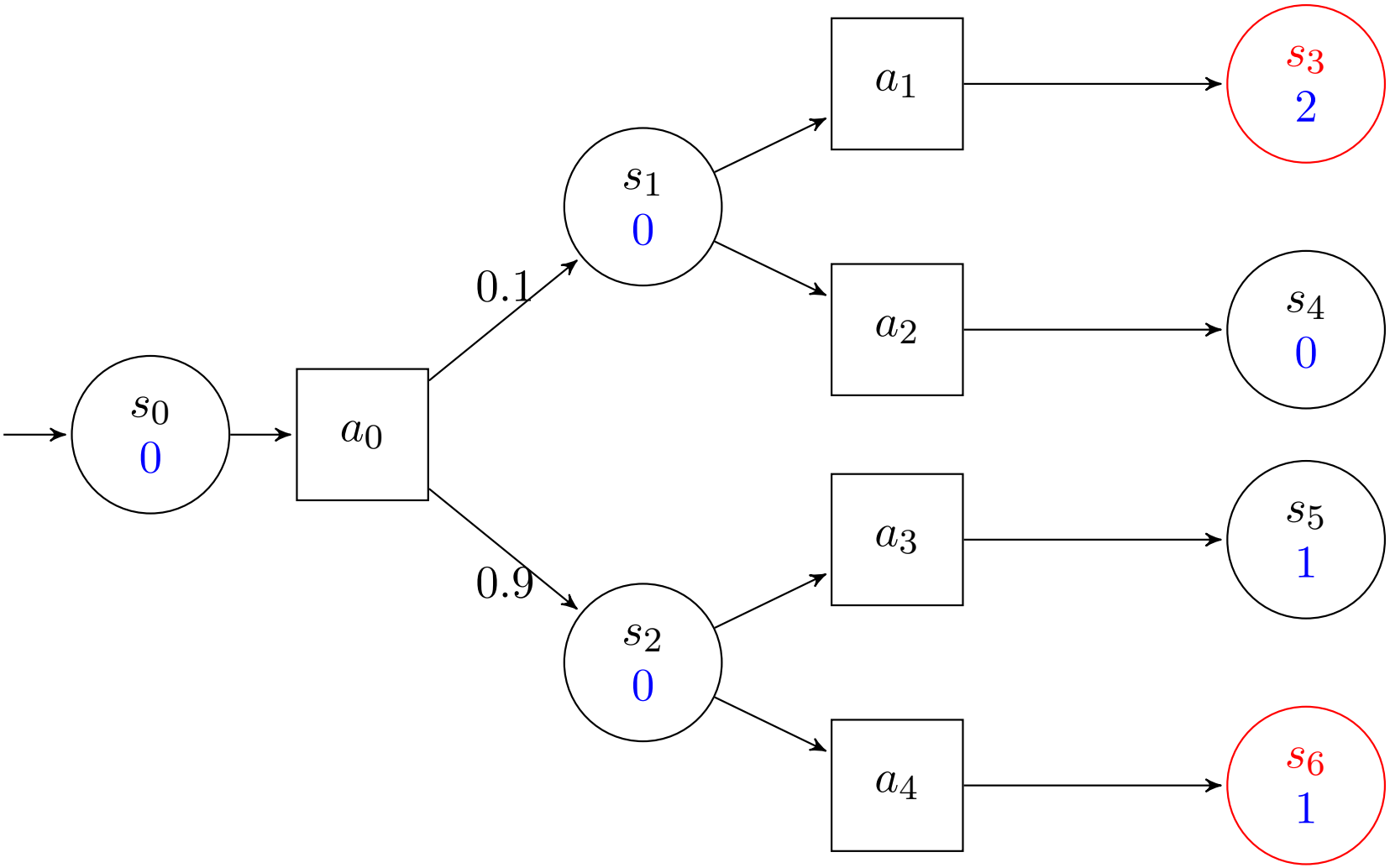


Figure: Example MDP with risk threshold $1 - \epsilon = 0.5$.

Motivating Example (2)

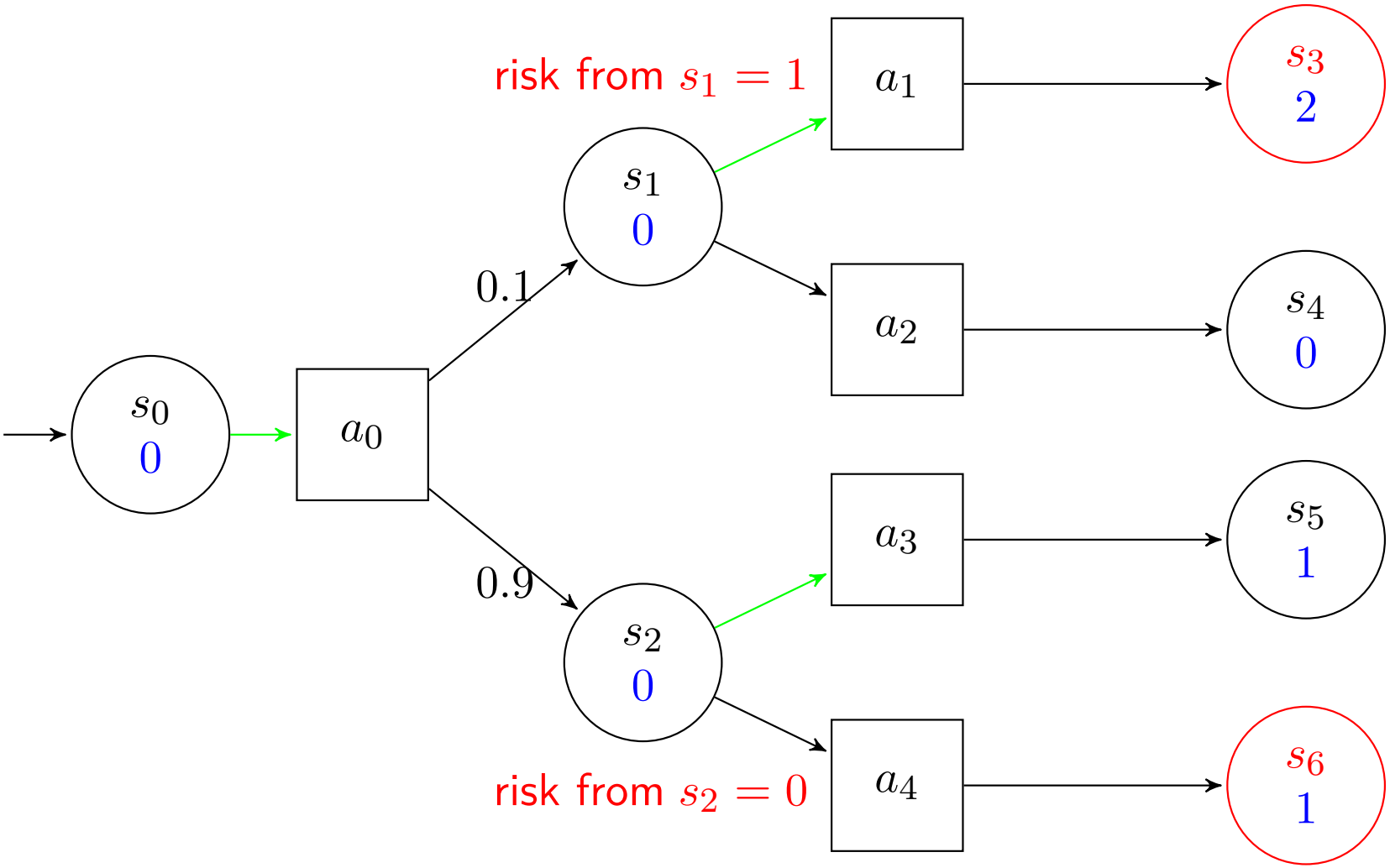


Figure: Example MDP with safety threshold $1 - \epsilon = 0.5$, optimal policy in green

Motivating example (3)

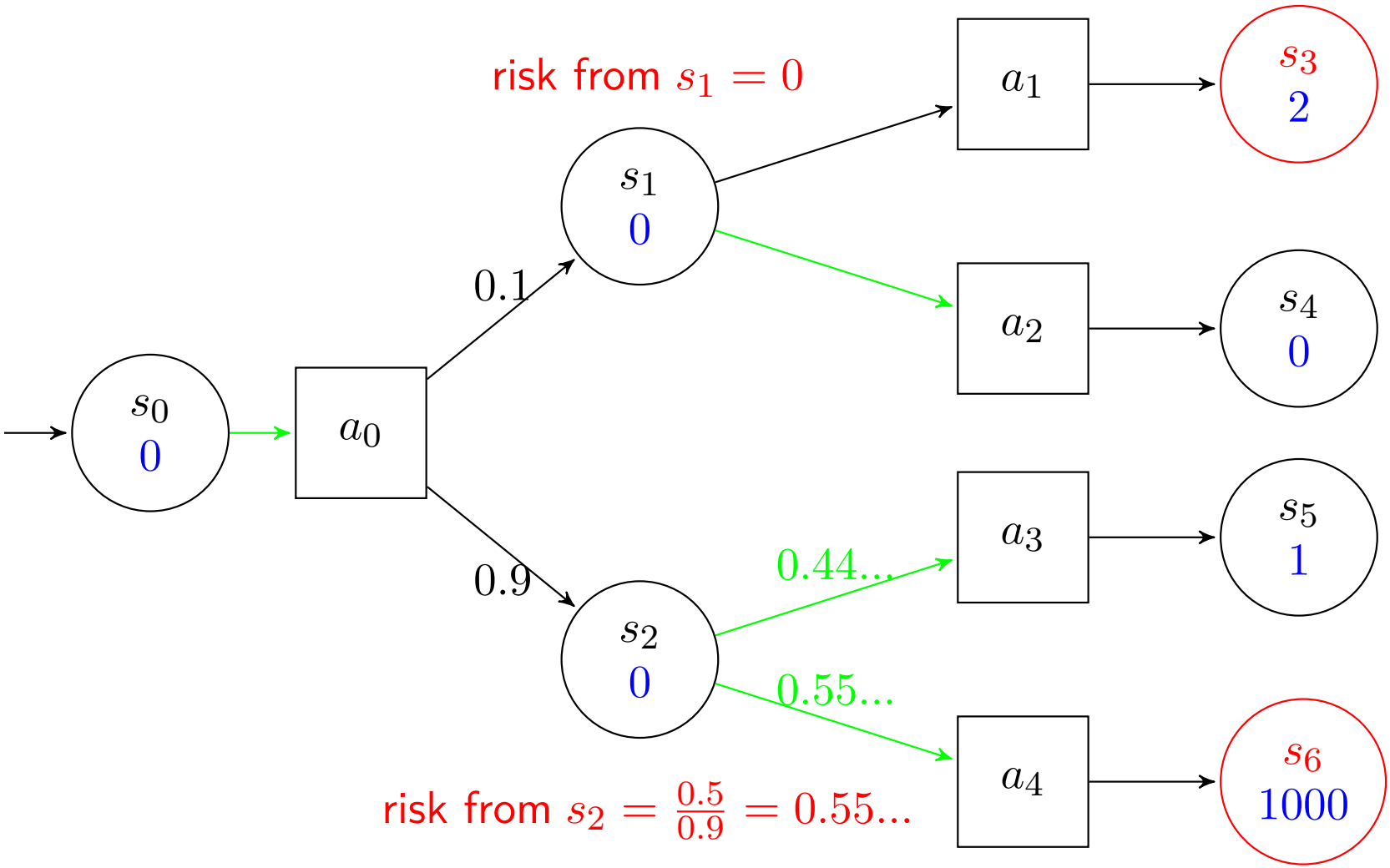


Figure: Example MDP with safety threshold $1 - \epsilon = 0.5$, optimal policy in green

Computing Minimal Reachability (optimal avoidance) Probability

Value Iteration

$$\beta_0^{\inf}(s) = 0 \text{ for all } s^{\otimes} \notin F^{\otimes},\ \beta_0^{\inf}(s) = 1 \text{ for all } s^{\otimes} \in F^{\otimes}$$

$$\beta_{n+1}^{\inf}(s) = \begin{cases} 1 & \text{if } s \in F^{\otimes} \\ \mathcal{B}_{\mathcal{M}}\left[\beta_n^{\inf}\right](s) & \text{otherwise,} \end{cases}$$

where $\mathcal{B}_{\mathcal{M}}[\beta](s) = \min_{a \in A} \sum_{s' \in S} P(s, a, s')\beta(s')$.

Sound Value Iteration

$$\beta_0^{\sup}(s) = \begin{cases} 0 & \text{if there is a safe policy from } s^{\otimes}, \\ 1 & \text{otherwise,} \end{cases}$$

$$\beta_{n+1}^{\sup}(s) = \begin{cases} 1 & \text{if } s^{\otimes} \in F^{\otimes} \\ \mathcal{B}_{\mathcal{M}}\left[\beta_n^{\sup}\right](s) & \text{otherwise,} \end{cases}$$

and terminate when the upper bound is δ-close the lower bound.

The Probabilistic Shield: Example

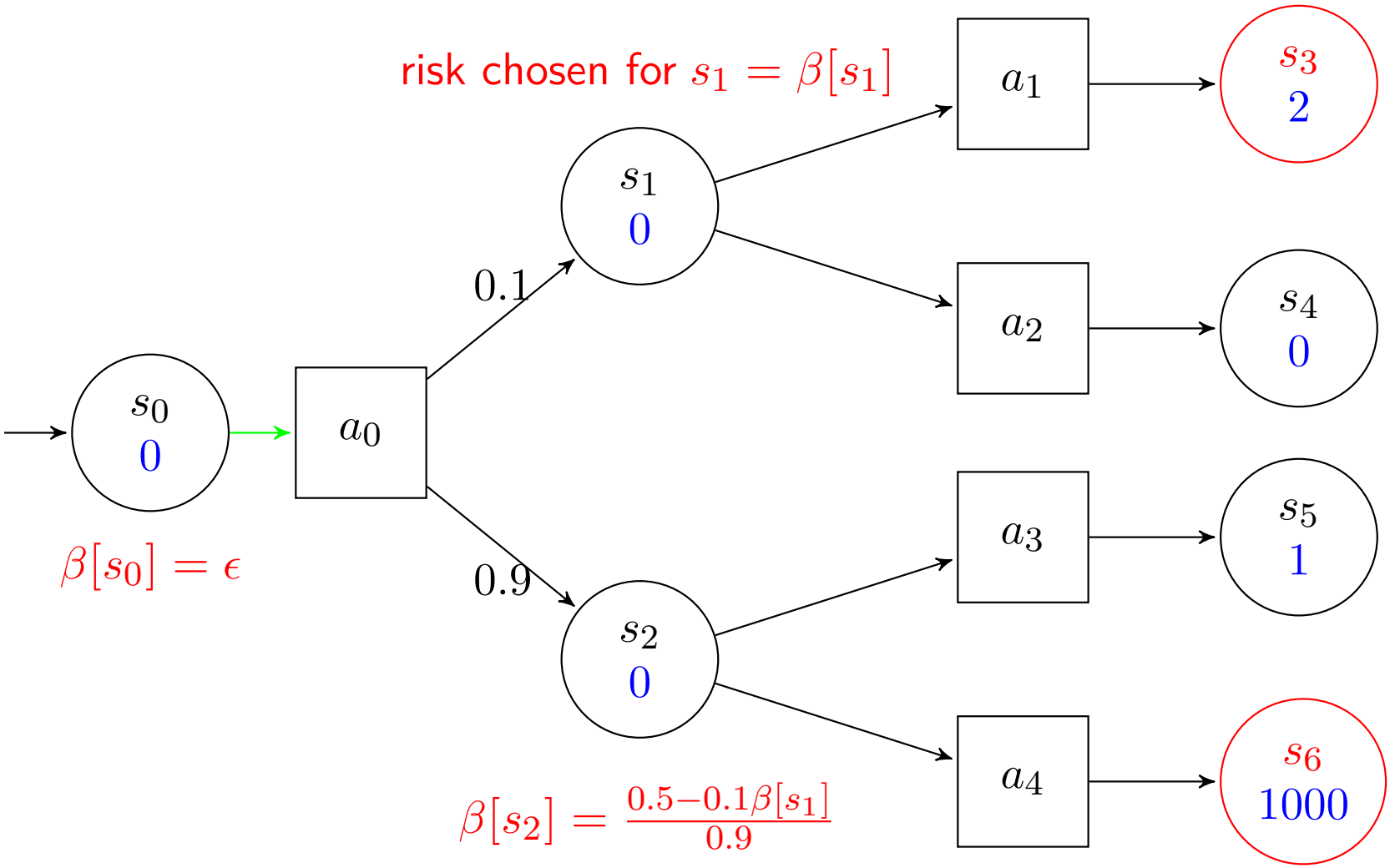


Figure: Example MDP with risk threshold $\epsilon = 0.5$

The Probabilistic Shielding: Full algorithm

Algorithm 3 Probabilistic Shielding

1: **Input:** An MDP $\mathcal{M}$, reward function R, labelling function L and safety formula φ (and DFA $\mathcal{D}$), initial safety threshold ϵ, SVI termination condition δ.

2: Compute an inductive δ-upper bound,

$$\beta(s^{\otimes}) \geq \min_{\pi^{\otimes}} \Pr^{\pi^{\otimes}}_{\mathcal{M}\otimes\mathcal{D}}(\rho^{\times} \models \mathbf{F}\,\texttt{accept} \mid s^{\otimes})$$

3: Construct the shield $\mathsf{Sh}(\mathcal{M})$ that enforces the condition,

$$\sum_{s'^{\otimes}} P^{\otimes}(s'^{\otimes} \mid s^{\otimes}_t, a_t)\,\alpha(s'^{\otimes}) \leq \beta_{t+1}$$

4: Learn a memoryless policy $\pi^{\star}$ in $\mathsf{Sh}(\mathcal{M})$ on actions $(a, \alpha_1, \ldots \alpha_k)$ with any RL algorithm.

5: **Return** $\pi^{\star}$.

Probabilistic Shielding: Implementation

Naive Approach (actions are full distributions)

The actions of $\text{Sh}(\mathcal{M})$ for a state (s, β) are all tuples $(\Delta, \alpha_1, \ldots \alpha_k)$ where,

- $\alpha_1, \ldots \alpha_k$ are the successor state risk levels (no less than β^*).
- Δ is a probability distribution such that $\sum_{a \in A(s)} \Delta(a) \left(\beta - \sum_{s' \in S} P(s, a, s')\alpha(s')\right) \geq 0$ (let's call this set $V_\alpha^{s,\beta}$).

Probabilistic Shielding: Implementation

Naive Approach (actions are full distributions)

The actions of $\text{Sh}(\mathcal{M})$ for a state (s, β) are all tuples $(\Delta, \alpha_1, \ldots \alpha_k)$ where,

- $\alpha_1, \ldots \alpha_k$ are the successor state risk levels (no less than β^*).
- Δ is a probability distribution such that $\sum_{a \in A(s)} \Delta(a) \left(\beta - \sum_{s' \in S} P(s, a, s')\alpha(s')\right) \geq 0$ (let's call this set $V_\alpha^{s,\beta}$).

More Efficient Approach (actions are extremal points)

The actions of $\text{Sh}(\mathcal{M})$ for a state (s, β) are all couples $(i, j, \alpha_1, \ldots \alpha_k)$ such that

- $\alpha_1, \ldots \alpha_k$ are the successor state risk levels (no less than β^*).
- (i, j) is an extremal point of $V_\alpha^{s,\beta}$ (in the simplex of Δ) – they are in finite number!.

Probabilistic Shielding: More Intuition

MASA-Safe-RL/notebooks/tutorials/09_probabilistic_shielding_minipacman.

Probabilistic Shielding: Safety Guarantee

Theorem 33 (Safety Guarantee)

The probabilistic shield defines an augmented state and augmented action MDP $Sh(\mathcal{M})$ where every policy is safe.

Corollary 34

We can transform any deterministic memoryless policy π of $Sh(\mathcal{M})$ to a memory-full stochastic policy $\widehat{\pi}$ of $\mathcal{M}$ with the same expected reward and expected cost.

Probabilistic Shielding: Optimality-Preserving Guarantee

Safety Guarantee in any Shield

We have the three following properties:

- There exists an optimal, memoryless, and deterministic policy $\pi^{\star}_{\beta}$ of $\mathsf{Sh}(\mathcal{M})$.
- The policy $\pi^{\star}_{\beta_{\mathcal{M}}}$ is a solution to the Safe RL problem.

Demo: Running Example

Demo: `MiniPacman` Setup

Recall the `MiniPacman` environment, with safety property $\mathbf{G}(\neg \texttt{ghost})$ and liveness property $\mathbf{F}(\texttt{food})$. We encode the liveness property via the reward function:

$$R(s,a) = \begin{cases} 1.0 & \text{if } \texttt{food} \in L(s) \\ 0.0 & \text{otherwise} \end{cases}$$

Demo Link

MASA-Safe-RL/tutorial/04_probabilistic_shielding_training_comparison.i

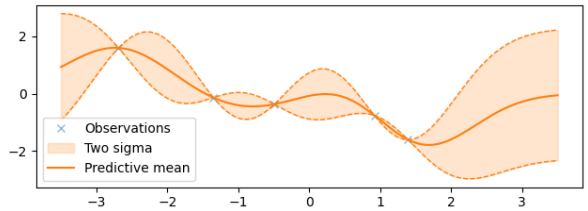
Safety Compliance via Shielding
Shielding without Prior Knowledge

Shielding: relaxing the assumptions and the requirements?

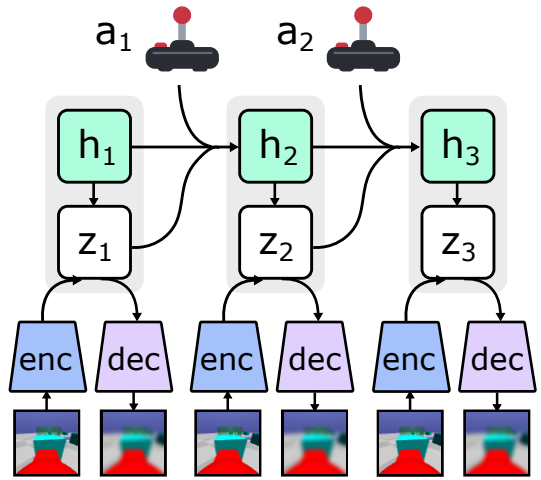
Question

What if the safety abstraction is unknown, or too large, and we do not need the output policy to be provably safe?

- Gaussian process dynamics models



- Model-based RL – world models (e.g. DreamerV3 RSSM Architecture)



Shielding: Gaussian process

- **Key idea:** check the action $a \sim \widehat{\pi}$ can be recovered to the control invariant set $\mathcal{X}_{\text{inv}}$ with a given backup policy π_{backup}, by propagating our uncertainty through the Gaussian process dynamics model.

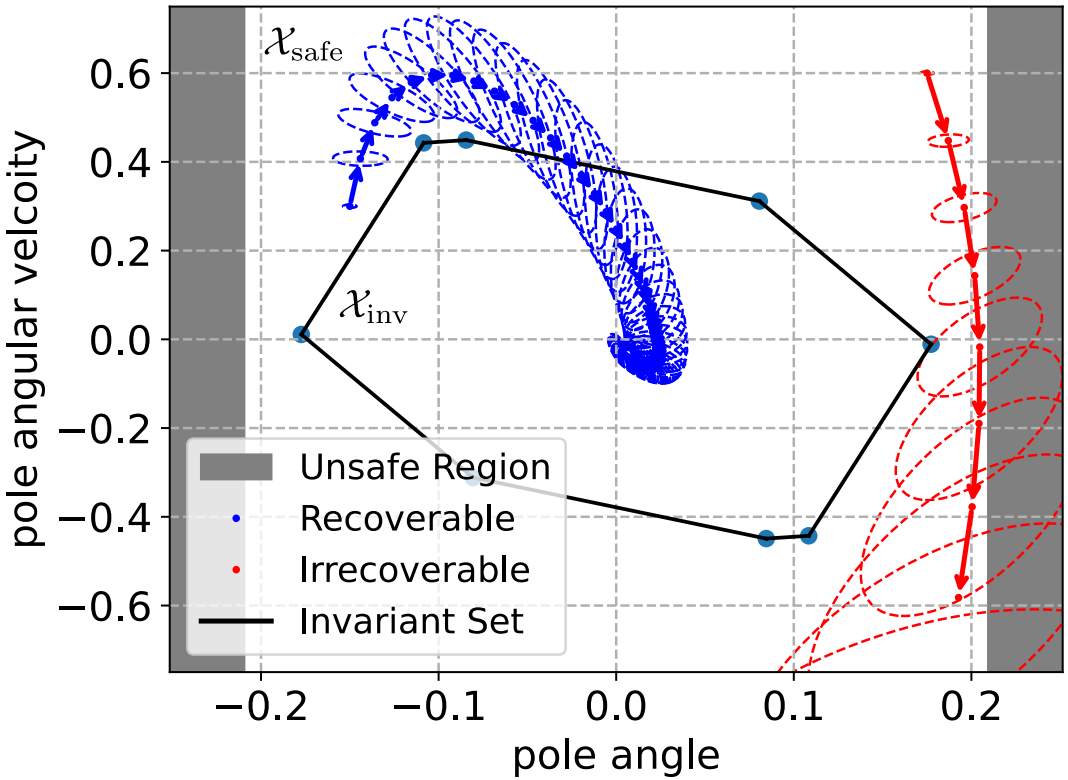


Figure: Uncertainty propagation through the Gaussian process dynamics model for Cartpole

Shielding: DreamerV3

- **Key idea:** check trajectories $\rho \sim p_\theta$ sampled in the world model satisfy the safety property φ, to estimate the probability $\Pr^{\pi}_{\mathcal{M}}(\{\rho \mid \rho \models \varphi\})$, if this probability is above a given threshold $(1-\epsilon)$, then the action $a \sim \widehat{\pi}$ is permissible otherwise a safe action $a' \sim \pi_{\text{backup}}$ is used.

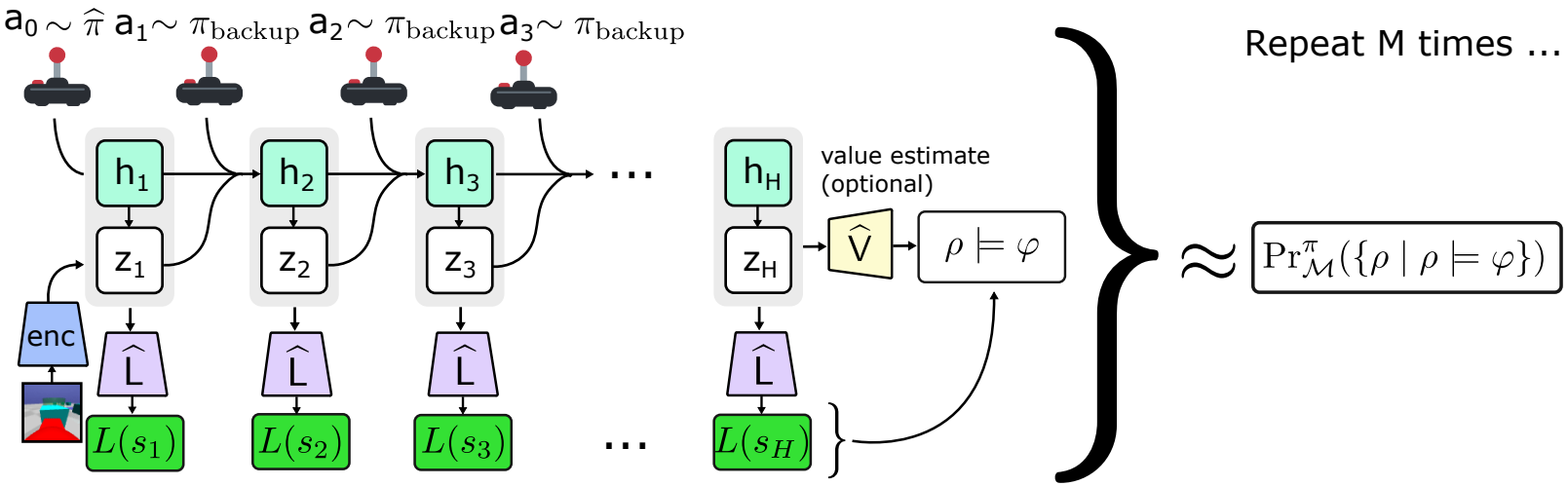


Figure: Estimating the safety probability with DreamerV3 RSSM

Shielding via Approximate Model Based Shielding (DreamerV3)

Seaquest: Goals and Constraints

- **Goal:** $\max_\pi \mathbb{E}_{s_t \sim P, a_t \sim \pi} \left[\sum_{t=0} \gamma^t R(s_t, a_t)\right]$.
- **Safety constraint:**
  $\varphi = \mathbf{G}(\neg \texttt{out-of-oxygen}) \wedge \mathbf{G}(\neg \texttt{surface} \rightarrow \mathbf{XG}(\texttt{surface} \rightarrow \texttt{diver}))$,

Qualitative Results (Goodall et. al 2023)

Shielded:

Unshielded:

References

References

- Salomon Sickert, Javier Esparza, Stefan Jaax, Jan Kretínský: Limit-Deterministic Büchi Automata for Linear Temporal Logic. CAV (2) 2016: 312-332
- Mohammadhosein Hasanbeig, Daniel Kroening, Alessandro Abate: Deep Reinforcement Learning with Temporal Logics. FORMATS 2020: 1-22
- Cameron Voloshin, Abhinav Verma, Yisong Yue: Eventual Discounting Temporal Logic Counterfactual Experience Replay. ICML 2023: 35137-35150
- Rajeev Alur, Suguman Bansal, Osbert Bastani, Kishor Jothimurugan: A Framework for Transforming Specifications in Reinforcement Learning. Principles of Systems Design 2022: 604-624
- Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, Miroslav Pajic: Control Synthesis from Linear Temporal Logic Specifications using Model-Free Reinforcement Learning. ICRA 2020: 10349-10355

- Wenjie Qiu, Wensen Mao, He Zhu: Instructing Goal-Conditioned Reinforcement Learning Agents with Temporal Logic Objectives. NeurIPS 2023
- Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, Ufuk Topcu: Safe Reinforcement Learning via Shielding. AAAI 2018: 2669-2678
- Goodall, Alexander W., and Francesco Belardinelli: Approximate model-based shielding for safe reinforcement learning. ECAI 2023: 883-890
- Mathias Jackermeier, Alessandro Abate: DeepLTL: Learning to Efficiently Satisfy Complex LTL Specifications. CoRR abs/2410.04631 (2024)
- Cameron Voloshin, Hoang Minh Le, Swarat Chaudhuri, Yisong Yue: Policy Optimization with Linear Temporal Logic Constraints. NeurIPS 2022

Thank you!

Thank you!