Formal Methods for Safe Multi-agent Reinforcement Learning

Omar Adalat, Dr. Francesco Belardinelli
With contributions from: Alex Goodall and Dr. Edwin Hamel de-le Court

Imperial College London

September 21, 2026

Key Open Problems in AI

AI systems are often deployed with no guarantees about their behaviour.

We aim at

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

[1] D. Amodei et al.; *Concrete Problems in AI Safety*, 2016.

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

Key Open Problems in AI

AI systems are often deployed with no guarantees about their behaviour.

We aim at

- **Generalisation** and **Robustness** w.r.t. out-of-distribution data.
- **Safety:** rewards with formal semantics (avoids reward misspecification/reward hacking/goal misgeneralization), safe exploration, requirements & constraints, risk management.[1]

[1] D. Amodei et al.; *Concrete Problems in AI Safety*, 2016.

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

Key Open Problems in AI

AI systems are often deployed with no guarantees about their behaviour.

We aim at

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

[1] D. Amodei et al.; *Concrete Problems in AI Safety*, 2016.

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

Key Open Problems in AI

AI systems are often deployed with no guarantees about their behaviour.

We aim at

- **Generalisation** and **Robustness** w.r.t. out-of-distribution data.
- **Safety:** rewards with formal semantics (avoids reward misspecification/reward hacking/goal misgeneralization), safe exploration, requirements & constraints, risk management.[1]
- **Security:** training data leakage, privacy, anonymity, robust to adversarial attacks.
- **AI Alignment:** AI that is aligned with human preferences and societal norms.[2]

[1] D. Amodei et al.; *Concrete Problems in AI Safety*, 2016.

[2] 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 for 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 for 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 for 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

MASA-Safe-RL: an open-source library to train safe agents

What is MASA-Safe-RL?

- 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.

MASA-Safe-RL: an open-source library to train safe agents

What is MASA-Safe-RL?

- 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/`

MASA-Safe-RL: an open-source library to train safe agents

What is MASA-Safe-RL?

- 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
   - Computation of Winning Regions
5. Shielding Multi-agent Systems
6. References

Why RL and Formal Methods?

Safe Learning Problem

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

Safe Learning Problem

How can we guarantee that learning agents will comply with 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)
- healthcare
- manufacturing (CNC machinery)
- ...

Safe Learning Problem

How can we guarantee that learning agents will comply with 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)
- healthcare
- manufacturing (CNC machinery)
- ...

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

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

Related Research Questions

- How do we formally define *safety*?

Related Research Questions

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

Related Research Questions

- How do we formally define *safety*?
- Are there 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 there 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 there 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 Pledge

By the end, you will 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 Pledge

By the end, you will 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 will also have a nice introduction into how to implement this things in practice with standard RL interfaces such as `Gymnasium` *and* `PettingZoo` *via the MASA-Safe-RL library.*

Running Example

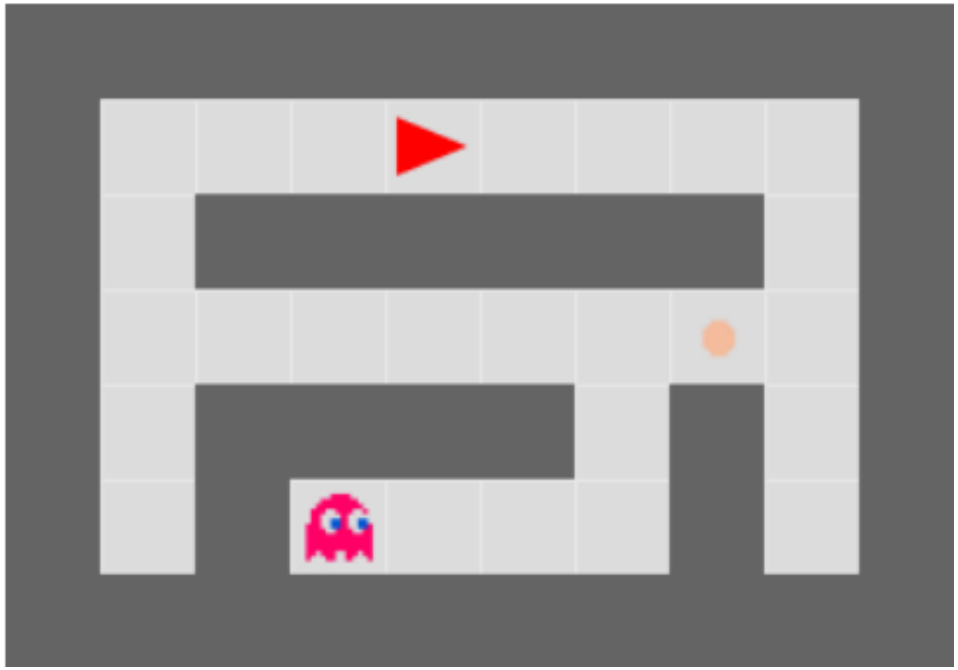


Figure: `MiniPacman` game.

Running Example

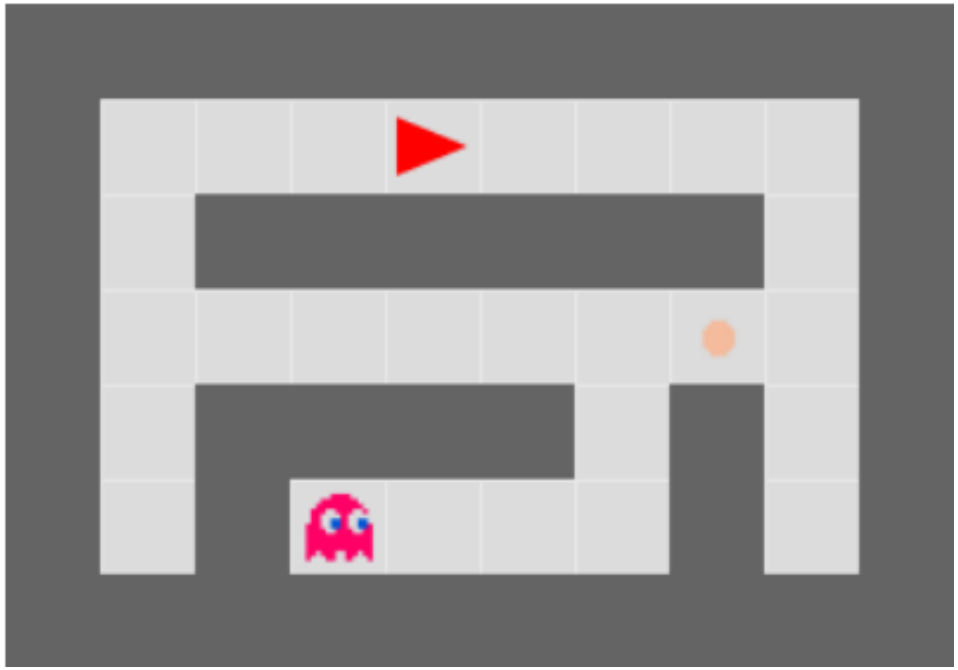


Figure: `MiniPacman` game.

Informally,

- **Goal**: "collect the food".

Running Example

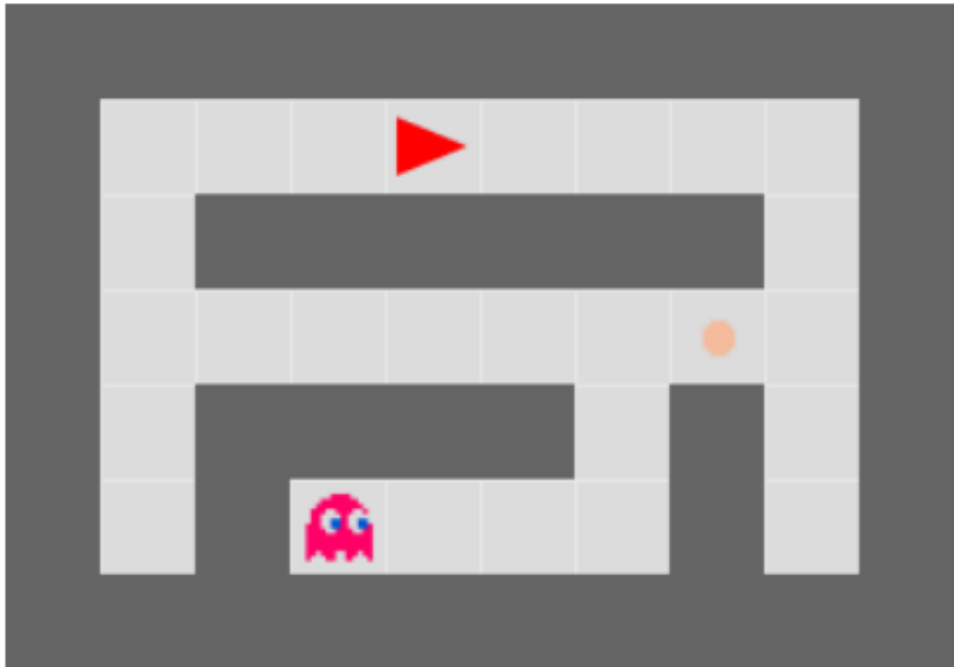


Figure: `MiniPacman` game.

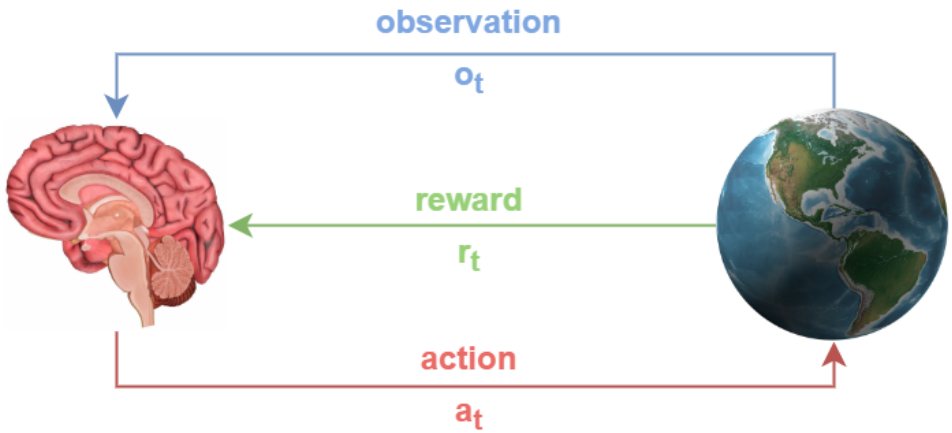
Informally,

- **Goal**: "collect the food".
- **Safety property**: "avoid the ghost".

What is Reinforcement Learning?

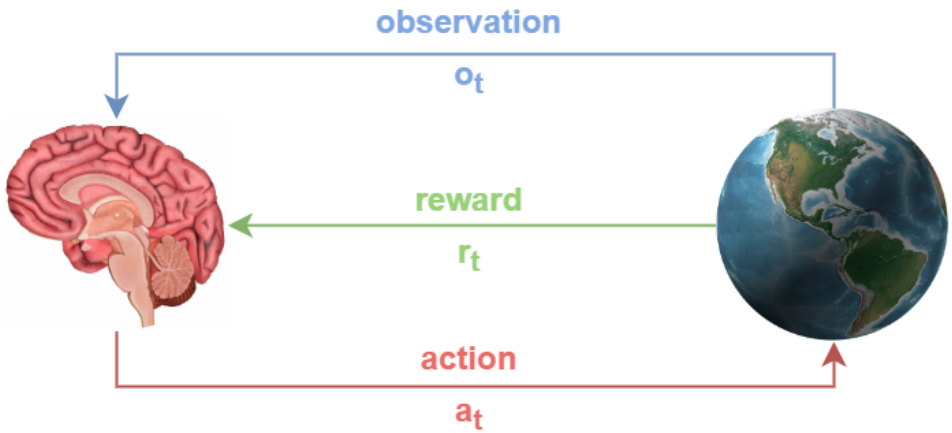
Agent-environment Interaction

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



Agent-environment Interaction

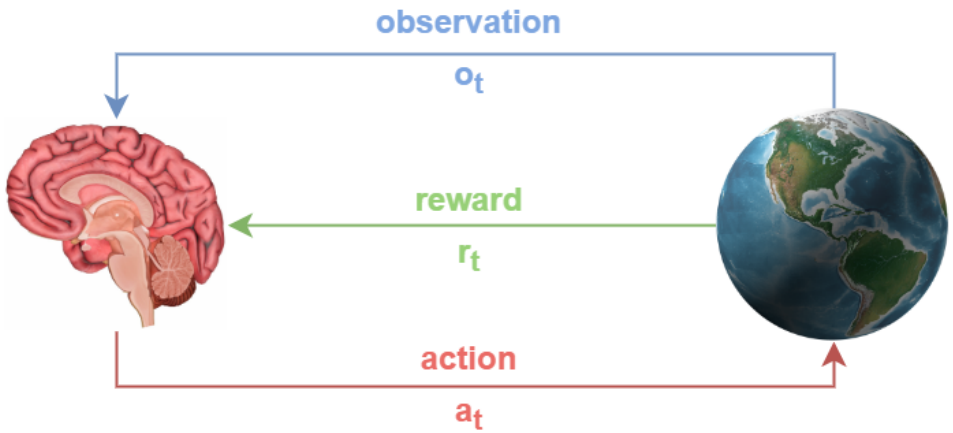
- In reinforcement learning (RL) the agent explores an unknown environment to maximize accumulated rewards.



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

Agent-environment Interaction

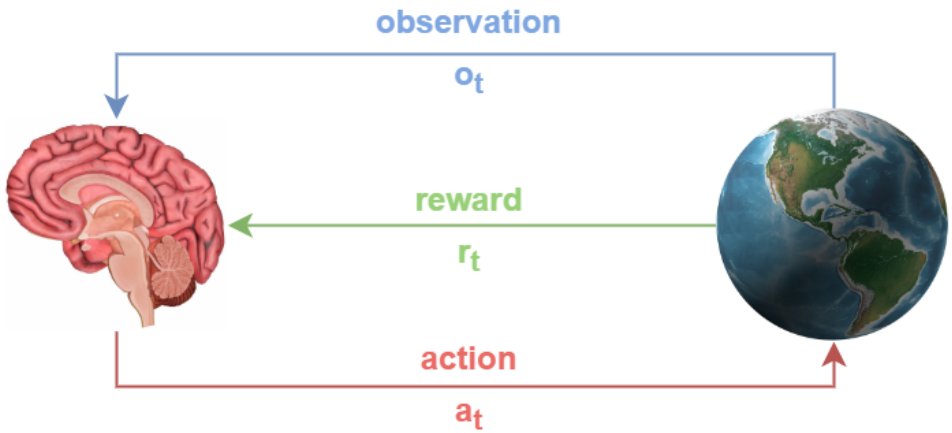
- In reinforcement learning (RL) the agent explores an unknown environment to maximize accumulated rewards.



- 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 vs. exploitation trade-off is a key problem in RL: how does the agent know that they have collected enough data to starting exploiting what they know?

Running Example (revisited)

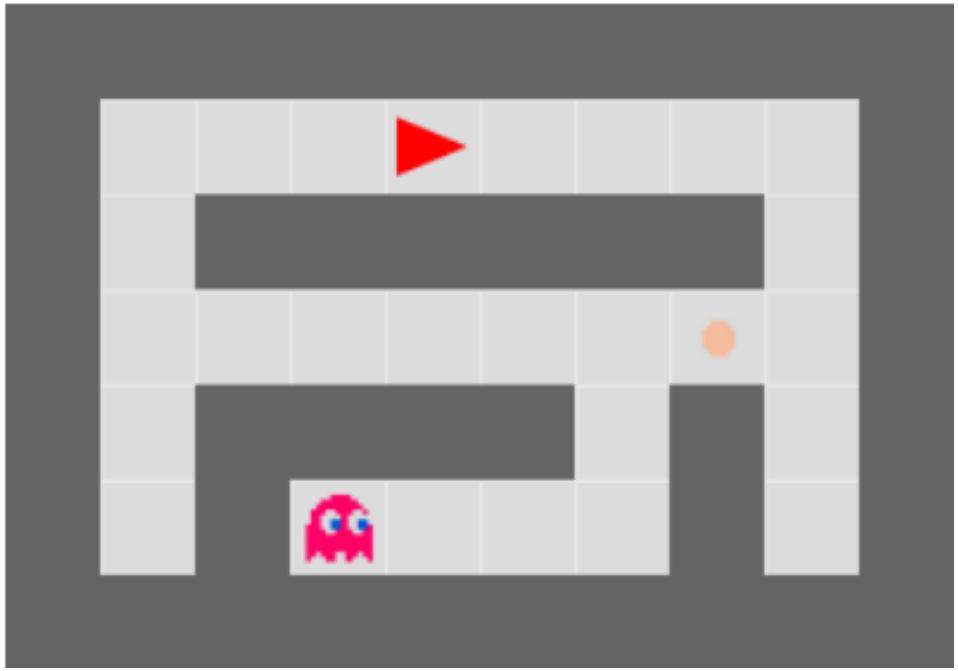


Figure: `MiniPacman` game.

Running Example (revisited)

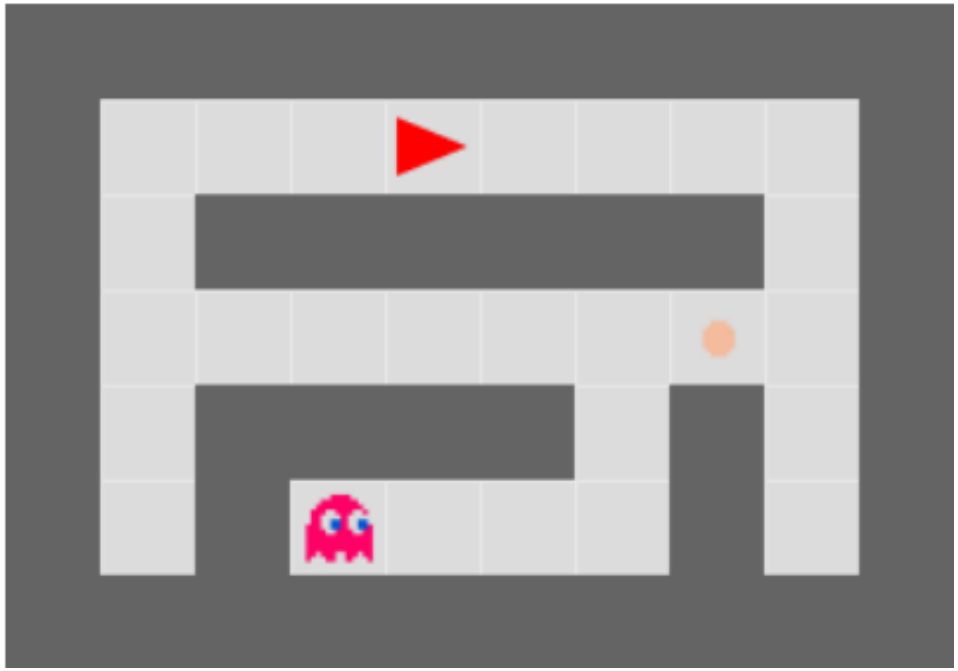


Figure: `MiniPacman` game.

- Observations: ?

Running Example (revisited)

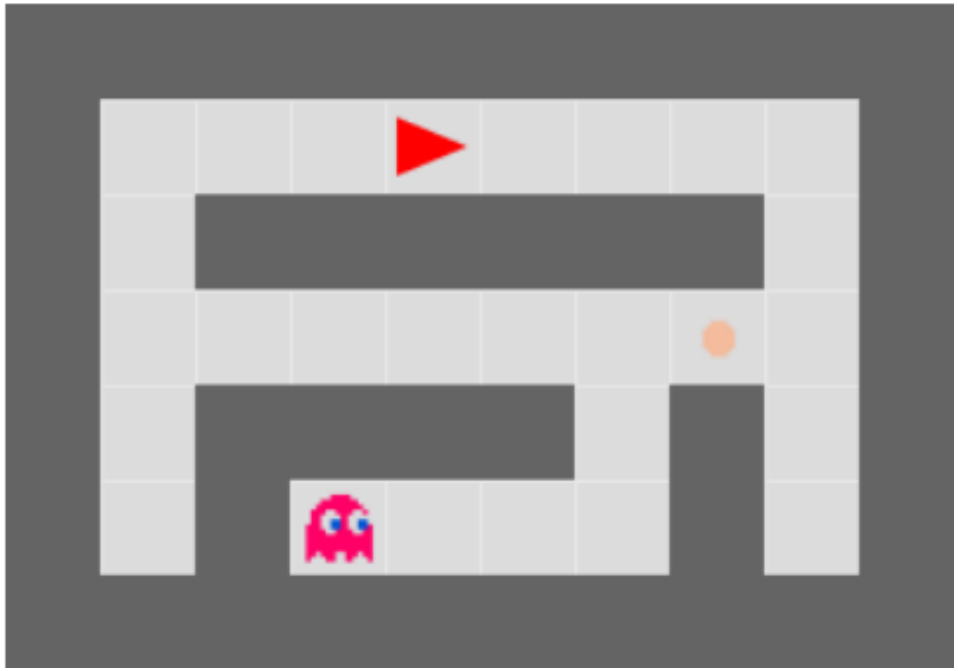


Figure: `MiniPacman` game.

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

Running Example (revisited)

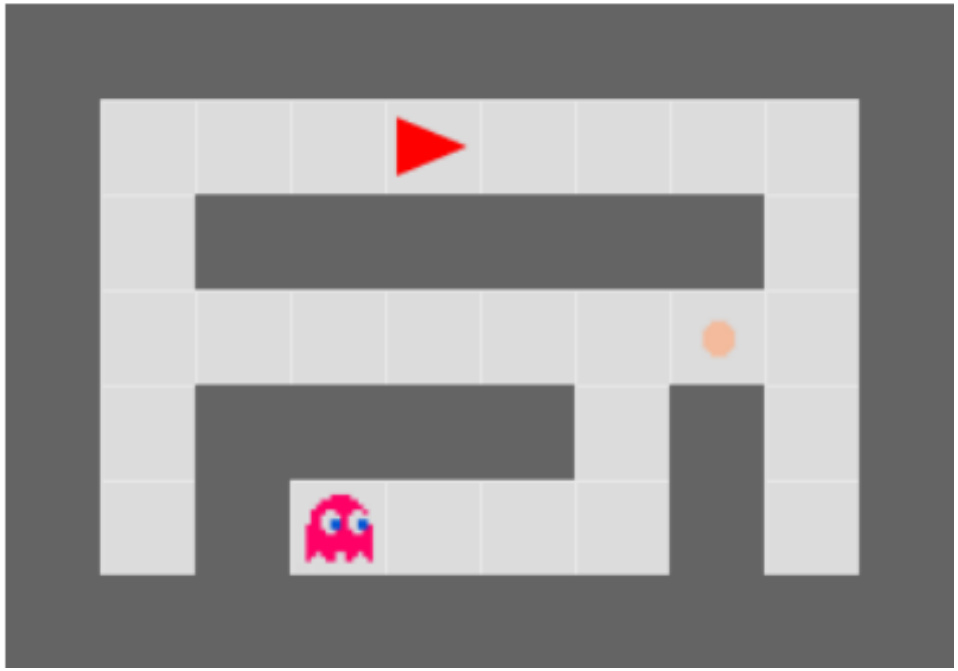


Figure: `MiniPacman` game.

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

Running Example (revisited)

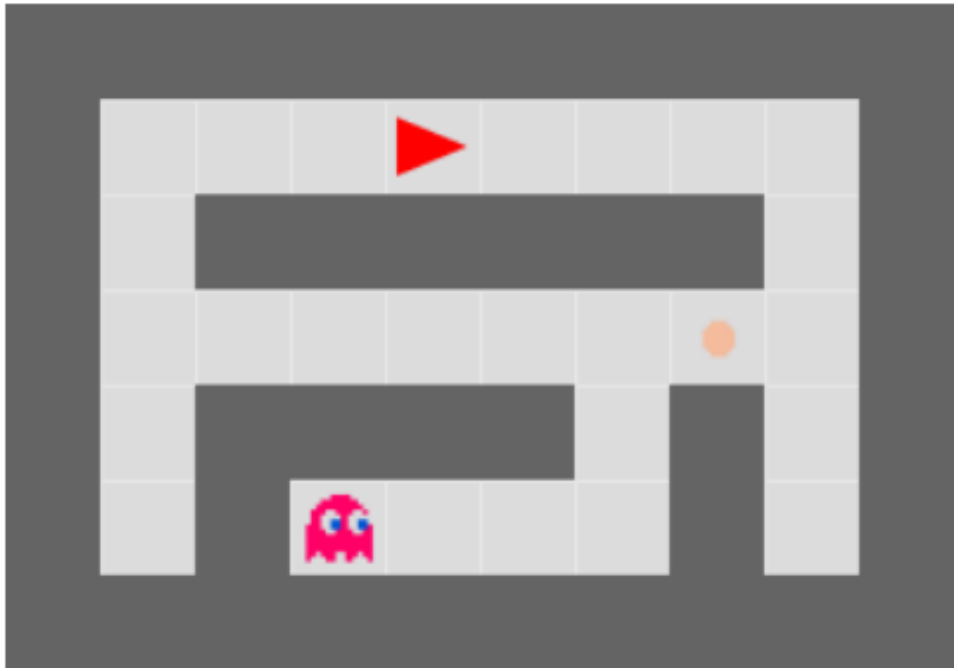


Figure: `MiniPacman` game.

- 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

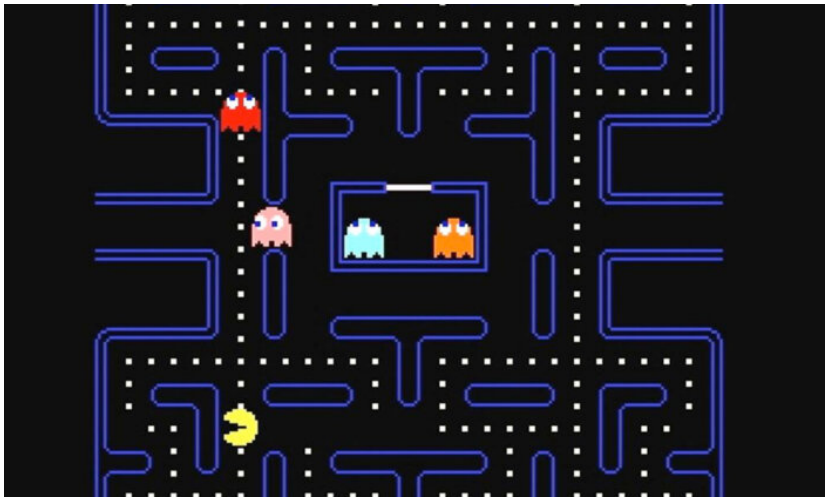
Modelling the environment

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., Self-driving cars, chip design, Minecraft)
- Successful applications in games (e.g., AlphaGo), robotics, finance, ...
- One of the key components of LLMs (e.g., RLHF and 'reasoning' models)

Reinforcement Learning in the Wild



Pac-Man

AlphaGo vs. Lee Sedol (Go)

Dactyl solving a Rubik's Cube

OpenAI Five (Dota 2)

Markov Decision Processes: A First Example

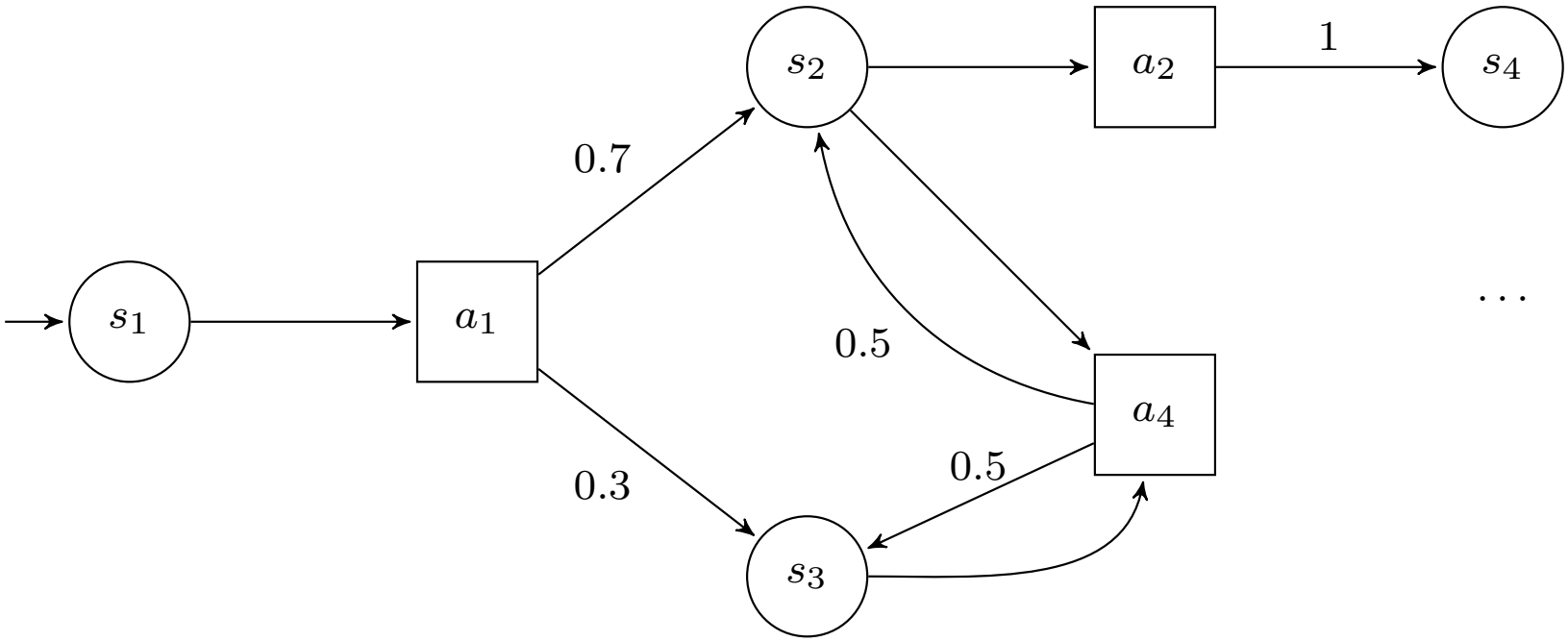


Figure: Example MDP

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.
- 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 (Path)

A *finite (resp. infinite) path* 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 $P(s, a)$.

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

Markov Decision Processes: Paths

Definition 2 (Path)

A *finite (resp. infinite) path* 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 $P(s, a)$.

Let 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: $\rho_2 = s_1 a_1 (s_3 a_4)^\omega$, $\rho_3 = s_1 a_1 (s_3 a_4 s_2 a_4)^\omega$. Note that for a finite block u that u^ω denotes infinite repetition of that block.

Policies

Definition 3 (Policy)

A **policy** π in an MDP $\mathcal{M}$ 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* (a.k.a. Markovian/positional) if, for every finite path $\rho = s_0, a_0, \ldots, s_n$, $\pi(\rho)$ only depends on s_n.

Example of a Stochastic Policy

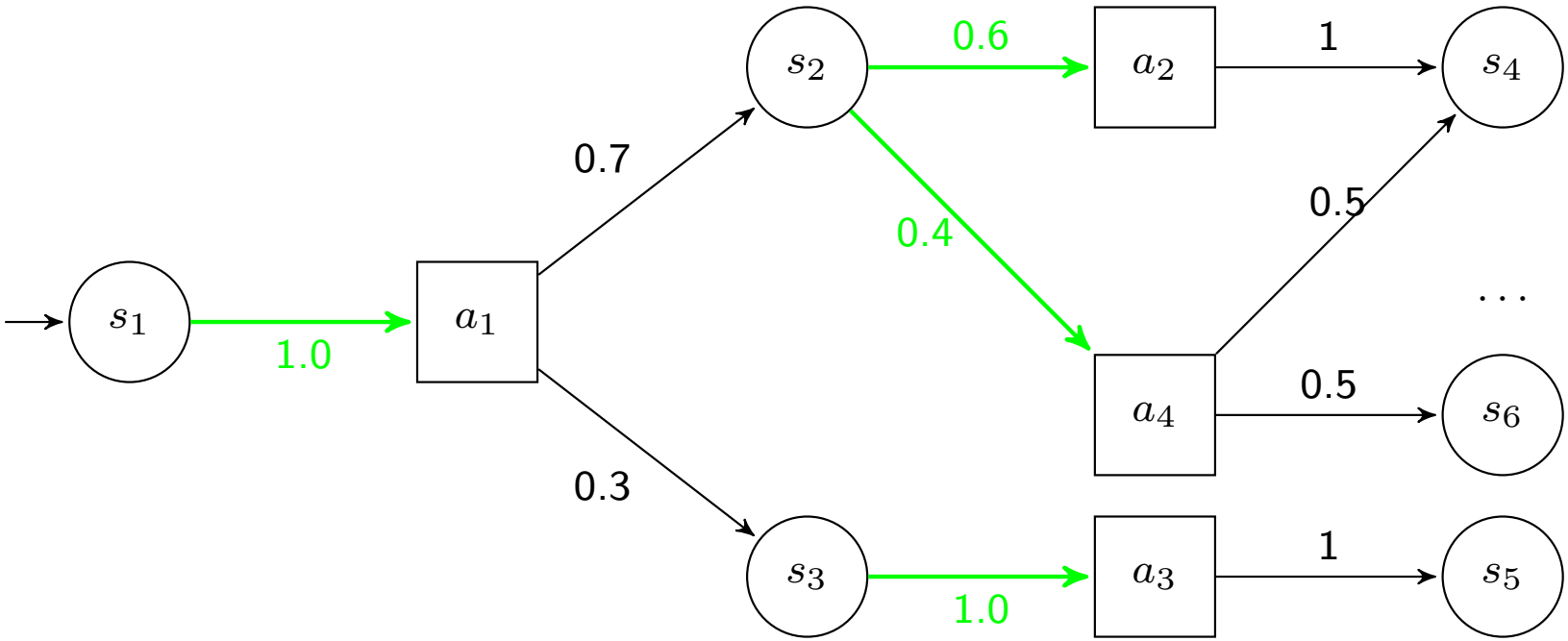


Figure: Example: Stochastic Policy

Example of a Deterministic Policy

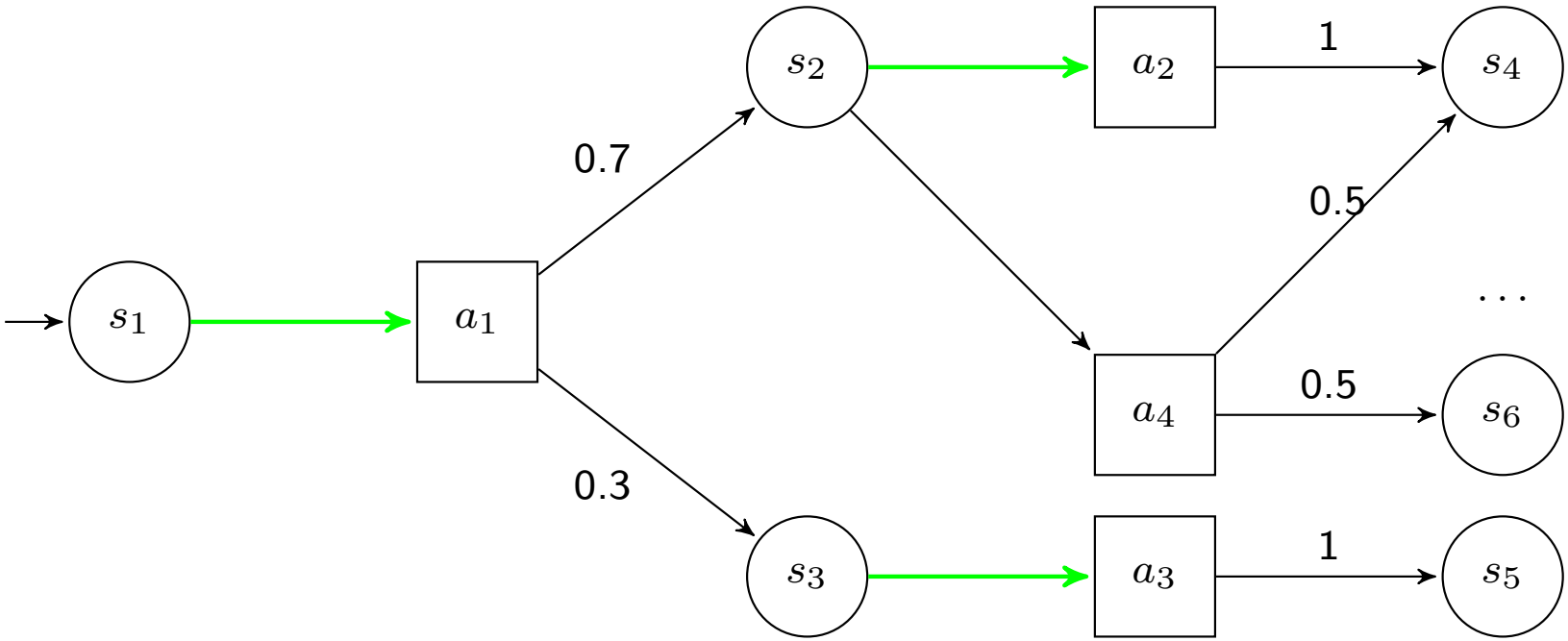


Figure: Example: Deterministic Policy

Markov Decision Processes with Rewards

Definition 4 (MDP with Rewards)

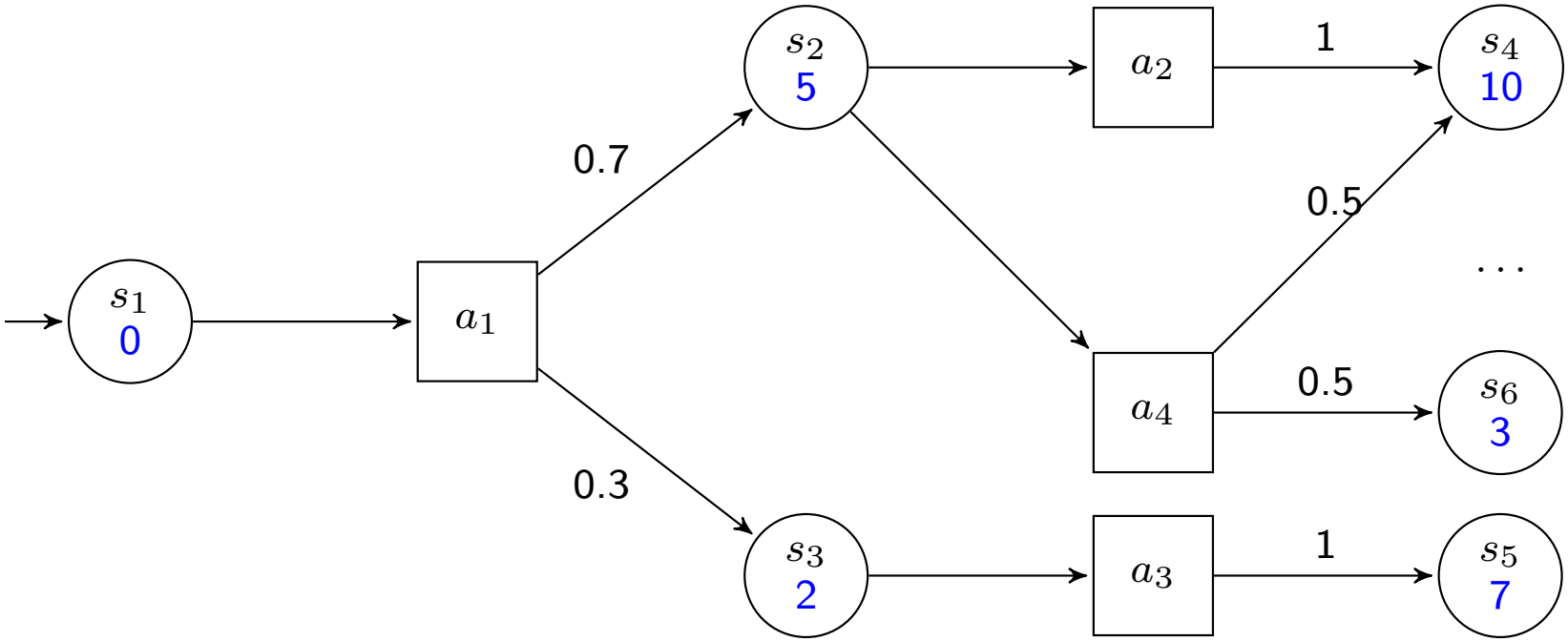
An **MDP with rewards** is an MDP $\mathcal{M} = (S, s_0, A, P, R, \gamma)$ equipped with

- 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 $\mathbb{R}$;
- a discount factor $\gamma \in [0, 1]$ that discounts future rewards.

Remark

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

MDP with Rewards: Example



Reward Optimal Policy

Definition 5 (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 5 (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 6 (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 5 (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 6 (Optimal Policy)

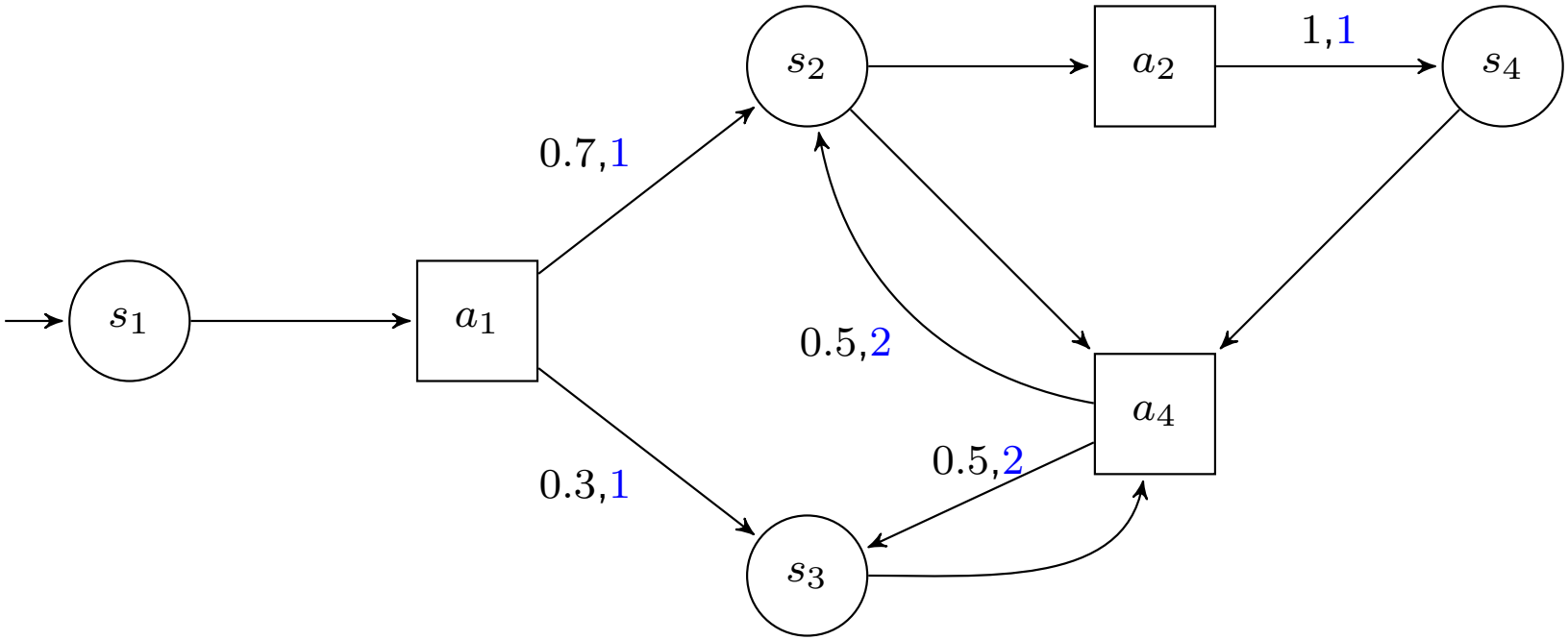
A policy is **optimal** if its expected cumulative reward is maximal amongst all available policies:

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

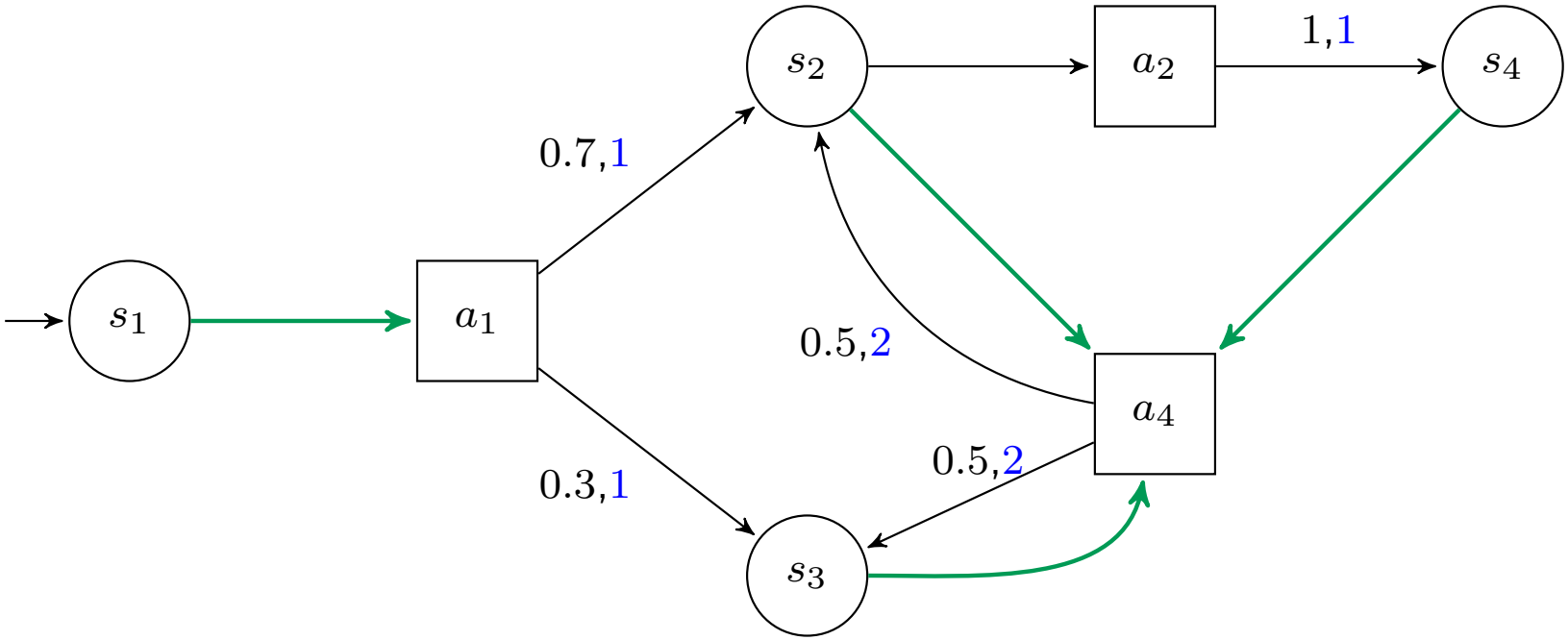
Remark

Memoryless policies are sufficient for reward optimal policies in MDPs.

Optimal Policy: Example



Optimal Policy: Example



Optimal policy π^* with expected cumulative discounted return:

$$1 + 2(\gamma + \gamma^2 + \ldots) = 1 + 2\frac{\gamma}{1-\gamma}$$

Value Functions for a Fixed Policy

State-value function

For a policy π, the **state-value function**

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s\right]$$

is the expected cumulative reward obtained by starting from state s and then following policy π.

Value Functions for a Fixed Policy

State-value function

For a policy π, the **state-value function**

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s\right]$$

is the expected cumulative reward obtained by starting from state s and then following policy π.

Action-value function

The **action-value function** (or **Q-value**)

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s,\ a_0 = a\right]$$

is the expected cumulative reward obtained by taking action a in state s, and then following policy π.

Optimal Value Functions

Optimal state-value function

The **optimal state-value function**

$$V^*(s) = \max_\pi V^\pi(s) = \max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s \right]$$

is the maximum expected cumulative reward achievable starting from state s.

Optimal Value Functions

Optimal state-value function

The **optimal state-value function**

$$V^*(s) = \max_\pi V^\pi(s) = \max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s \right]$$

is the maximum expected cumulative reward achievable starting from state s.

Optimal action-value function

The **optimal action-value function** (or **optimal Q-value**)

$$Q^*(s, a) = \max_\pi Q^\pi(s, a) = \max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s,\ a_0 = a \right]$$

is the maximum expected cumulative reward achievable after taking action a in state s.

From Optimal Q-values to an Optimal Policy

Choosing an optimal action

If we know Q^*, an optimal policy can choose an action with maximal Q-value in every state:

$$\pi^*(s) \in \arg \max_{a \in A(s)} Q^*(s, a).$$

From Optimal Q-values to an Optimal Policy

Choosing an optimal action

If we know Q^*, an optimal policy can choose an action with maximal Q-value in every state:

$$\pi^*(s) \in \arg \max_{a \in A(s)} Q^*(s, a).$$

Interpretation

At each state s:

1. Evaluate the available actions using $Q^*(s, a)$.
2. Choose an action with maximal Q-value.
3. Repeat at the next state.

From Optimal Q-values to an Optimal Policy

Choosing an optimal action

If we know Q^*, an optimal policy can choose an action with maximal Q-value in every state:

$$\pi^*(s) \in \arg \max_{a \in A(s)} Q^*(s, a).$$

Interpretation

At each state s:

1. Evaluate the available actions using $Q^*(s, a)$.
2. Choose an action with maximal Q-value.
3. Repeat at the next state.

Takeaway

Knowing Q^* is sufficient to synthesize an optimal policy.

Finding Optimal Policies: Known MDPs

When the environment model is known

Suppose the transition probabilities $P(s' \mid s, a)$ and reward function $R(s, a)$ are known.

Then finding an optimal policy is a **planning** problem: no interaction with the environment is required to learn the dynamics.

Finding Optimal Policies: Known MDPs

When the environment model is known

Suppose the transition probabilities $P(s' \mid s, a)$ and reward function $R(s, a)$ are known.

Then finding an optimal policy is a **planning** problem: no interaction with the environment is required to learn the dynamics.

Common approaches

- **Dynamic programming**
  - Value iteration.
  - Policy iteration.

Finding Optimal Policies: Known MDPs

When the environment model is known

Suppose the transition probabilities $P(s' \mid s, a)$ and reward function $R(s, a)$ are known.

Then finding an optimal policy is a **planning** problem: no interaction with the environment is required to learn the dynamics.

Common approaches

- **Dynamic programming**
  - Value iteration.
  - Policy iteration.
- **Linear programming**
  - Optimize the expected cumulative reward.
  - Bellman equations are encoded as constraints.

Finding Optimal Policies: Unknown MDPs

Reinforcement Learning

In reinforcement learning, the transition dynamics and/or reward structure are generally not fully known in advance.

The agent must learn how to act through interaction with the environment:

$$s_t \xrightarrow{a_t} (r_t, s_{t+1}).$$

Finding Optimal Policies: Unknown MDPs

Reinforcement Learning

In reinforcement learning, the transition dynamics and/or reward structure are generally not fully known in advance.

The agent must learn how to act through interaction with the environment:

$$s_t \xrightarrow{a_t} (r_t, s_{t+1}).$$

Common approaches

- **Value-based methods** learn action values and derive a policy from them.
  - Q-learning aims to learn $Q^*(s, a)$ directly.
  - Example: Deep Q-Networks (DQN).

Finding Optimal Policies: Unknown MDPs

Reinforcement Learning

In reinforcement learning, the transition dynamics and/or reward structure are generally not fully known in advance.

The agent must learn how to act through interaction with the environment:

$$s_t \xrightarrow{a_t} (r_t, s_{t+1}).$$

Common approaches

- **Value-based methods** learn action values and derive a policy from them.
  - Q-learning aims to learn $Q^*(s, a)$ directly.
  - Example: Deep Q-Networks (DQN).
- **Explicit policy methods** directly learn a parameterised policy $\pi_\theta(a \mid s)$.
  - Examples: REINFORCE, PPO and SAC.

Actor–critic methods

Many methods with an explicit policy also learn a **critic**:

- the *actor* is the policy $\pi_\theta(a \mid s)$,
- the *critic* estimates the value of the current policy, e.g., $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s \right]$, and is used to evaluate and improve the actor.

Modern Reinforcement Learning Methods

Actor–critic methods

Many methods with an explicit policy also learn a **critic**:

- the *actor* is the policy $\pi_\theta(a \mid s)$,
- the *critic* estimates the value of the current policy, e.g., $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, s_0 = s \right]$, and is used to evaluate and improve the actor.

On-policy vs. off-policy

- **On-policy:** primarily learns from experience generated by the current policy. For example, PPO is an **on-policy actor–critic** method.
- **Off-policy:** can reuse experience generated by previous or different behaviour policies. For example, SAC is an **off-policy actor–critic** method.

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.

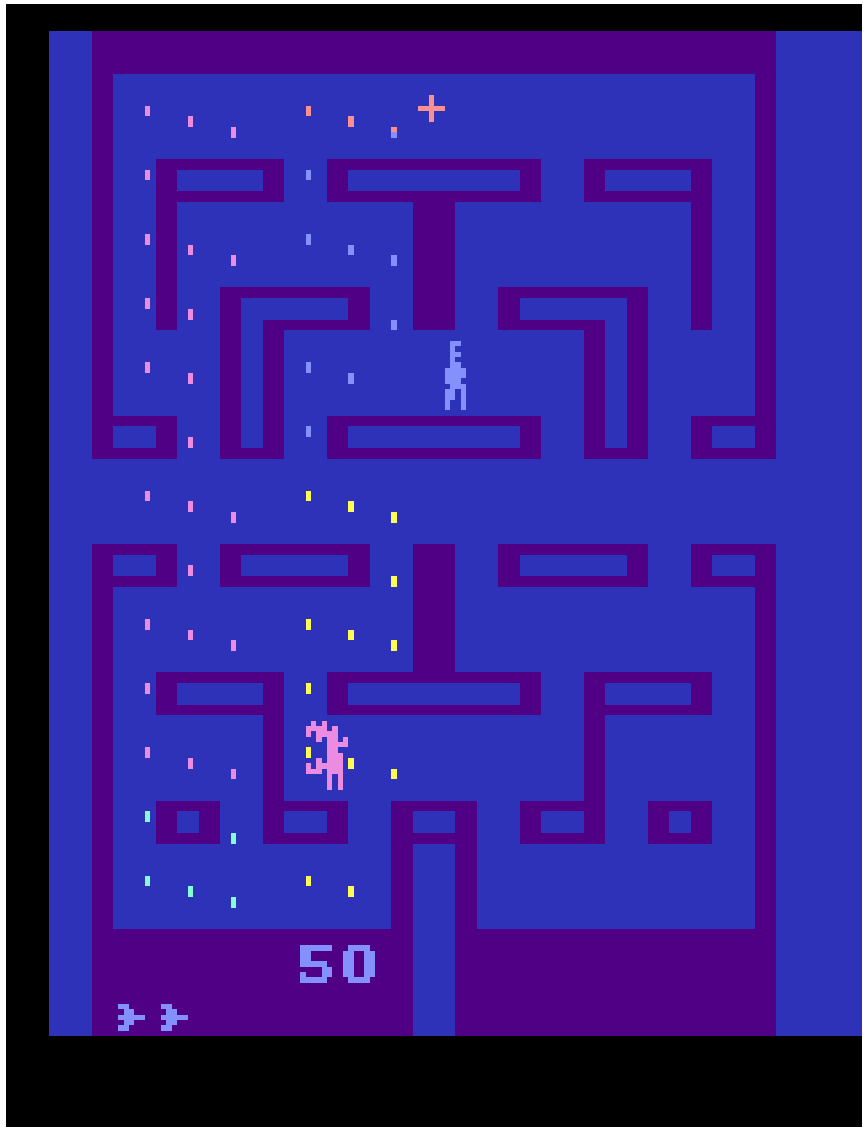


Figure: Atari game: Alien

Standard RL Problems: Continuous Control

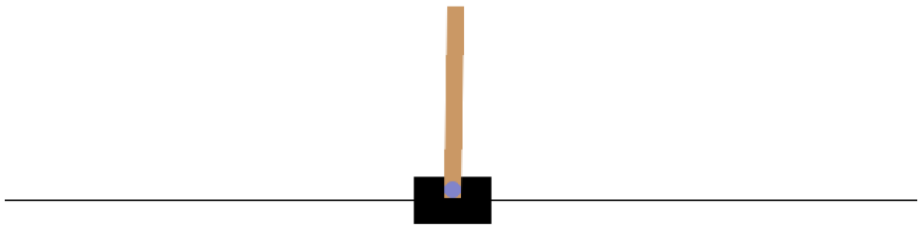


Figure: The Cart Pole environment

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

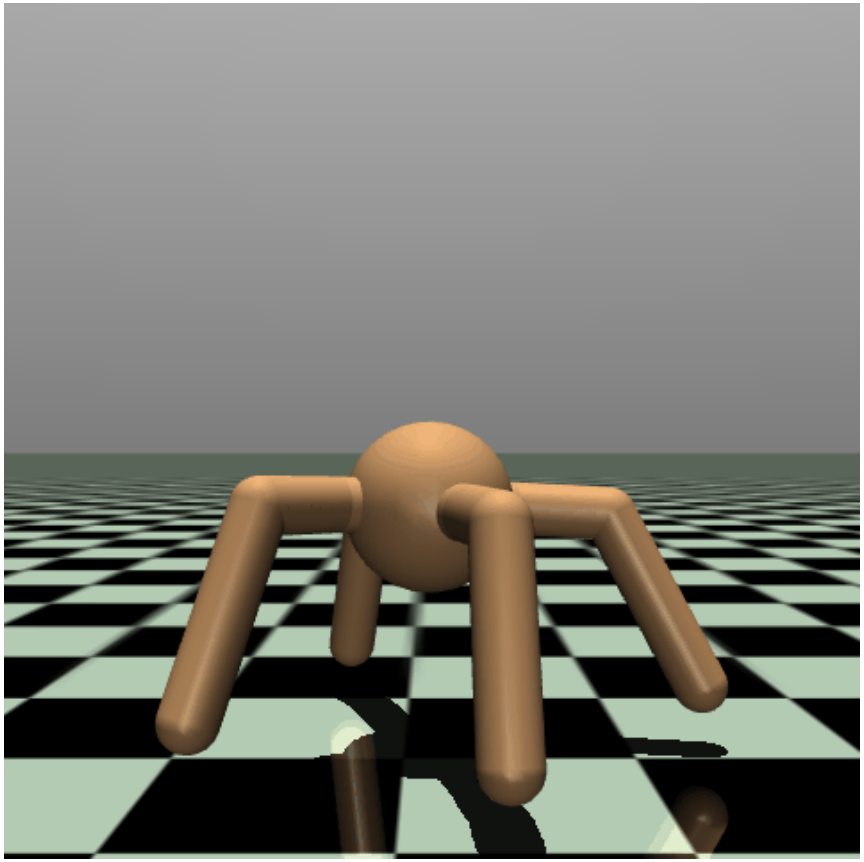


Figure: MuJoCo "Ant" model

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.

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 7 (Constrained MDP)

Constrained MDPs extend *MDPs with rewards* with

- an additional scalar *cost function* $C : S \times A \to \mathbb{R}$
- a separate *discount factor* $\gamma_c \in [0, 1]$.

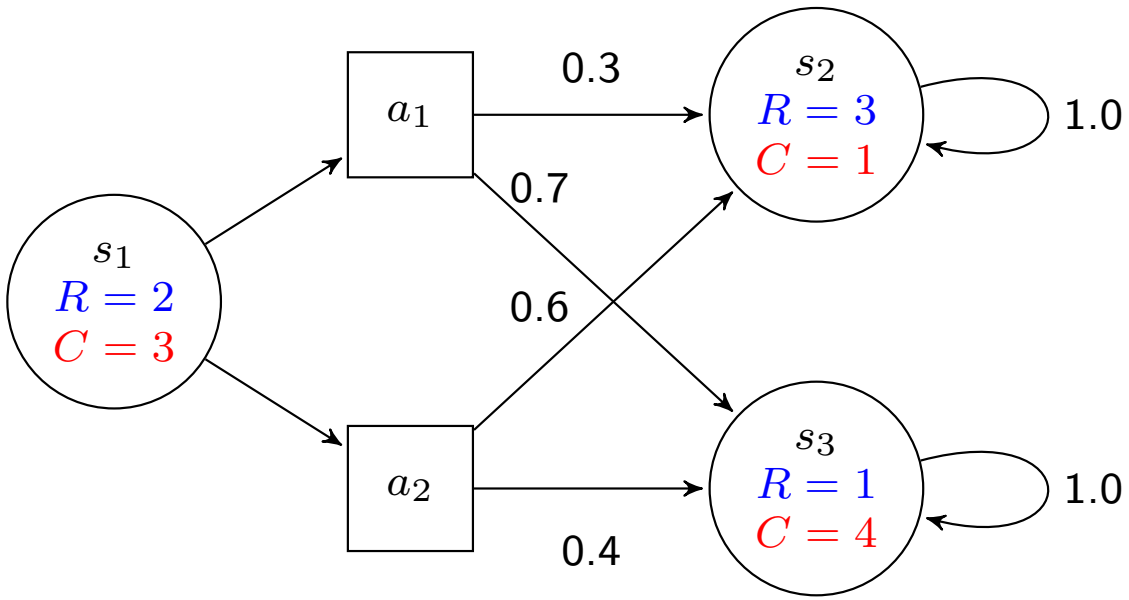
Constrained Reinforcement Learning

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

Definition 7 (Constrained MDP)

Constrained MDPs extend *MDPs with rewards* with

- an additional scalar *cost function* $C : S \times A \to \mathbb{R}$
- a separate *discount factor* $\gamma_c \in [0, 1]$.



Constrained RL: Problem Formulation with CMDPs

Problem Statement

Input: CMDP $\mathcal{M}$ (with costs C and discount 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 $\leq d$:

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

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 vs. Primal-dual (pros and cons)

Primal Methods

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

Primal vs. 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

Primal vs. 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., can be paired with any RL algorithm easily (e.g., off-policy, model-based).

Primal vs. 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., 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.

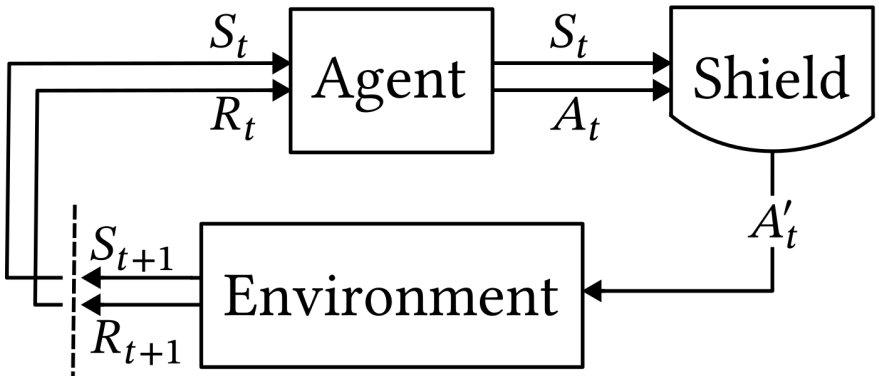


Figure: Post-posed shielding, replaces or modifies the agent's actions A_t with safe alternative A'_t if necessary.

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*.

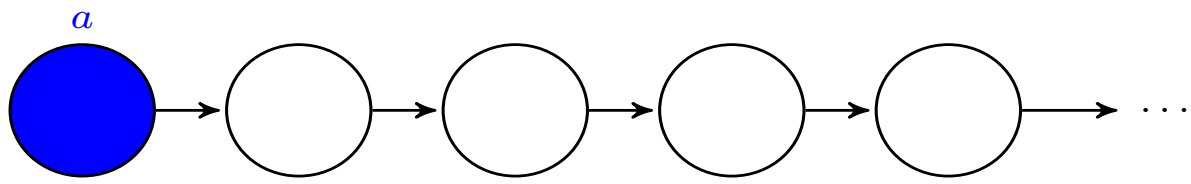
Definition 8 (LTL Syntax)

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

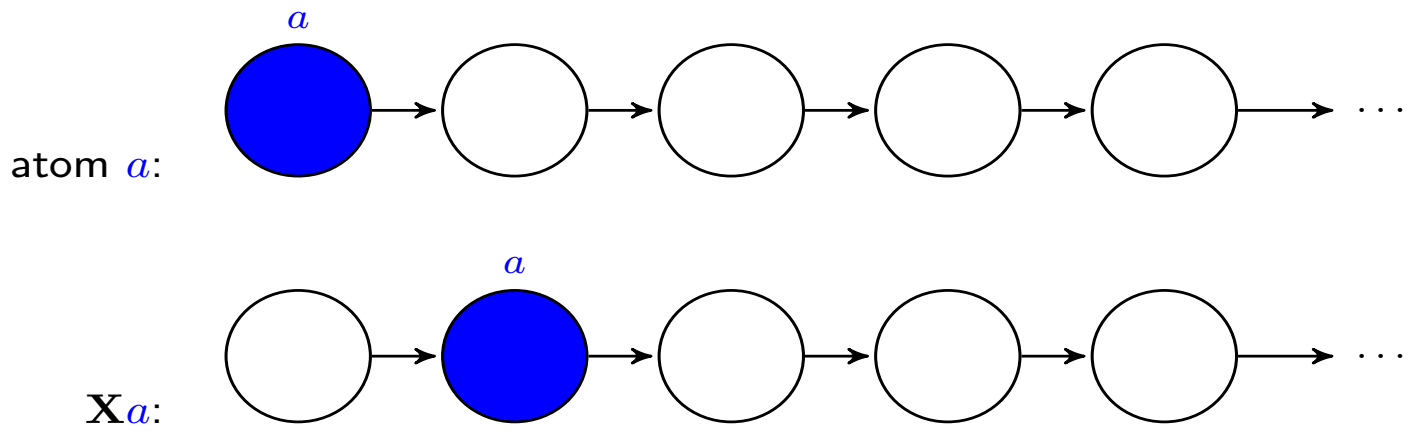
- Examples: $\mathbf{G}(\neg b)$, $a \vee \mathbf{F}b$, $\mathbf{F}(a\mathbf{U}(\neg(\mathbf{X}b)))$
- Suitable to express invariants, (conditional) safety, reachability, reach-avoidance, maintenance, fairness, etc.

Linear Temporal Logic: semantics (intuition)

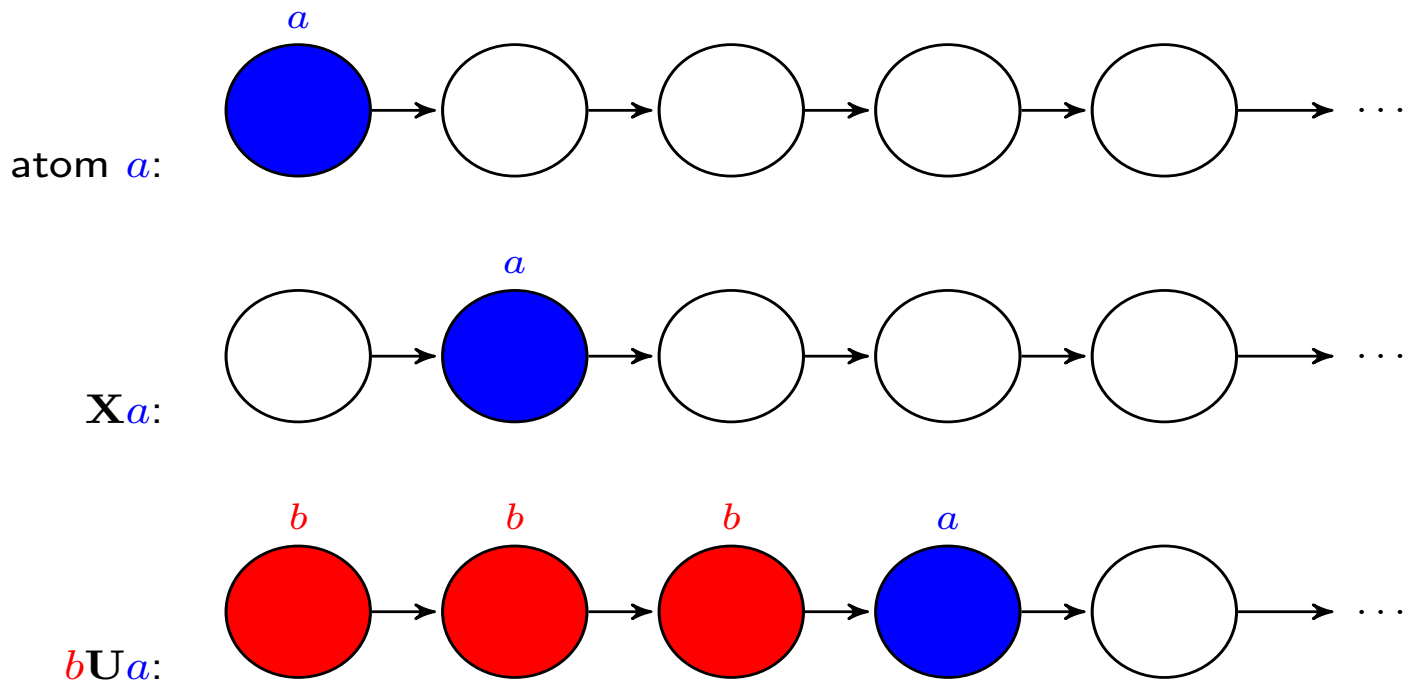
atom a:



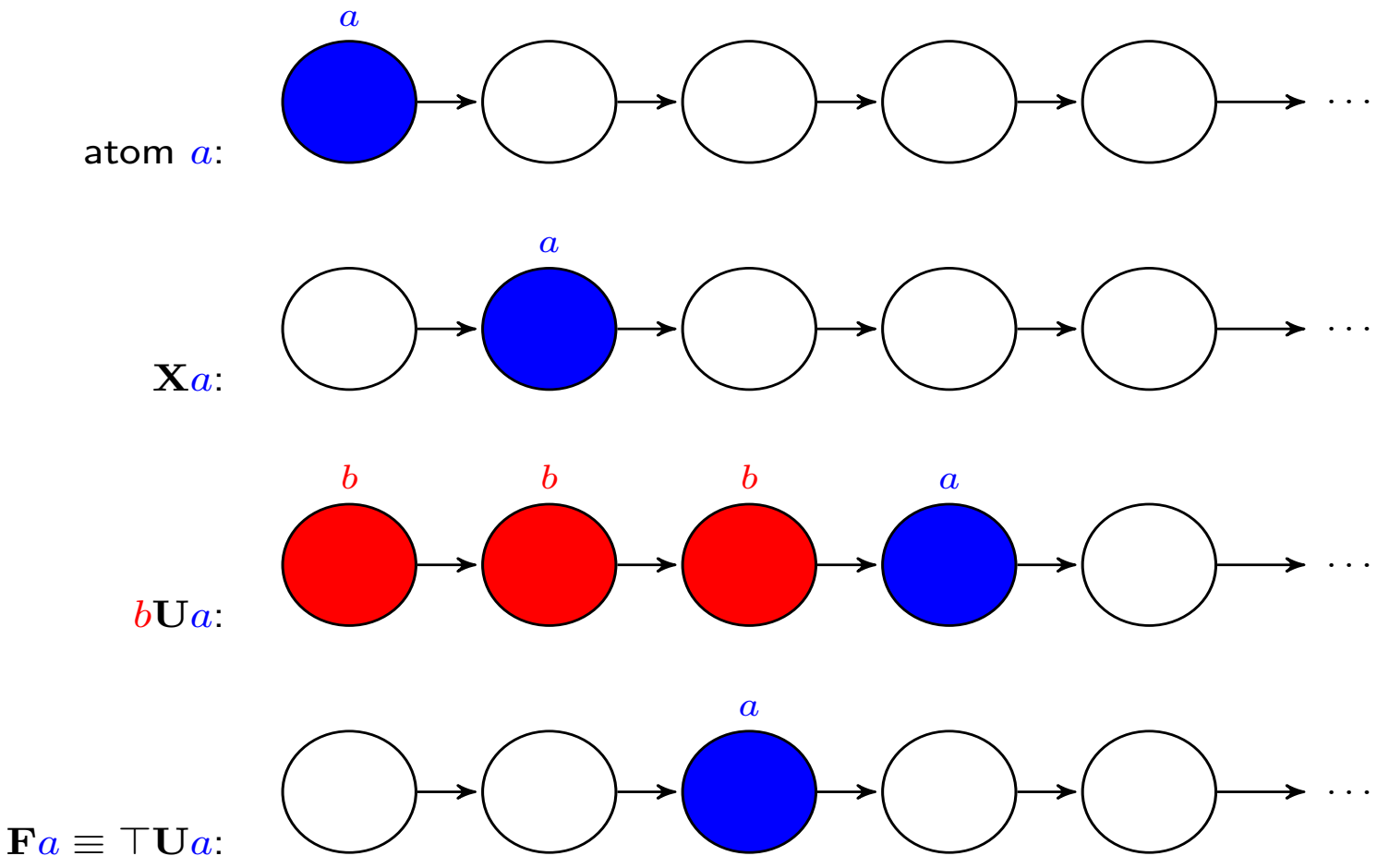
Linear Temporal Logic: semantics (intuition)



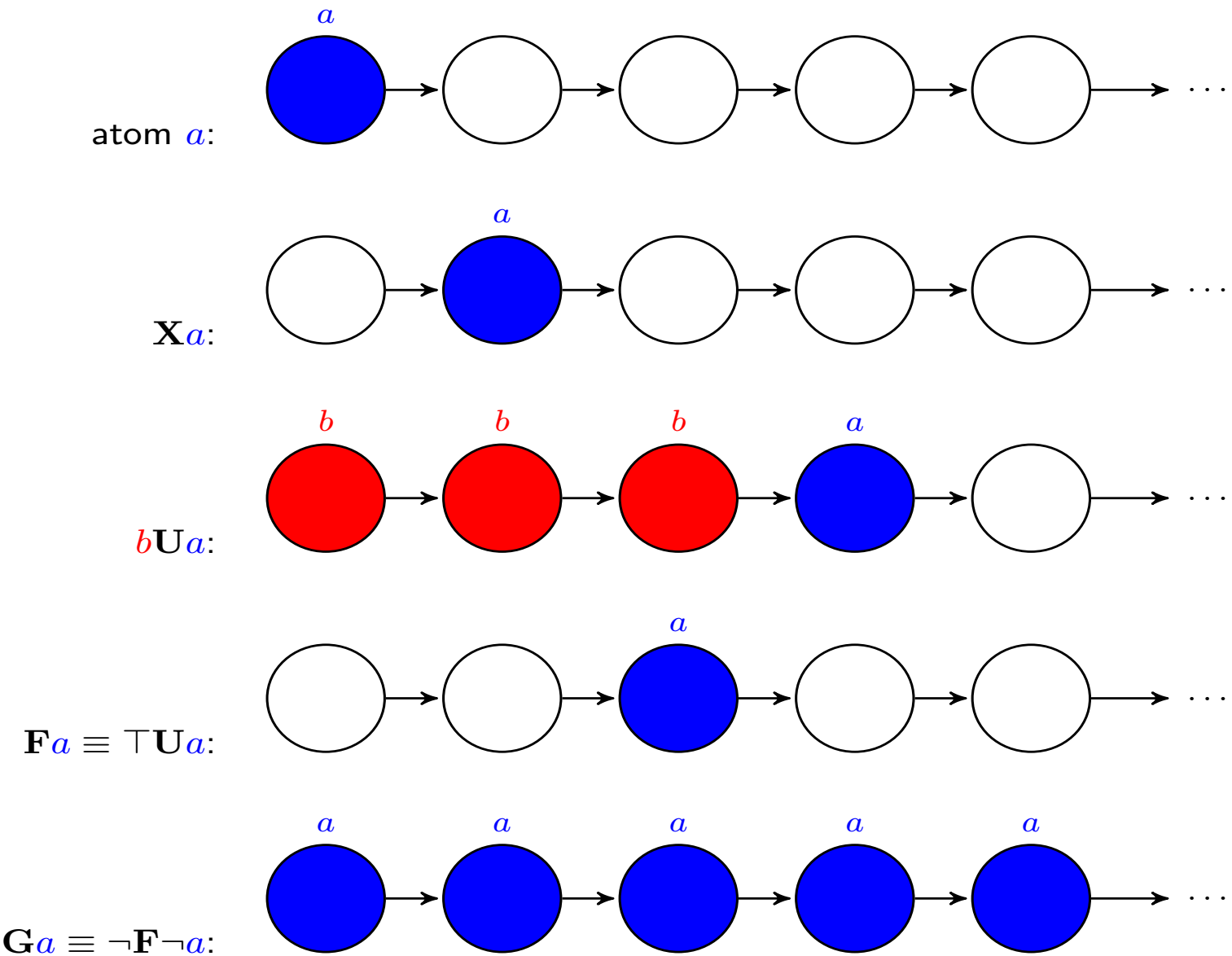
Linear Temporal Logic: semantics (intuition)



Linear Temporal Logic: semantics (intuition)



Linear Temporal Logic: semantics (intuition)



Linear Temporal Logic: Semantics (over Infinite Words)

Definition 9 (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

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*.

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

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)

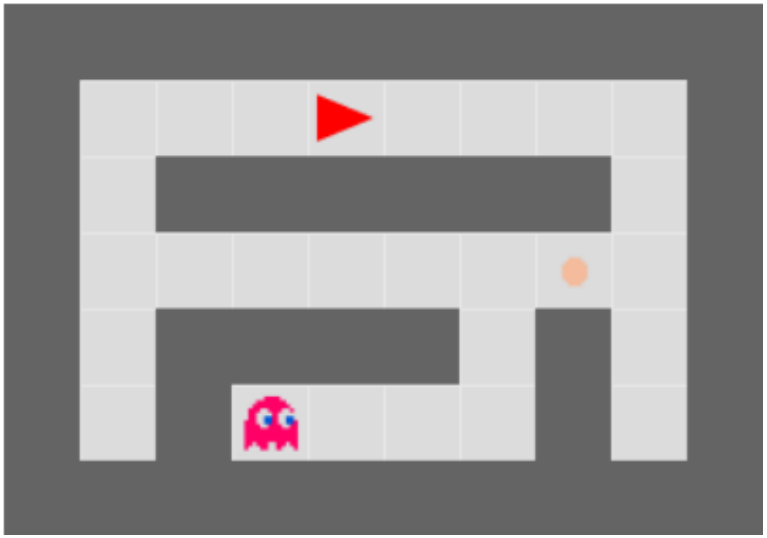


Figure: `MiniPacman` game.

LTL Task Specification

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

Labelled Markov Decision Processes

Definition 10 (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 10 (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 10 (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 10 (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

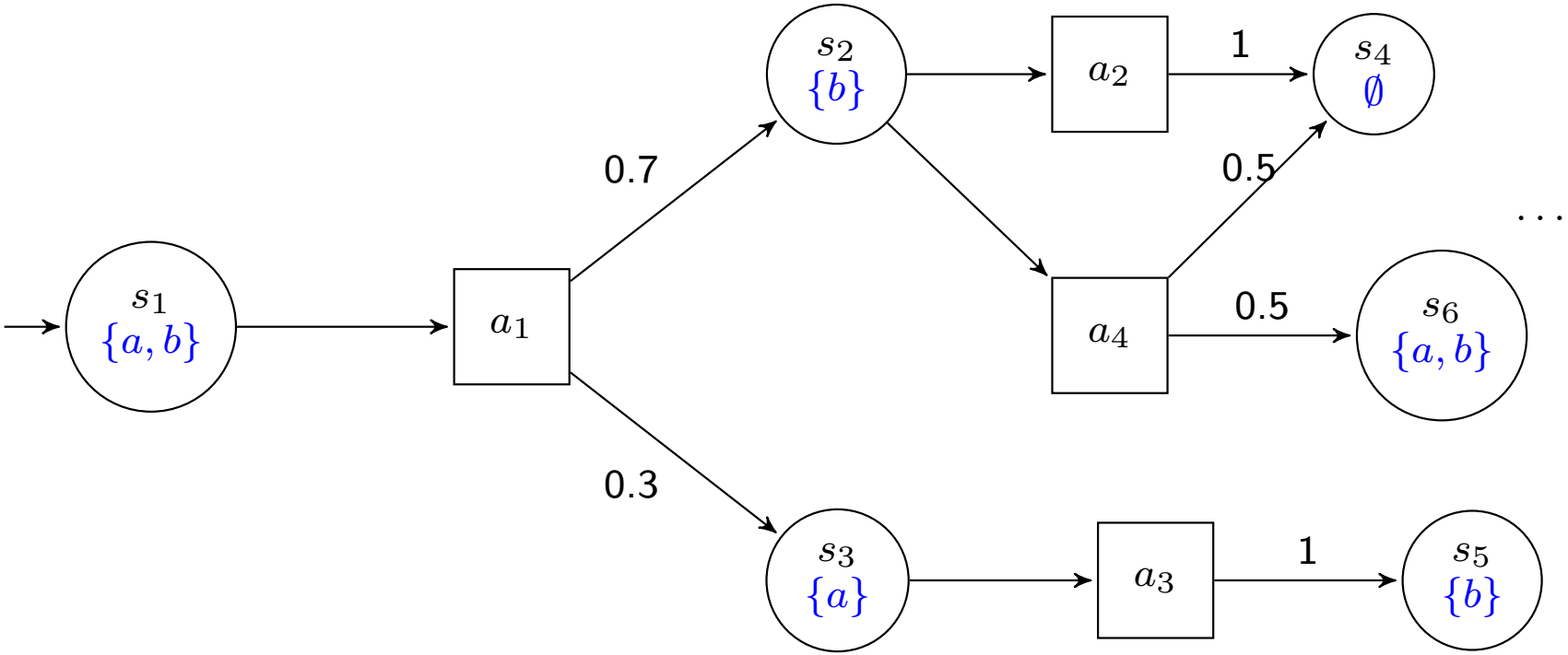
Definition 10 (Labelled MDP)

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

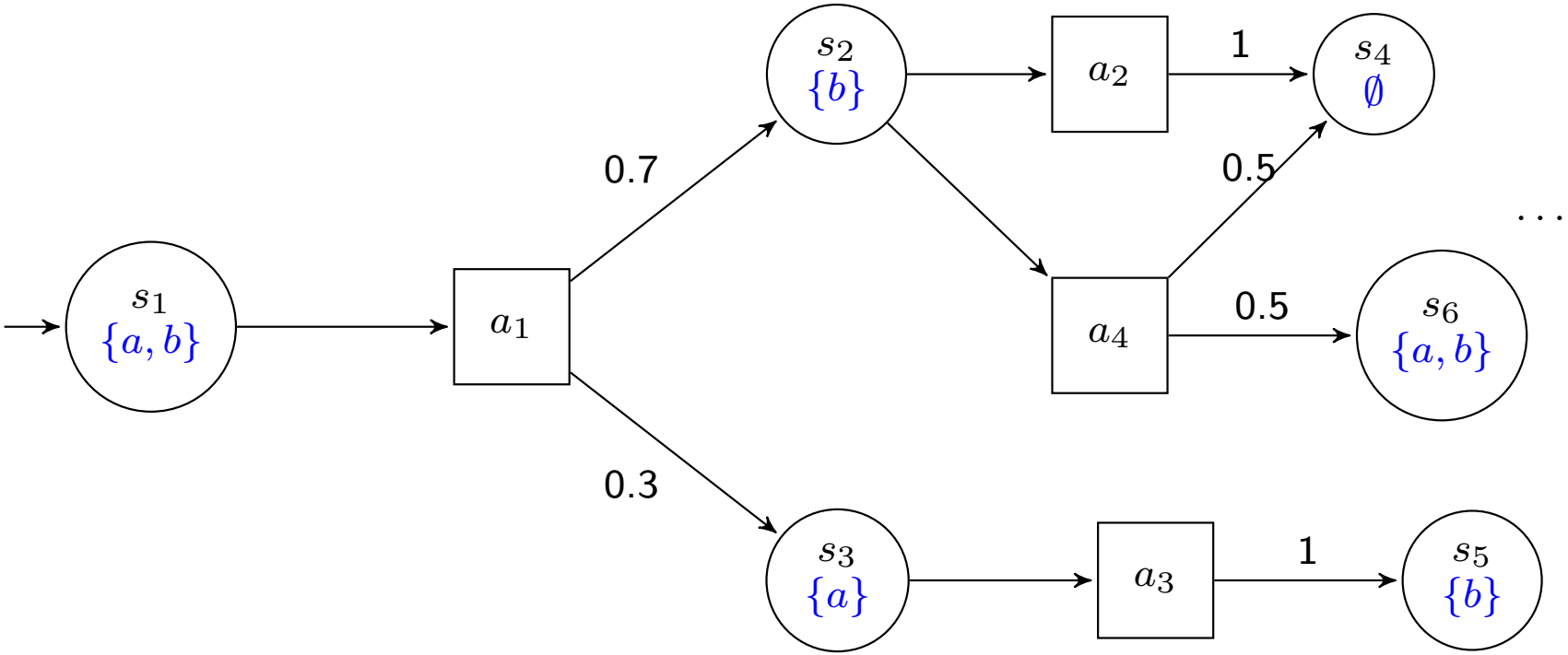
Running example: `MiniPacman` game

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

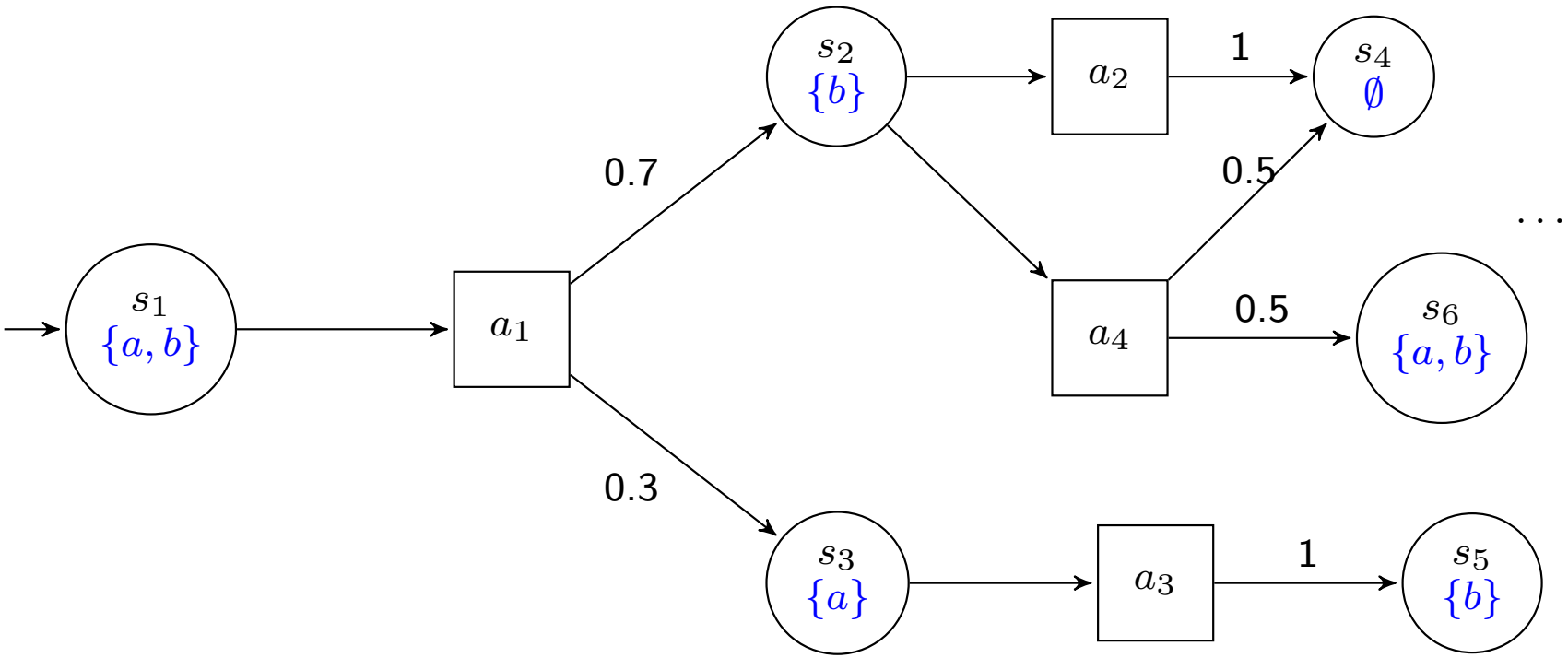


Labelled MDP: Example



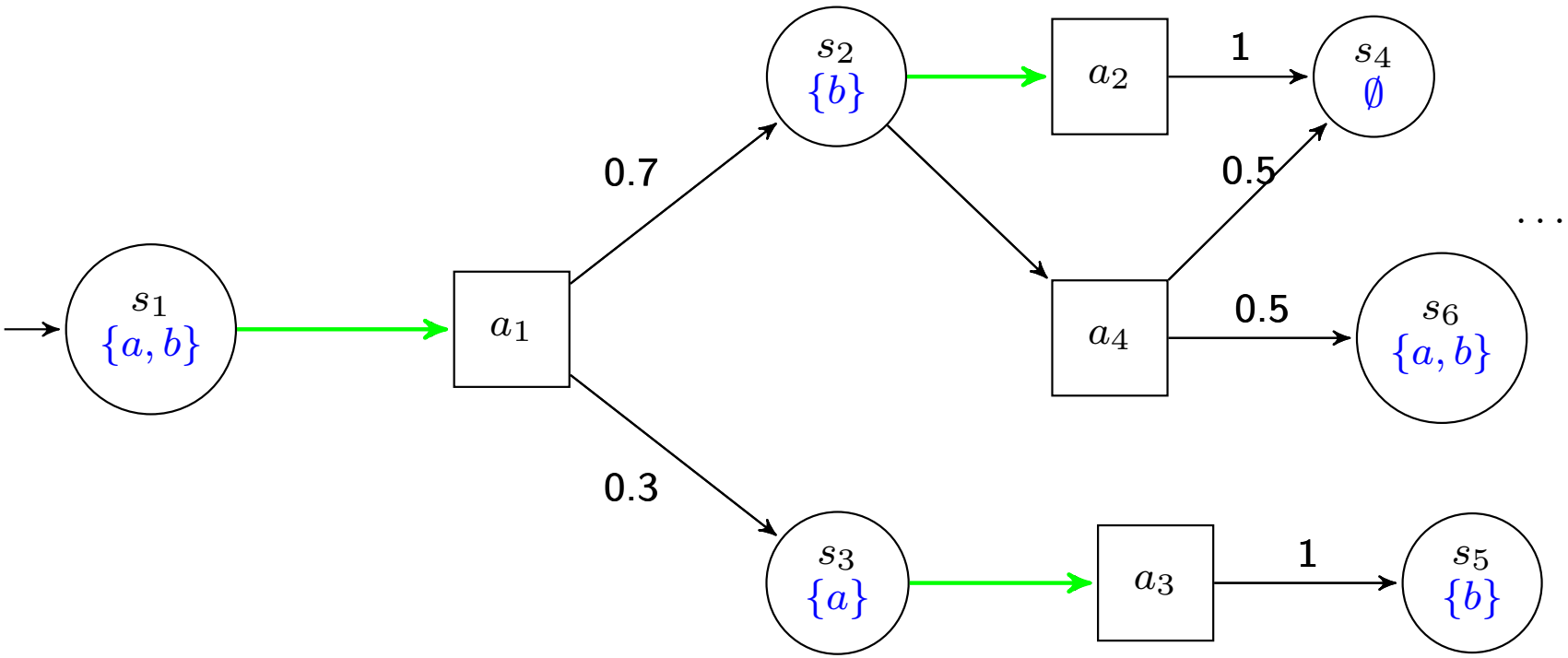
What is the maximum probability that a memoryless policy can satisfy $\varphi = \mathbf{G}(a \rightarrow \mathbf{X}(b))$.

Probability Optimal Policy



A satisfaction-probability-optimal policy π^* achieves $\Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi) = 0.7$.

Probability Optimal Policy



A satisfaction-probability-optimal policy π^* achieves $\Pr^{\pi}_{\mathcal{M}}(\rho \models \varphi) = 0.7$.

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

- **Learning using 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

- **Learning using 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

- **Learning using 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 vs. Liveness

Intuition

- **Safety:** "nothing bad ever happens".
- **Co-safety (e.g., reachability):** "something good eventually happens".

Safety vs. Liveness

Intuition

- **Safety:** "nothing bad ever happens".
- **Co-safety (e.g., reachability):** "something good eventually happens".

Let $\Sigma = 2^{AP}$ and let $P \subseteq \Sigma^\omega$ be a property.
For $w, w' \in \Sigma$, let $w' \preceq w$ mean that "w' is a prefix of word w".

Definition 11 (Bad prefix)

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 vs. Liveness

Intuition

- **Safety:** "nothing bad ever happens".
- **Co-safety (e.g., reachability):** "something good eventually happens".

Let $\Sigma = 2^{AP}$ and let $P \subseteq \Sigma^\omega$ be a property.
For $w, w' \in \Sigma$, let $w' \preceq w$ mean that "w' is a prefix of word w".

Definition 11 (Bad prefix)

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 12 (Safety property)

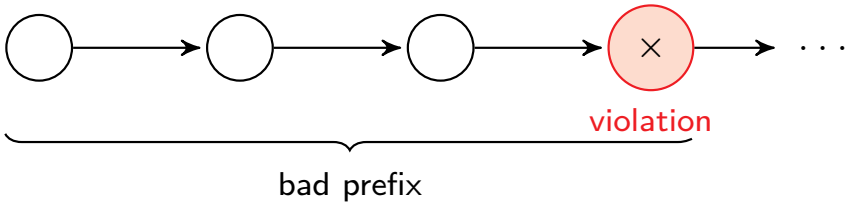
A property $P \subseteq \Sigma^\omega$ is a **safety property** iff for every $w \notin P$ there exists a finite prefix $w_{\text{pref}} \preceq w$ such that w_{pref} is a bad prefix for P.

Safety and Co-safety: Finite Witnesses

Safety: finite witness of failure

A property $P \subseteq \Sigma^\omega$ is a **safety property** if every violating trace has a finite **bad prefix**:

$$w \notin P \implies \exists u \preceq w : \forall v \in \Sigma^\omega,\ uv \notin P.$$



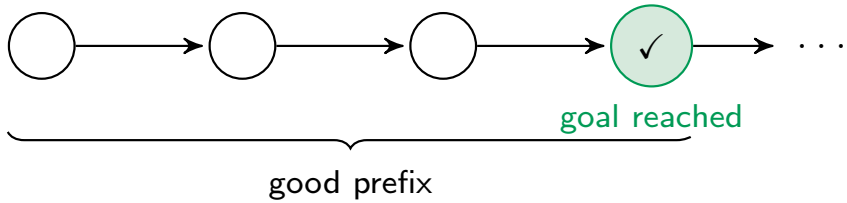
$\mathbf{G}\,\neg\texttt{ghost}$

Once `ghost` occurs, no continuation can repair the violation.

Co-safety: finite witness of success

A property $P \subseteq \Sigma^\omega$ is a **co-safety property** if every satisfying trace has a finite **good prefix**:

$$w \in P \implies \exists u \preceq w : \forall v \in \Sigma^\omega,\ uv \in P.$$



$\mathbf{F}\,\texttt{food}$

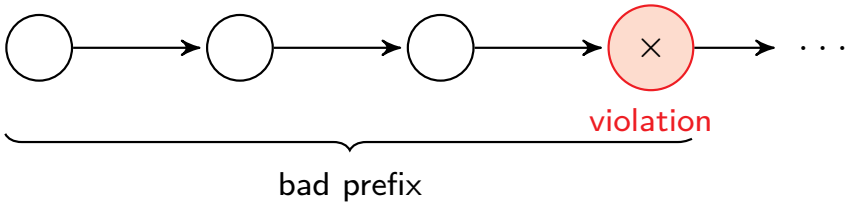
Once `food` occurs, every continuation still satisfies the property.

Safety and Co-safety: Finite Witnesses

Safety: finite witness of failure

A property $P \subseteq \Sigma^\omega$ is a **safety property** if every violating trace has a finite **bad prefix**:

$$w \notin P \implies \exists u \preceq w : \forall v \in \Sigma^\omega,\ uv \notin P.$$



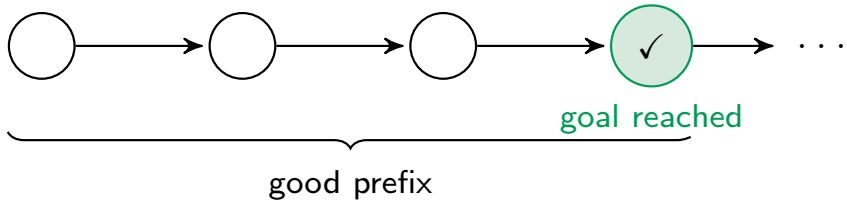
$\mathbf{G}\,\neg\texttt{ghost}$

Once `ghost` occurs, no continuation can repair the violation.

Co-safety: finite witness of success

A property $P \subseteq \Sigma^\omega$ is a **co-safety property** if every satisfying trace has a finite **good prefix**:

$$w \in P \implies \exists u \preceq w : \forall v \in \Sigma^\omega,\ uv \in P.$$



$\mathbf{F}\,\texttt{food}$

Once `food` occurs, every continuation still satisfies the property.

Key distinction

Safety can be falsified after a finite bad prefix; co-safety can be verified with a finite good prefix.

Running Example (Revisited Again...)

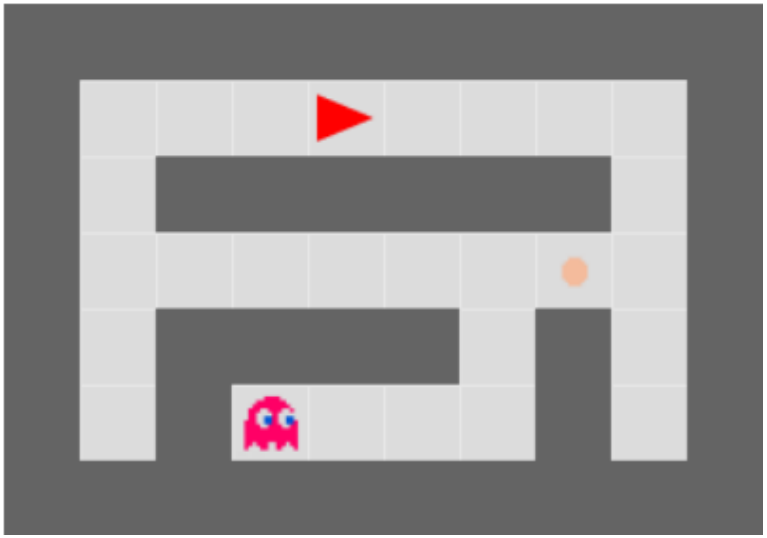


Figure: `MiniPacman` game.

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 13 (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 13 (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 13 (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 13 (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 liveness formula!

The Safety Fragment of LTL: Recap

Definition 13 (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 liveness 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 14 (Monitorability (Kupferman & Vardi, 2001))

*For every **safety** LTL formula φ, there exists a **deterministic finite automaton (DFA)** $\mathcal{D}_\varphi$ that accepts exactly the finite bad prefixes of φ. A finite word is accepted when the DFA run ends in an accepting state.*
Thus, for every $\tau \in \Sigma^\omega$,

$$\tau \models \varphi \quad \textit{iff} \quad \textit{no finite prefix of } \tau \textit{ is accepted by } \mathcal{D}_\varphi.$$

Hence, every violation of φ is detectable after a finite prefix.

The Safety Fragment of LTL: Key Properties

Theorem 14 (Monitorability (Kupferman & Vardi, 2001))

*For every **safety** LTL formula φ, there exists a **deterministic finite automaton (DFA)** $\mathcal{D}_\varphi$ that accepts exactly the finite bad prefixes of φ. A finite word is accepted when the DFA run ends in an accepting state.*
Thus, for every $\tau \in \Sigma^\omega$,

$$\tau \models \varphi \quad \textit{iff} \quad \textit{no finite prefix of } \tau \textit{ is accepted by } \mathcal{D}_\varphi.$$

Hence, every violation of φ is detectable after a finite prefix.

Corollary 15

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

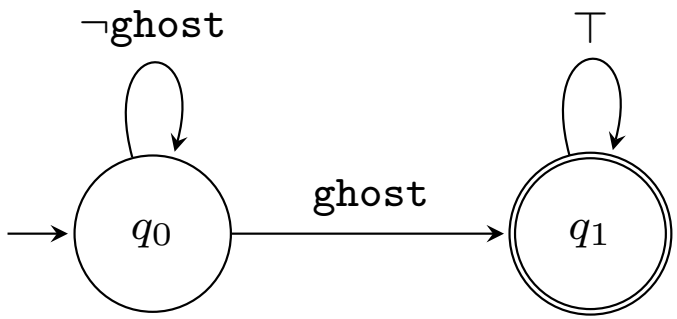
- *We can work with **DFAs** (finite memory).*
- *Synchronizing an MDP with a DFA yields a product MDP where safety reduces to **reachability of a rejecting sink**.*

Deterministic Finite Automata

Definition 16 (DFA)

A **Deterministic Finite Automaton** 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*, with *initial state* $q_0 \in Q$
- $\delta : Q \times \Sigma \to Q$ is a **total** *transition function*,
- $F \subseteq Q$ is the set of accepting states.



DFA: Language

Let $w\downarrow$ denote the last element of a finite word $w \in \Sigma^*$.

Definition 17 (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

Let $w\downarrow$ denote the last element of a finite word $w \in \Sigma^*$.

Definition 17 (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 18 (Language of a DFA)

The language recognized by a DFA $\mathcal{D}$ is:

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

DFA: Language

Let $w\downarrow$ denote the last element of a finite word $w \in \Sigma^*$.

Definition 17 (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 18 (Language of a DFA)

The language recognized by a DFA $\mathcal{D}$ 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:

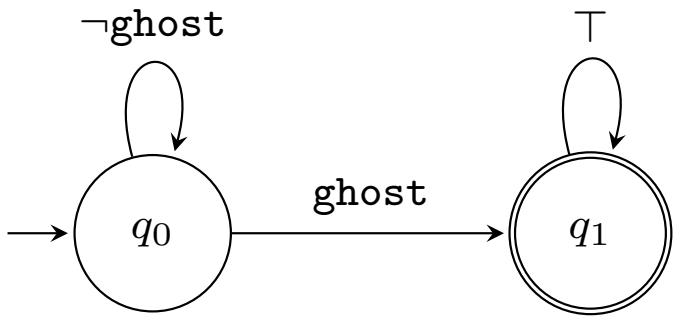
- **bad prefixes** of a safety LTL formula (accepting = "violation detected").

The DFA in the previous slide recognizes the language of words $(\neg\texttt{ghost})^*\texttt{ghost}\top^*$.

DFA: Examples

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

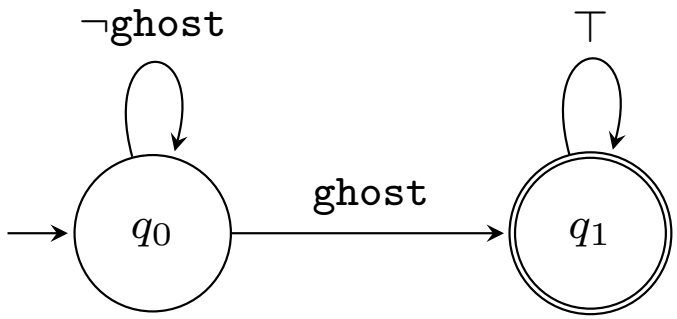
Once ghost is observed, safety is violated forever.



DFA: Examples

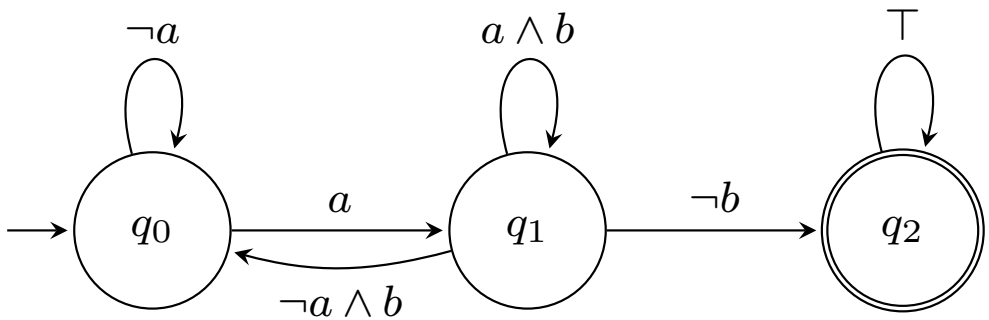
Example 1: Invariant $\mathbf{G}(\neg \texttt{ghost})$

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.



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 $\rho = s_0 a_0 s_1 a_1 \dots$ is $L(\rho) = L(s_0)L(s_1)\dots \in (2^{AP})^\omega$.

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 $\rho = s_0 a_0 s_1 a_1 \dots$ is $L(\rho) = L(s_0)L(s_1)\dots \in (2^{AP})^\omega$.

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 19 (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 19 (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

This construction is adapted from the model checking procedure for LTL formulas.

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

This construction is adapted from the model checking procedure for LTL formulas.

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

This construction is adapted from the model checking procedure for LTL formulas.

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:

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

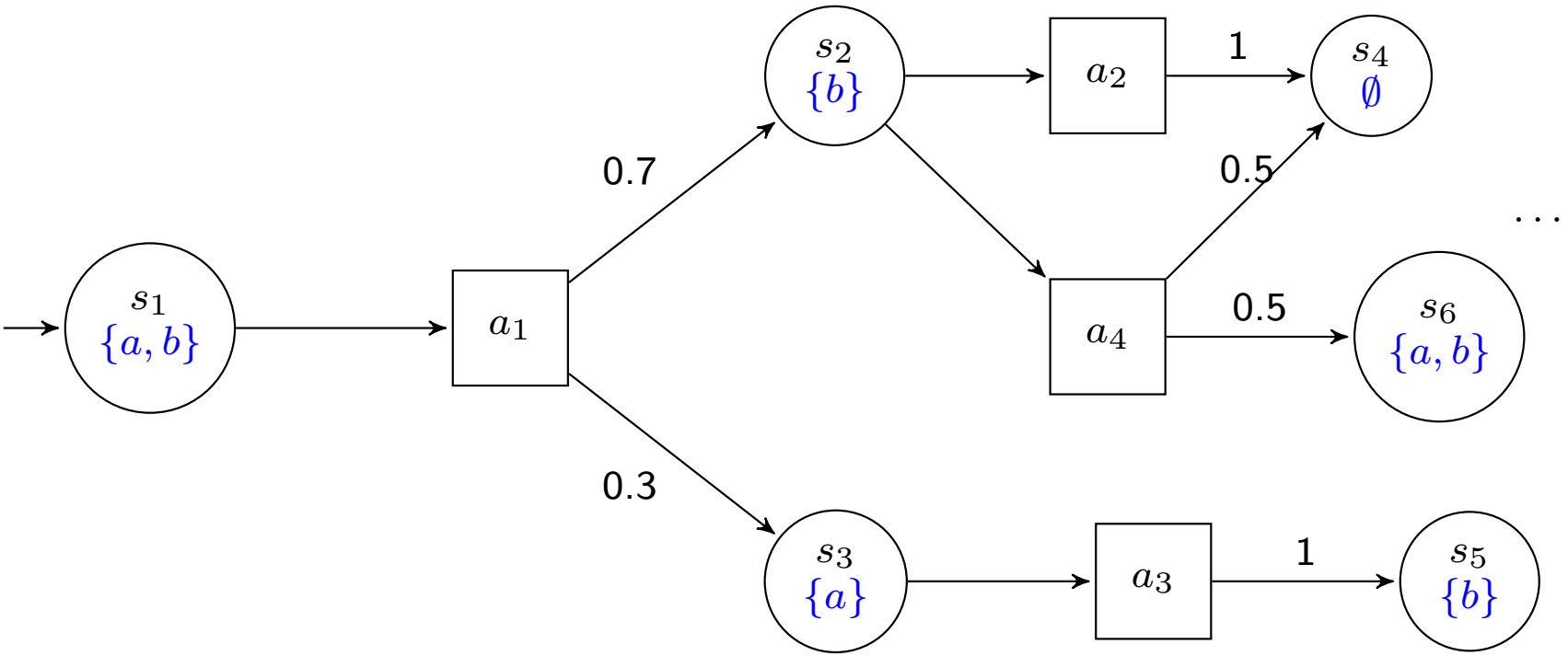
Interpretation

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

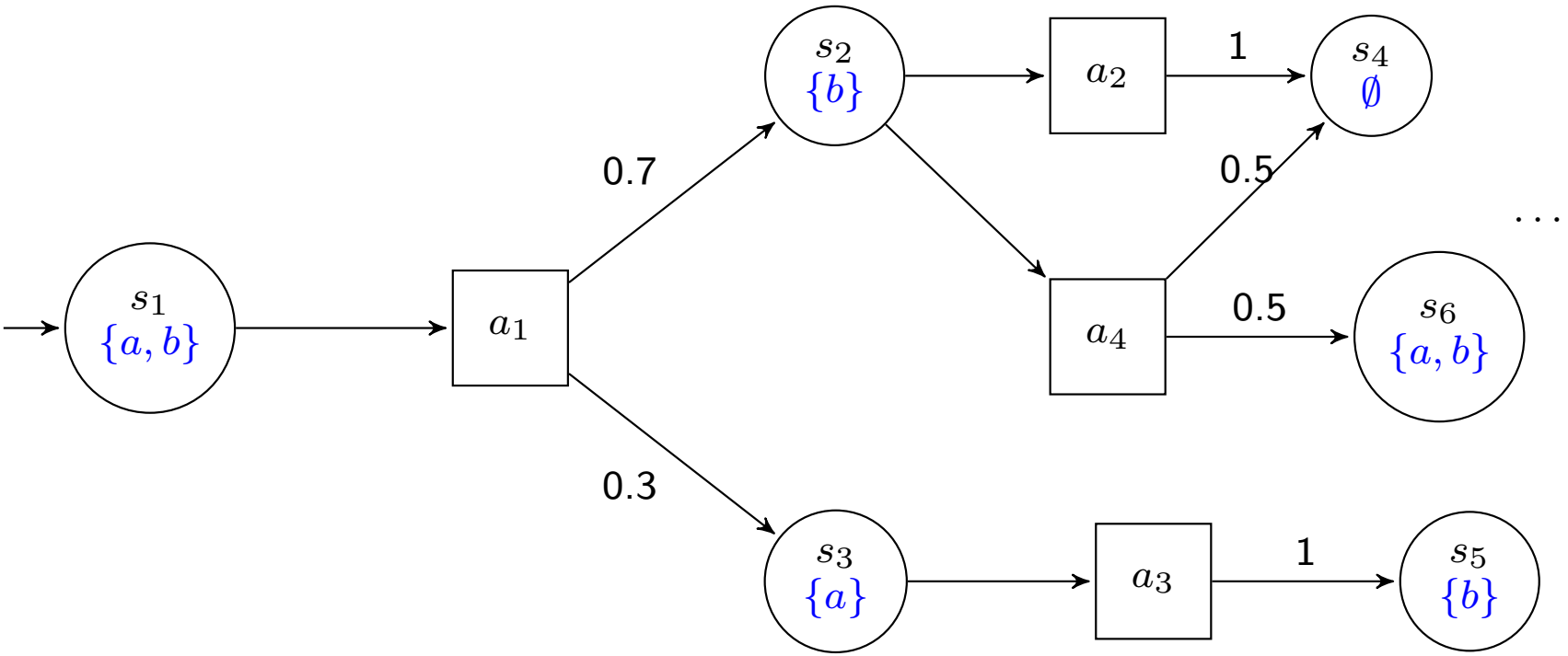
Rejecting sink

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)



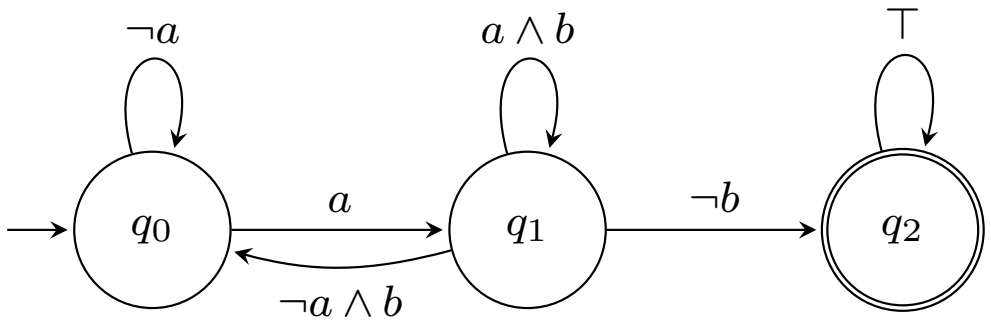
Product MDP: Example (Labelled MDP)



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

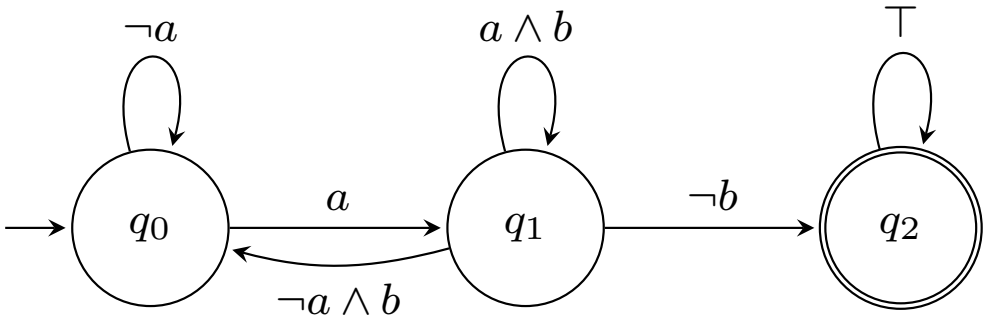
Product MDP: Example (DFA)

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: $\{s_1, s_2, s_3\}$:

$$s_1 \xrightarrow{a_1} \begin{cases} s_2 & \text{w.p. } 0.7 \\ s_3 & \text{w.p. } 0.3 \end{cases} \quad \text{(ignoring all outgoing transitions of } s_2, s_3\text{)}.$$

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

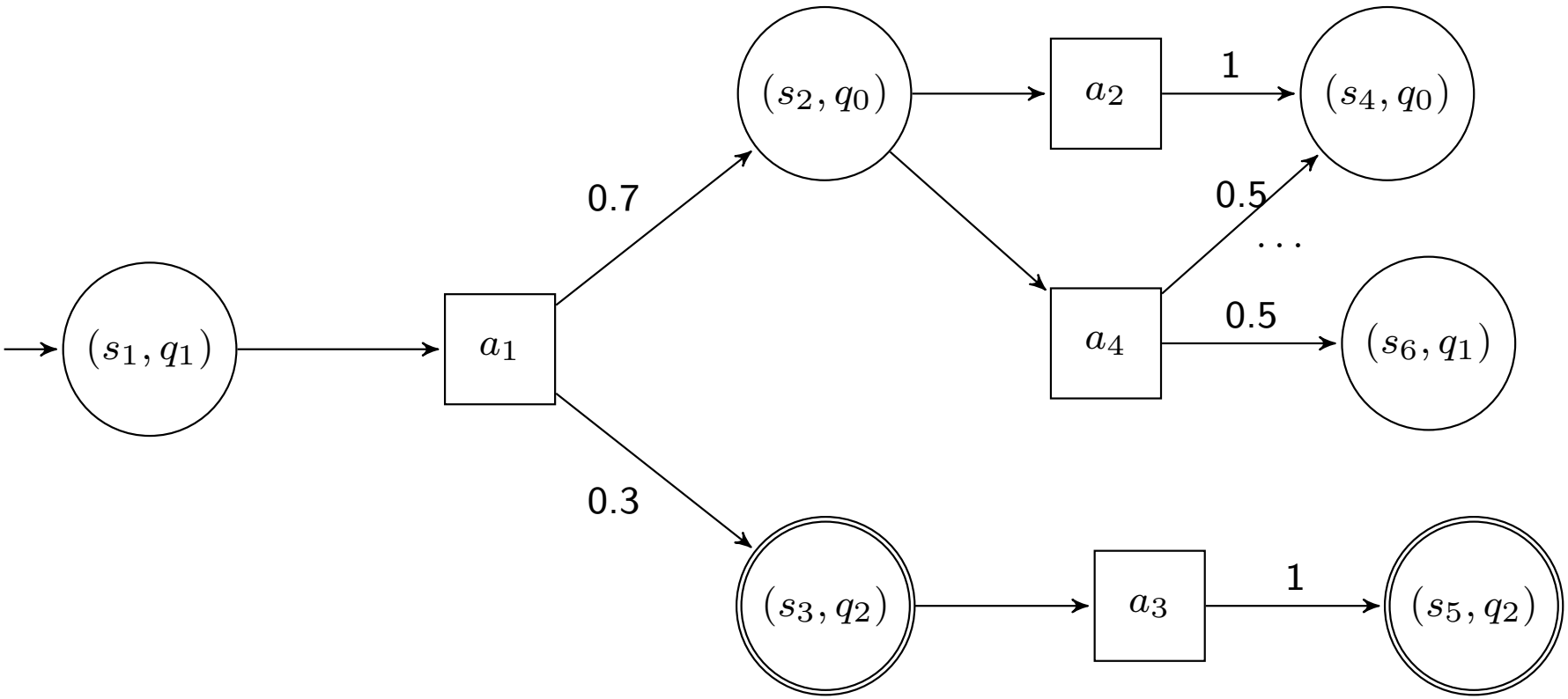
- $L(s_3) = \{a\} \models a \wedge \neg b$
  So,
  $$\delta(q_1, L(s_3)) = q_2,$$

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 set F of states 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** $L^{\otimes}$ 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 set F of states 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** $L^{\otimes}$ such that,

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

Product MDP: Key Theorem

Theorem 20 (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 20 (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 in 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 in A dependent on the current state s only.

Corollary 21 (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 in A dependent on the current state s only.

Corollary 21 (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

Why Compute a Winning Region?

The question

Given the product MDP, from which states can the agent **guarantee that the safety violation is never reached**?

Why Compute a Winning Region?

The question

Given the product MDP, from which states can the agent **guarantee that the safety violation is never reached**?

Why do we need it?

- A state is **winning** if the agent still has some way to remain safe forever.
- Once we know the winning states, we can identify the **winning actions**: actions that keep every possible successor inside the winning region.
- A shield can then restrict the RL agent to these actions, independently of how the agent optimizes its reward.

Why Compute a Winning Region?

The question

Given the product MDP, from which states can the agent **guarantee that the safety violation is never reached**?

Why do we need it?

- A state is **winning** if the agent still has some way to remain safe forever.
- Once we know the winning states, we can identify the **winning actions**: actions that keep every possible successor inside the winning region.
- A shield can then restrict the RL agent to these actions, independently of how the agent optimizes its reward.

High-level idea

1. Start with all non-violating product states.
2. Remove any state from which *every* available action may leave the current safe candidate set.
3. Repeat until no more states can be removed.

The states that remain form the **winning region** W. It may be empty.

Almost-Sure Winning Region

Definition 22 (Almost-sure Winning Region)

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

$$\mathrm{W} := \left\{ s \in S^{\otimes} \;\middle|\; \exists \pi^{\otimes} : \Pr_{\mathcal{M}\otimes\mathcal{D}}^{\pi^{\otimes}}\left(\neg \mathbf{F}\, \mathtt{accept} \mid s\right) = 1 \right\}.$$

Almost-Sure Winning Region

Definition 22 (Almost-sure Winning Region)

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

$$\mathrm{W} := \left\{ s \in S^{\otimes} \;\middle|\; \exists \pi^{\otimes} : \Pr_{\mathcal{M}\otimes\mathcal{D}}^{\pi^{\otimes}}\left(\neg \mathbf{F}\,\mathsf{accept} \mid s\right) = 1 \right\}.$$

Definition 23 (Winning Actions)

For every $s \in \mathrm{W}$, define

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

These are exactly the actions that preserve almost-sure safety from s.

Almost-Sure Winning Region

Definition 22 (Almost-sure Winning Region)

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

$$\mathsf{W} := \left\{ s \in S^{\otimes} \;\middle|\; \exists \pi^{\otimes} : \Pr_{\mathcal{M}\otimes\mathcal{D}}^{\pi^{\otimes}}\left(\neg \mathbf{F}\,\mathsf{accept} \mid s\right) = 1 \right\}.$$

Definition 23 (Winning Actions)

For every $s \in \mathsf{W}$, define

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

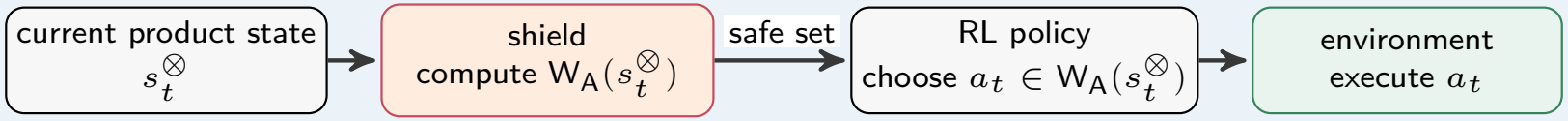
These are exactly the actions that preserve almost-sure safety from s.

Intuition

- If $s \in \mathsf{W}$, then $\mathsf{W}_{\mathsf{A}}(s) \neq \emptyset$.
- If $s \notin \mathsf{W}$, every policy has non-zero probability of eventually reaching $F^{\otimes}$.

Shielding Architectures: Pre-emptive vs Post-posed

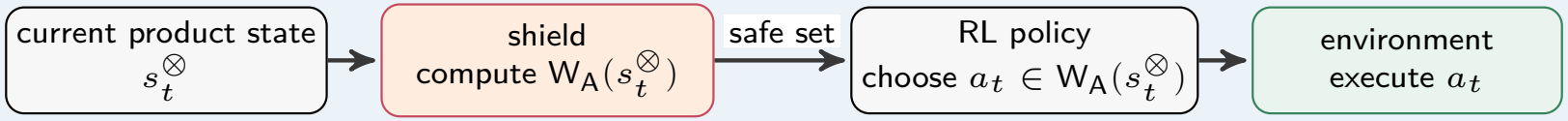
Pre-emptive shielding



The shield filters the action set *before* the policy chooses, so unsafe actions are never available to the policy.

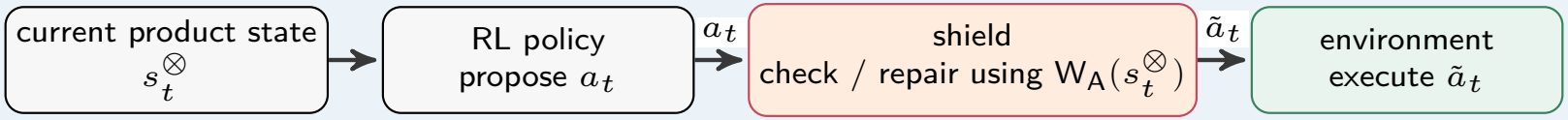
Shielding Architectures: Pre-emptive vs Post-posed

Pre-emptive shielding



The shield filters the action set *before* the policy chooses, so unsafe actions are never available to the policy.

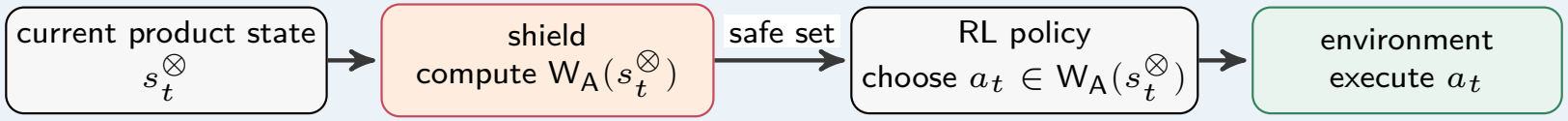
Post-posed shielding



The policy proposes first; the shield passes safe proposals through and replaces unsafe ones with a safe alternative $\tilde{a}_t \in \mathsf{W}_\mathsf{A}(s_t^{\otimes})$.

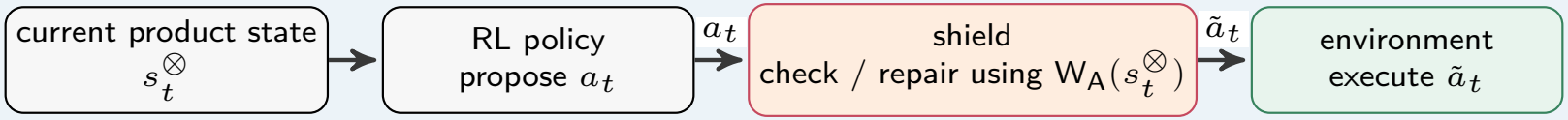
Shielding Architectures: Pre-emptive vs Post-posed

Pre-emptive shielding



The shield filters the action set *before* the policy chooses, so unsafe actions are never available to the policy.

Post-posed shielding



The policy proposes first; the shield passes safe proposals through and replaces unsafe ones with a safe alternative $\tilde{a}_t \in \mathsf{W}_\mathsf{A}(s_t^{\otimes})$.

In common: both use $\mathsf{W}_\mathsf{A}(s_t^{\otimes})$. Compute W by fixpoint, then derive W_A.

Computing Winning Regions: Overview

Goal

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

- the **almost-sure winning region** W,
- the ϵ-**winning regions** W_ϵ.

Computing Winning Regions: Overview

Goal

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

- the **almost-sure winning region** W,
- the ϵ-**winning regions** W_ϵ.

Almost-sure winning region W (qualitative)

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

Computing Winning Regions: Overview

Goal

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

- the **almost-sure winning region** W,
- the ϵ-**winning regions** W_ϵ.

Almost-sure winning region W (qualitative)

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

$$s \text{ is removed if } \forall a \in A^{\otimes}(s),\ \mathrm{supp}\Big(P^{\otimes}(\cdot \mid s, a)\Big) \not\subseteq X.$$

Computing Winning Regions: Overview

Goal

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

- the **almost-sure winning region** W,
- the ϵ-**winning regions** W_ϵ.

Almost-sure winning region W (qualitative)

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

$$s \text{ is removed if } \forall a \in A^{\otimes}(s),\ \mathrm{supp}\Big(P^{\otimes}(\cdot \mid s, a)\Big) \not\subseteq X.$$

- The fixpoint W is the largest set of states from which safety can be enforced with probability 1. Notice that W can be empty.

Computing Winning Regions: Advanced Ideas

Safety Abstractions

Build a safety abstraction $\text{Abs}(\mathcal{M})$ that retains only the labels and dynamics relevant to safety.
Safety-equivalent states can then be merged, reducing the model before reachability analysis.

Probabilistic Shielding: ϵ-Winning Region (known transition probabilities)

With known transition probabilities, W_ϵ and the corresponding action sets $\mathrm{W}_{\mathrm{A},\epsilon}(s)$ can be computed from quantitative reachability probabilities, e.g., by interval-bound value iteration or linear programming.

Computing Winning Regions: Advanced Ideas

Safety Abstractions

Build a safety abstraction $\text{Abs}(\mathcal{M})$ that retains only the labels and dynamics relevant to safety.
Safety-equivalent states can then be merged, reducing the model before reachability analysis.

Probabilistic Shielding: ϵ-Winning Region (known transition probabilities)

With known transition probabilities, W_ϵ and the corresponding action sets $\mathsf{W}_{\mathsf{A},\epsilon}(s)$ can be computed from quantitative reachability probabilities, e.g., by interval-bound value iteration or linear programming.

Probabilistic Shielding: unknown transition probabilities

When transition probabilities are unknown, interval or robust MDPs can provide conservative estimates of W_ϵ and $\mathsf{W}_{\mathsf{A},\epsilon}(s)$ via worst-case reachability analysis.

Remark: unlike the almost-sure case, $\mathsf{W}_{\mathsf{A},\epsilon}(s)$ is determined by a quantitative risk bound, not merely by support containment.

Safety-relevant abstraction: Example

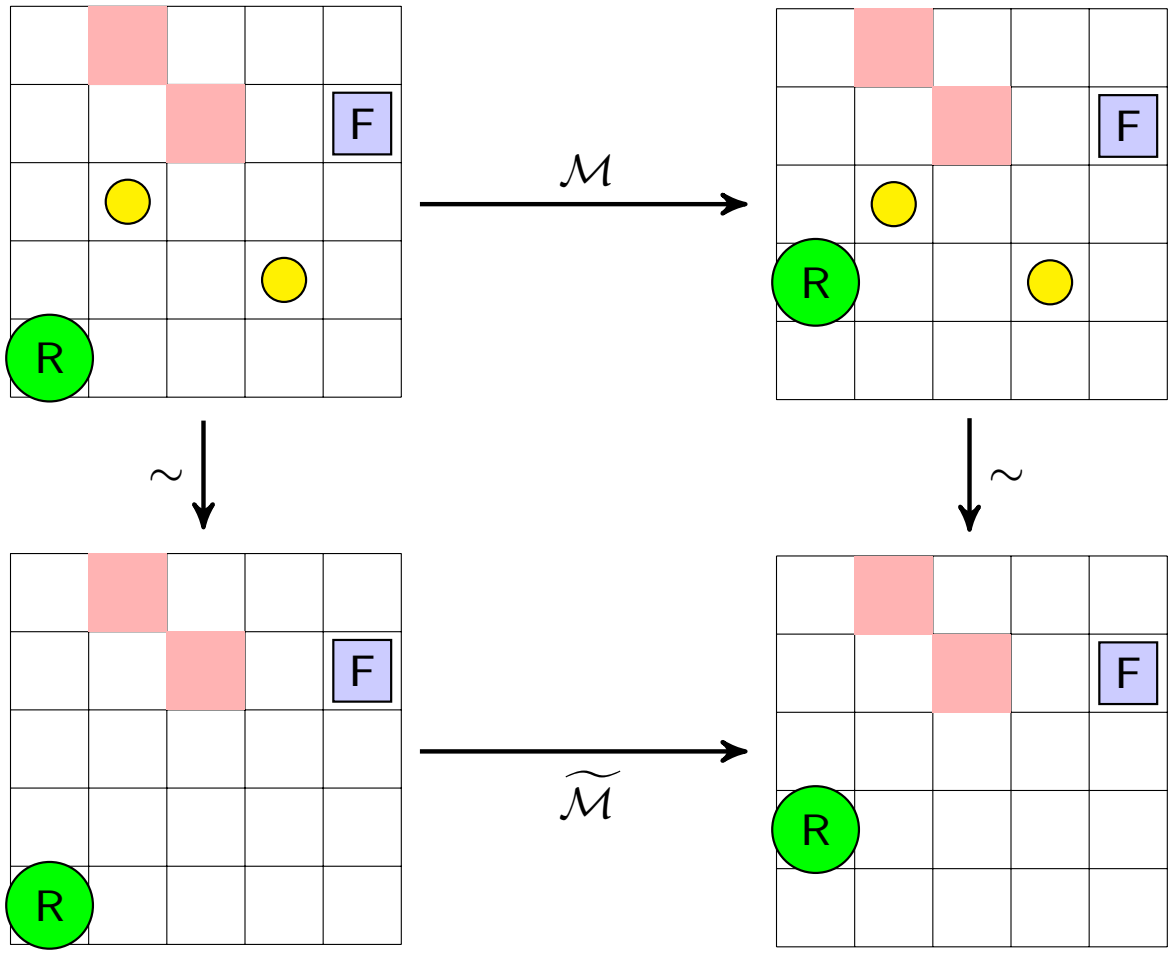


Figure: Illustration of the safety-relevant abstraction: $\text{Abs}(\mathcal{M})$ is the "same" MDP as $\mathcal{M}$ but without coins.

Demo: PacmanWithCoins: Safety Abstraction

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

Greatest Fixed Point & Winning Region Computation

Controllable predecessor

For $X \subseteq S^{\otimes}$, define

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

These are the states from which some action keeps the next state in X with probability 1.

Greatest Fixed Point & Winning Region Computation

Controllable predecessor

For $X \subseteq S^{\otimes}$, define

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

These are the states from which some action keeps the next state in X with probability 1.

Greatest fixed point

The almost-sure winning region is

$$\mathsf{W} = \nu X. \Big((S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}(X) \Big).$$

Intuitively, W is the largest subset of safe product states from which the agent can always choose an action whose possible successors remain inside W.

Greatest Fixed Point Algorithm

Fixed-point operator

Define

$$f(X) = (S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}(X).$$

Then

$$\mathsf{W} = \nu X.\, f(X).$$

Greatest Fixed Point Algorithm

Fixed-point operator

Define

$$f(X) = (S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}(X).$$

Then

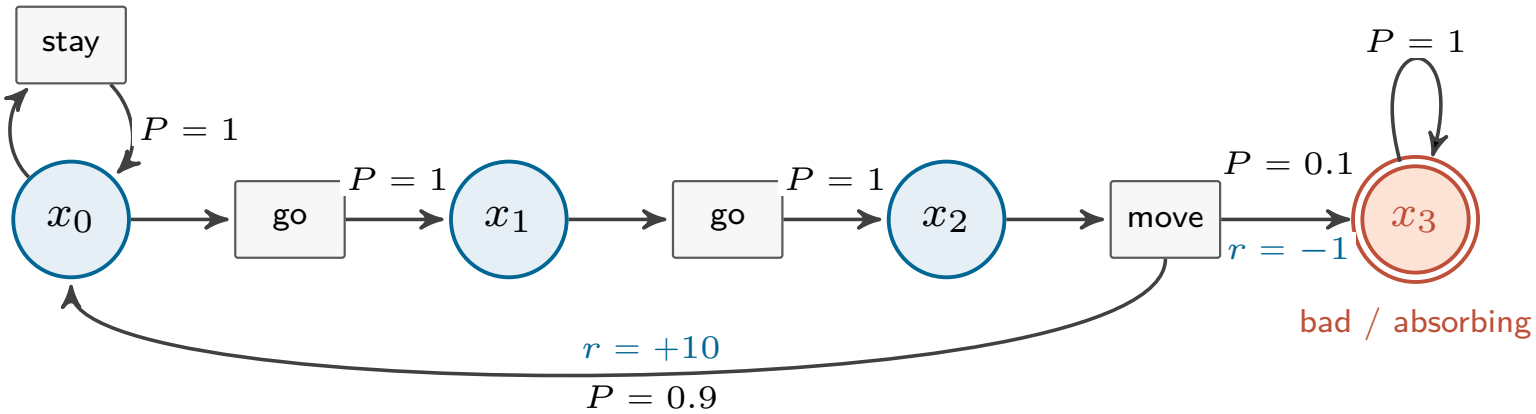
$$\mathsf{W} = \nu X.\, f(X).$$

Algorithm 1 Computing the Almost-Sure Winning Region

1: $X_{\text{old}} \leftarrow S^{\otimes} \setminus F^{\otimes}$
2: $X \leftarrow f(X_{\text{old}})$
3: **while** $X \neq X_{\text{old}}$ **do**
4: $\quad X_{\text{old}} \leftarrow X$
5: $\quad X \leftarrow f(X)$
6: **end while**
7: **return** $\mathsf{W} \leftarrow X$

Winning Region: A Four-State Example

Let $S^{\otimes} = \{x_0, x_1, x_2, x_3\}$ and $F^{\otimes} = \{x_3\}$.



Circles are **product states**; boxes are **actions**. Edge labels show transition probability P and illustrative transition reward r; x_3 is the bad absorbing state.

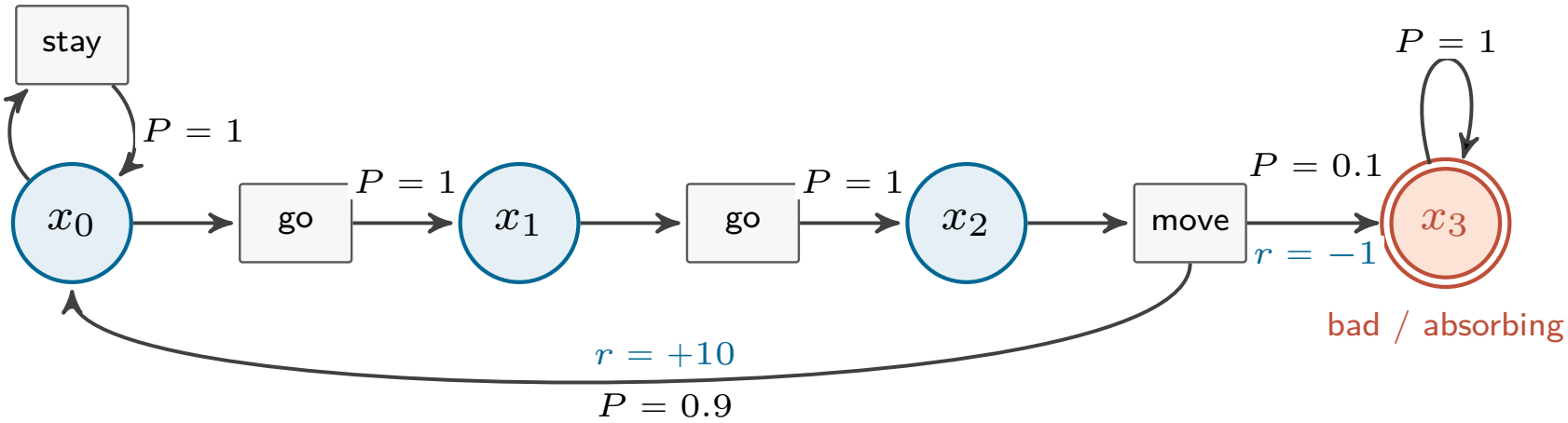
Question

From which states can the agent avoid x_3 **forever with probability** 1?

At x_2, move is tempting in reward terms (0.9 chance of $+10$), but it still has non-zero probability of entering the bad state.

Winning Region: Step-by-Step Computation

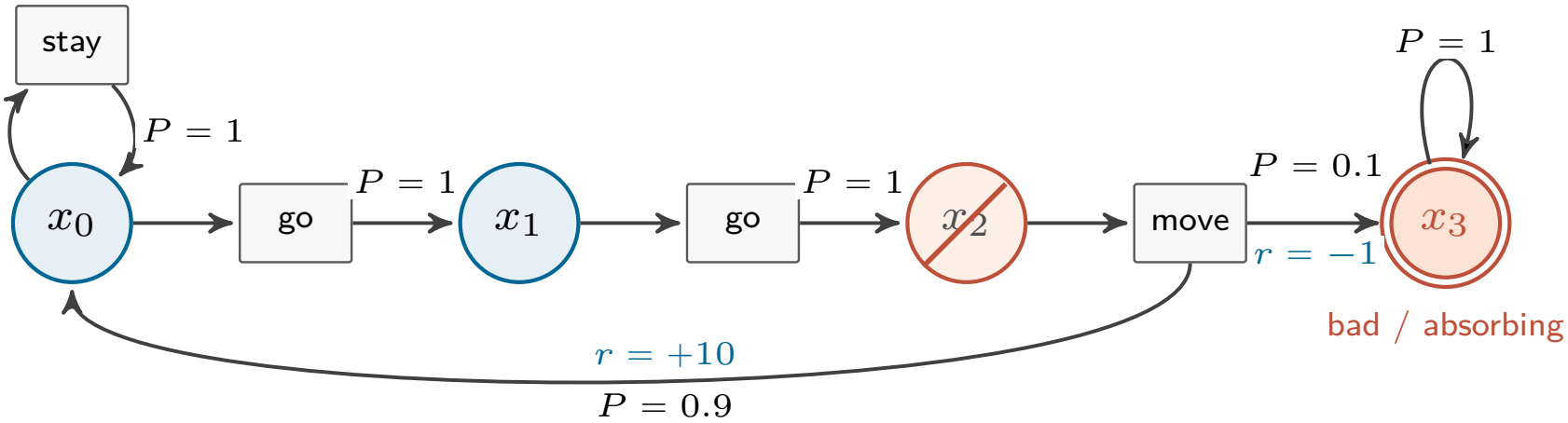
$X_0 = S^{\otimes} \setminus F^{\otimes}, \qquad X_{k+1} = (S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}(X_k).$



$X_0 = \{x_0, x_1, x_2\}$ Exclude $x_3 \in F^{\otimes}$, the bad state.

Winning Region: Step-by-Step Computation

$X_0 = S^{\otimes} \setminus F^{\otimes}, \qquad X_{k+1} = (S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}(X_k).$

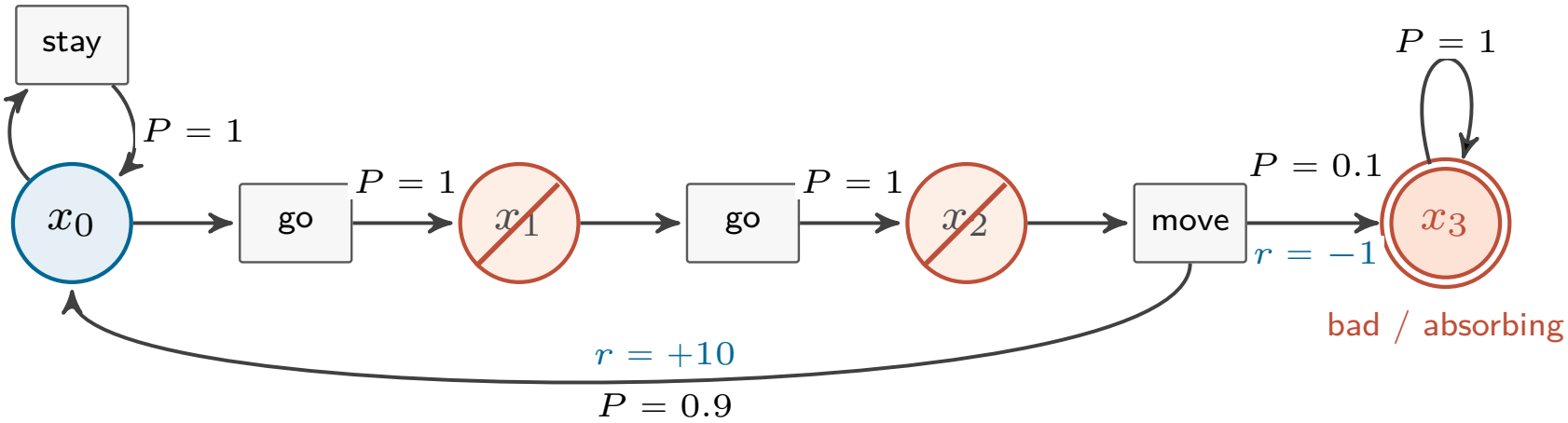


$X_0 = \{x_0, x_1, x_2\}$ Exclude $x_3 \in F^{\otimes}$, the bad state.

$X_1 = \{x_0, x_1\}$ Remove x_2: move reaches x_3 with probability 0.1, despite the 0.9 high-reward outcome.

Winning Region: Step-by-Step Computation

$X_0 = S^{\otimes} \setminus F^{\otimes}, \qquad X_{k+1} = (S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}(X_k).$



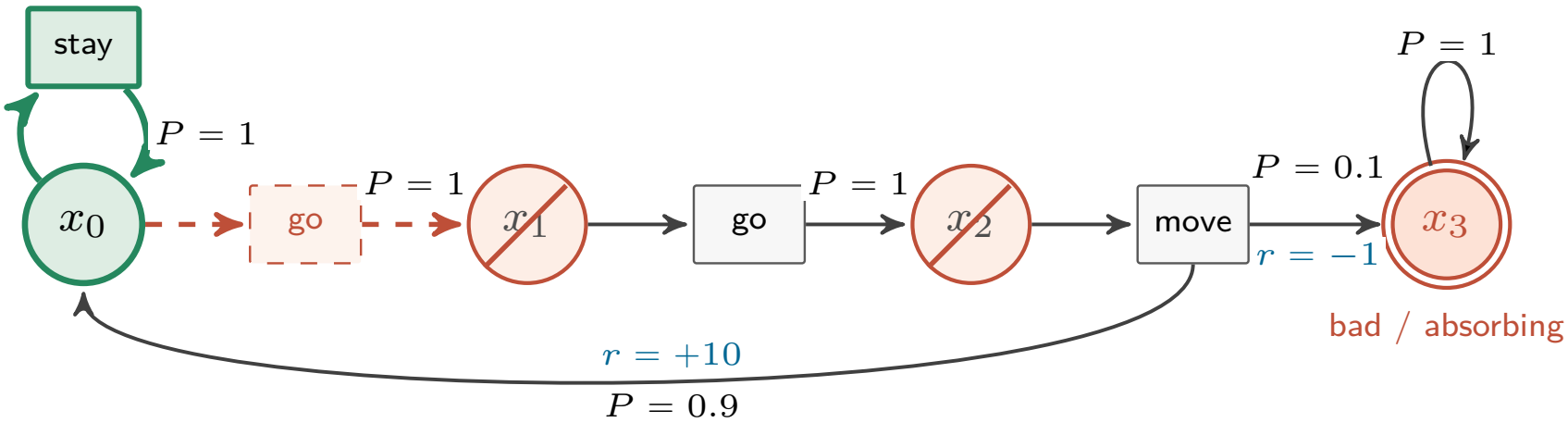
$X_0 = \{x_0, x_1, x_2\}$ Exclude $x_3 \in F^{\otimes}$, the bad state.

$X_1 = \{x_0, x_1\}$ Remove x_2: move reaches x_3 with probability 0.1, despite the 0.9 high-reward outcome.

$X_2 = \{x_0\}$ Remove x_1: its only action leads to $x_2 \notin X_1$.

Winning Region: Step-by-Step Computation

$X_0 = S^{\otimes} \setminus F^{\otimes}, \qquad X_{k+1} = (S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}(X_k).$



$X_0 = \{x_0, x_1, x_2\}$ Exclude $x_3 \in F^{\otimes}$, the bad state.

$X_1 = \{x_0, x_1\}$ Remove x_2: move reaches x_3 with probability 0.1, despite the 0.9 high-reward outcome.

$X_2 = \{x_0\}$ Remove x_1: its only action leads to $x_2 \notin X_1$.

$X_3 = X_2 = \{x_0\}$ No change: stay keeps x_0 inside X_2.

Winning region and winning actions

$\mathsf{W} = \{x_0\}, \qquad \mathsf{W}_{\mathsf{A}}(x_0) = \{\mathsf{stay}\}.$

Reward is irrelevant to this qualitative fixpoint: any non-zero support on a bad state makes an action non-winning.

Formal Guarantees: Existence & Termination

Existence of Greatest Fixed Point

Since f is monotone on the finite lattice $(2^{S^{\otimes}}, \subseteq)$, it has a greatest fixed point (Knaster–Tarski theorem).

Formal Guarantees: Existence & Termination

Existence of Greatest Fixed Point

Since f is monotone on the finite lattice $(2^{S^{\otimes}}, \subseteq)$, it has a greatest fixed point (Knaster–Tarski theorem).

Termination

The greatest fixed point is guaranteed to terminate. Starting from $X_0 = S^{\otimes} \setminus F^{\otimes}$,

$$X_0 \supseteq X_1 \supseteq X_2 \supseteq \cdots, \qquad X_{k+1} = f(X_k).$$

Hence the algorithm terminates after at most $|S^{\otimes} \setminus F^{\otimes}|$ strict decreases. However, we may find W $= \emptyset$.

Formal Guarantees: Complexity & Optimality

Linear complexity

With a worklist implementation, the winning region W can be computed in

$$O\left(|S^{\otimes}| + \sum_{s \in S^{\otimes}} |A^{\otimes}(s)| + |E^{\otimes}|\right),$$

where $E^{\otimes}$ is the set of non-zero transitions in the product MDP.

Intuitively, unsafe states are propagated backwards through the graph. Using a worklist, each state is removed at most once, each action is invalidated at most once, and each transition needs to be processed at most once. Hence the total work is **linear** in the size of the explicit product MDP.

Formal Guarantees: Complexity & Optimality

Linear complexity

With a worklist implementation, the winning region W can be computed in

$$O\left(|S^{\otimes}| + \sum_{s \in S^{\otimes}} |A^{\otimes}(s)| + |E^{\otimes}|\right),$$

where $E^{\otimes}$ is the set of non-zero transitions in the product MDP.

Intuitively, unsafe states are propagated backwards through the graph. Using a worklist, each state is removed at most once, each action is invalidated at most once, and each transition needs to be processed at most once. Hence the total work is **linear** in the size of the explicit product MDP.

Minimal interference theorem

Actions are only replaced when forced to, so minimal interference (a.k.a maximal permissiveness) is guaranteed, and the **optimal safe policy** can be found by shielding (Alshiekh et al, 2018).

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: Construct the product MDP $\mathcal{M} \otimes \mathcal{D}$ for the safety monitor $\mathcal{D}$.
3: Compute the winning region W and winning actions $\mathsf{W}_\mathsf{A}(s)$.
4: Construct the shielded product MDP by setting $A^{\mathsf{Sh}}(s) \leftarrow \mathsf{W}_\mathsf{A}(s)$ for every $s \in \mathsf{W}$.
5: Learn a memoryless policy $\pi^\star$ on the shielded product MDP with any RL algorithm.
6: **Return** $\pi^\star$.

Demo: Winning Region Shielding

MASA-Safe-RL/tutorial/05_minipacman_deterministic_shielding.ipynb

Shielding Multi-agent Systems

From RL to Multi-agent RL

Joint actions

For n agents, at time t the agents choose a **joint action**

$$\mathbf{a}_t = (a_t^1, \ldots, a_t^n) \in A_1 \times \cdots \times A_n.$$

The environment transition therefore depends on the joint action:

$$P(s' \mid s, \mathbf{a}).$$

From RL to Multi-agent RL

Joint actions

For n agents, at time t the agents choose a **joint action**

$$\mathbf{a}_t = (a_t^1, \ldots, a_t^n) \in A_1 \times \cdots \times A_n.$$

The environment transition therefore depends on the joint action:

$$P(s' \mid s, \mathbf{a}).$$

Policies

Each agent may have its own policy

$$\pi_i(a_i \mid s),$$

Forming the joint policy:

$$\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n).$$

Learning Strategically in Multi-agent RL

Behaviour of other agents

In multi-agent RL, the quality of agent i's policy depends on *how the other agents behave*:

$$J_i(\pi_i, \pi_{-i}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, \mathbf{a}_t) \right].$$

Learning Strategically in Multi-agent RL

Behaviour of other agents

In multi-agent RL, the quality of agent i's policy depends on *how the other agents behave*:

$$J_i(\pi_i, \pi_{-i}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, \mathbf{a}_t) \right].$$

Three strategic fragments

1. **Cooperative games**
2. **Zero-sum (a.k.a. competitive) games**
3. **General-sum games** (a.k.a. mixed-motive games)

Cooperative Stochastic Games

Cooperative Reward Maximisation

Agents share a common objective (reward) forming a **coordination problem**:

$$\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}) = \max_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t) \right].$$

Cooperative Stochastic Games

Cooperative Reward Maximisation

Agents share a common objective (reward) forming a **coordination problem**:

$$\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}) = \max_{\boldsymbol{\pi}} \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \mathbf{a}_t) \right].$$

Example: Cooperative Warehouse Robots

Several robots must move packages through a warehouse.

- Robots share a **common team reward**, e.g., number of packages delivered.
- They benefit from coordinating routes and avoiding blocking one another.
- The objective is to find a joint policy

$$\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}).$$

Key idea: everybody wins or loses together.

Zero-Sum / Adversarial Games

Two agents (or coalitions) have opposing objectives, e.g. $J_1 = -J_2$.

Zero-Sum Games

Agent 1 therefore solves

$$\max_{\pi_1} \min_{\pi_2} J_1(\pi_1, \pi_2).$$

Learning is about finding a strategy robust to an **adversarial best response**.

Zero-Sum / Adversarial Games

Two agents (or coalitions) have opposing objectives, e.g. $J_1 = -J_2$.

Zero-Sum Games

Agent 1 therefore solves

$$\max_{\pi_1} \min_{\pi_2} J_1(\pi_1, \pi_2).$$

Learning is about finding a strategy robust to an **adversarial best response**.

Example: Cyber Defender vs. Attacker

A defender protects a computer system while an attacker tries to compromise it.

- The defender is rewarded for keeping the system secure.
- The attacker is rewarded for reaching a compromised state.
- Their objectives are directly opposed:

$$J_{\text{defender}} = -J_{\text{attacker}}.$$

Key idea: the defender must remain secure against the attacker's adversarial best response.

General-Sum Games (Mixed Motive)

General-Sum Games

Each agent maximizes its own return J_i. A natural solution concept is a **Nash equilibrium** π^*, where no agent benefits from deviating alone:

$$J_i(\pi_i^*, \pi_{-i}^*) \geq J_i(\pi_i, \pi_{-i}^*) \qquad \forall i,\ \forall \pi_i.$$

Agents may therefore need to both **cooperate and compete as beneficial**.

General-Sum Games (Mixed Motive)

General-Sum Games

Each agent maximizes its own return J_i. A natural solution concept is a **Nash equilibrium** π^*, where no agent benefits from deviating alone:

$$J_i(\pi_i^*, \pi_{-i}^*) \geq J_i(\pi_i, \pi_{-i}^*) \qquad \forall i,\ \forall \pi_i.$$

Agents may therefore need to both **cooperate and compete as beneficial**.

Example: Mixed-Motive Autonomous Driving

Several autonomous vehicles interact at a junction or when merging.

- Each vehicle wants to reach its destination quickly.
- Vehicles benefit from cooperation, e.g. allowing another car to merge.
- But their objectives can conflict, e.g. both may prefer to go first.
- Each agent therefore has its own reward:

$$J_i(\pi_i, \pi_{-i}),$$

  and generally $J_i \neq -J_j$.

Key idea: agents sometimes benefit from cooperation and sometimes compete.

Centralised and Decentralised Learning

Centralised Training and Execution (CTCE)

A central controller observes the global state and selects the joint action:

$$\mathbf{a} \sim \pi(\cdot \mid s).$$

This allows agents to coordinate explicitly.

Centralised and Decentralised Learning

Centralised Training and Execution (CTCE)

A central controller observes the global state and selects the joint action:

$$\mathbf{a} \sim \pi(\cdot \mid s).$$

This allows agents to coordinate explicitly.

Centralised Training, Decentralised Execution (CTDE)

Training may use global state and joint-action information, while execution is local:

$$a_i \sim \pi_i(\cdot \mid o_i).$$

This enables shared information during learning without requiring a central controller at deployment.

Why safety becomes interesting

Safety factorisation

Safety may depend on **combinations of agents' actions**. Consequently, a shield may reason about:

- individual/local actions, or
- the complete joint action $\mathbf{a}$.

Decentralised training induces **non-stationarity** from the perspective of any single-agent.

Game-theoretic dynamics

Other agents may be adversarial from a safety-perspective - they do not care about the safety of others, only maximising their reward.

Coalitional Winning Region: Robust Fixed Point

Robust controllable predecessor

Let $C \subseteq \{1, \ldots, n\}$ be the focal coalition and $\bar{C} = \{1, \ldots, n\} \setminus C$. For $s^{\otimes} = (s, q)$,

$$\mathsf{CPre}_C(X) = \left\{ s^{\otimes} \;\middle|\; \exists \mathbf{a}_C \in A_C(s)\ \forall \mathbf{a}_{\bar{C}} \in A_{\bar{C}}(s) : \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) \subseteq X \right\}.$$

The coalition chooses once; agents outside C are unrestricted.

Coalitional Winning Region: Robust Fixed Point

Robust controllable predecessor

Let $C \subseteq \{1, \ldots, n\}$ be the focal coalition and $\bar{C} = \{1, \ldots, n\} \setminus C$. For $s^{\otimes} = (s, q)$,

$$\mathsf{CPre}_C(X) = \left\{ s^{\otimes} \;\middle|\; \exists \mathbf{a}_C \in A_C(s)\ \forall \mathbf{a}_{\bar{C}} \in A_{\bar{C}}(s) : \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) \subseteq X \right\}.$$

The coalition chooses once; agents outside C are unrestricted.

Greatest fixed point

$$\boxed{\mathsf{W}^C = \nu X . \Big((S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}_C(X) \Big)}$$

Equivalently, $X_0 = S^{\otimes} \setminus F^{\otimes}$ and $X_{k+1} = X_k \cap \mathsf{CPre}_C(X_k)$ until convergence.

Coalitional Winning Region: Robust Fixed Point

Robust controllable predecessor

Let $C \subseteq \{1, \ldots, n\}$ be the focal coalition and $\bar{C} = \{1, \ldots, n\} \setminus C$. For $s^{\otimes} = (s, q)$,

$$\mathsf{CPre}_C(X) = \left\{ s^{\otimes} \;\middle|\; \exists \mathbf{a}_C \in A_C(s)\ \forall \mathbf{a}_{\bar{C}} \in A_{\bar{C}}(s) : \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) \subseteq X \right\}.$$

The coalition chooses once; agents outside C are unrestricted.

Greatest fixed point

$$\boxed{\mathsf{W}^C = \nu X . \Big((S^{\otimes} \setminus F^{\otimes}) \cap \mathsf{CPre}_C(X) \Big)}$$

Equivalently, $X_0 = S^{\otimes} \setminus F^{\otimes}$ and $X_{k+1} = X_k \cap \mathsf{CPre}_C(X_k)$ until convergence.

Key quantifier order

$$\boxed{\exists \mathbf{a}_C\ \forall \mathbf{a}_{\bar{C}}\ \forall {s'}^{\otimes} \in \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) :\ {s'}^{\otimes} \in X}$$

This is ***not*** $\forall \mathbf{a}_{\bar{C}} \exists \mathbf{a}_C$: the coalition cannot react after observing the outsiders' simultaneous choices.

Coalitional Winning Actions: Centralised vs. Decentralised

Centralised execution - One controller for the coalition

For $s^{\otimes} \in \mathsf{W}^{C}$, keep every robust coalition tuple:

$$\mathsf{W}_{\mathsf{A}}^{C}(s^{\otimes}) = \{\mathbf{a}_C \in A_C(s) \mid \forall \mathbf{a}_{\bar{C}} \in A_{\bar{C}}(s) : \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) \subseteq \mathsf{W}^{C}\}.$$

Decentralised execution - n controllers for the coalition

Independent simultaneous choices require local masks with

$$\boxed{\prod_{i \in C} M_i(s^{\otimes}) \subseteq \mathsf{W}_{\mathsf{A}}^{C}(s^{\otimes})}$$

Hence *every* locally selectable combination is winning against all non-coalition actions. No teammate action is observed.

Coalitional Winning Actions: Centralised vs. Decentralised

Centralised execution - One controller for the coalition

For $s^{\otimes} \in \mathsf{W}^{C}$, keep every robust coalition tuple:

$$\mathsf{W}_{\mathsf{A}}^{C}(s^{\otimes}) = \{\mathbf{a}_C \in A_C(s) \mid \forall \mathbf{a}_{\bar{C}} \in A_{\bar{C}}(s) : \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) \subseteq \mathsf{W}^{C}\}.$$

Decentralised execution - n controllers for the coalition

Independent simultaneous choices require local masks with

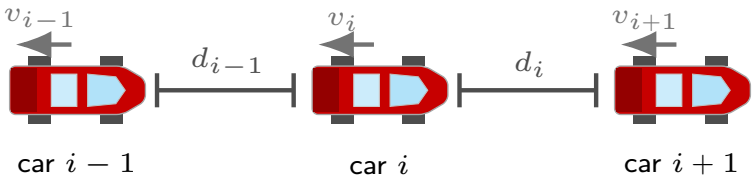
$$\boxed{\prod_{i \in C} M_i(s^{\otimes}) \subseteq \mathsf{W}_{\mathsf{A}}^{C}(s^{\otimes})}$$

Hence *every* locally selectable combination is winning against all non-coalition actions. No teammate action is observed.

Coordination-only winning actions cannot be factorised freely

If $\mathsf{W}_{\mathsf{A}}^{C}(s^{\otimes}) = \{(L, L), (R, R)\}$, then giving both agents $\{L, R\}$ is unsound: $\{L, R\} \times \{L, R\}$ also admits (L, R) and (R, L). A decentralised shield must instead expose a safe rectangle, e.g. $\{L\} \times \{L\}$ or $\{R\} \times \{R\}$.

Projection to Local Action Masks



$$\phi = \bigwedge_i (0 < d_i < 200)$$

Local view. Car i observes (v_i, v_{i+1}, d_i) and picks $a_i \in \{-2, 0, 2\}\,\mathrm{m/s^2}$.

- The global property is relational, but each policy/shield sees only a slice of the platoon.
- A shield for car i cannot rule out the car behind crashing into it.
- The resulting damaged stop can make car i fail safety with respect to car $i+1$.
- The greatest fixed point therefore needs to be computed simultaneously or ordered e.g.

$$\langle \bigwedge_{j<i} \phi_j \rangle \, \mathcal{M}^i_{\mathsf{Sh}} \, \langle \phi_i \rangle$$

- This can be projected to local action masks, it simply means that the coalition is aware of the safety promise that other agents will enforce, to enable a greater set of behaviour.

Source of figure: Brorholt et al, 2025

Circular Teammate Reliance: Why It Is Sound

Circular-Assume Guarantee Soundness

(Adalat et al, 2026): Agent $i \in C$ does not know its teammates' simultaneous actions. It may only rely on each teammate $j \in C$ obeying its certified mask $M_j(s^{\otimes})$. Agents outside C carry no guarantee. This is a **permitted assumption**, as circular assume–guarantee reliance is sound when the whole tuple is certified simultaneously.

Circular Teammate Reliance: Why It Is Sound

Circular-Assume Guarantee Soundness

(Adalat et al, 2026): Agent $i \in C$ does not know its teammates' simultaneous actions. It may only rely on each teammate $j \in C$ obeying its certified mask $M_j(s^{\otimes})$. Agents outside C carry no guarantee. This is a **permitted assumption**, as circular assume–guarantee reliance is sound when the whole tuple is certified simultaneously.

One simultaneous certificate

The local masks are accepted only when, together,

$$\forall \mathbf{a}_C \in \prod_{i \in C} M_i(s^{\otimes}),\ \forall \mathbf{a}_{\bar{C}} \in A_{\bar{C}}(s): \quad \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) \subseteq \mathsf{W}^C.$$

Thus the guarantee covers *all independently selectable combinations*, not runtime coordination between teammates.

Circular Teammate Reliance: Why It Is Sound

Circular-Assume Guarantee Soundness

(Adalat et al, 2026): Agent $i \in C$ does not know its teammates' simultaneous actions. It may only rely on each teammate $j \in C$ obeying its certified mask $M_j(s^{\otimes})$. Agents outside C carry no guarantee. This is a **permitted assumption**, as circular assume–guarantee reliance is sound when the whole tuple is certified simultaneously.

One simultaneous certificate

The local masks are accepted only when, together,

$$\forall \mathbf{a}_C \in \prod_{i \in C} M_i(s^{\otimes}),\ \forall \mathbf{a}_{\bar{C}} \in A_{\bar{C}}(s): \quad \operatorname{supp} P^{\otimes}(\cdot \mid s^{\otimes}, \mathbf{a}_C, \mathbf{a}_{\bar{C}}) \subseteq \mathrm{W}^C.$$

Thus the guarantee covers *all independently selectable combinations*, not runtime coordination between teammates.

Invariant argument

Start in $s_0^{\otimes} \in \mathrm{W}^C$. If every coalition member obeys its local mask, then for arbitrary outsider actions every possible successor remains in W^C. By induction, $F^{\otimes}$ is never reached.

Open problems

- **Scalability:** Product state spaces and joint-action spaces grow exponentially w.r.t. the number of agents.

Open problems

- **Scalability:** Product state spaces and joint-action spaces grow exponentially w.r.t. the number of agents.
- **Unknown dynamics:** How can we provide meaningful safety guarantees while simultaneously learning the transition model?

Open problems

- **Scalability:** Product state spaces and joint-action spaces grow exponentially w.r.t. the number of agents.
- **Unknown dynamics:** How can we provide meaningful safety guarantees while simultaneously learning the transition model?
- **Strategic agents:** How should safety guarantees change when other agents are adaptive, competitive, or do not respect our assumptions?

Open problems

- **Scalability:** Product state spaces and joint-action spaces grow exponentially w.r.t. the number of agents.
- **Unknown dynamics:** How can we provide meaningful safety guarantees while simultaneously learning the transition model?
- **Strategic agents:** How should safety guarantees change when other agents are adaptive, competitive, or do not respect our assumptions?
- **Shielding agents that communicate:** How do we effectively shield agents that communicate, when communication is often *emergent*? What about adverserial communication?

References

References

- Richard S. Sutton, Andrew G. Barto: Reinforcement Learning - An Introduction, 2nd Edition. MIT Press 2018
- Orna Kupferman, Moshe Y. Vardi: Model Checking of Safety Properties. CAV 1999: 172-183
- Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, Ufuk Topcu: Safe Reinforcement Learning via Shielding. AAAI 2018: 2669-2678
- Ingy Elsayed-Aly, Suda Bharadwaj, Christopher Amato, Rüdiger Ehlers, Ufuk Topcu, Lu Feng: Safe Multi-Agent Reinforcement Learning via Shielding. AAMAS 2021
- Stefano V. Albrecht, Filippos Christianos, Lukas Schäfer: Multi-Agent Reinforcement Learning: Foundations and Modern Approaches. MIT Press 2024
- Omar Adalat, Edwin Hamel-De le Court, Francesco Belardinelli: Contract-Based Compositional Shielding for Safe Multi-Agent Reinforcement Learning, EUMAS 2026

- 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!