AALBORG UNIVERSITET

Shielding for Safe Reinforcement Learning in Complex Environments

Christian Schilling

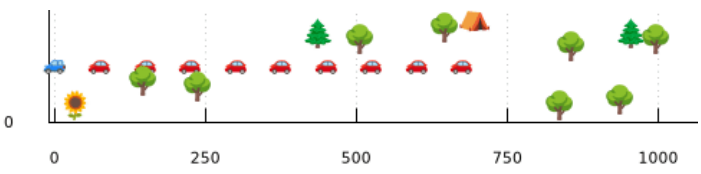


Table of contents

Motivation

Shielding

Shielding for finite state spaces

Shielding for infinite state spaces

Shielding for multi-agent systems

Summary

Table of contents

Motivation

Shielding

Shielding for finite state spaces

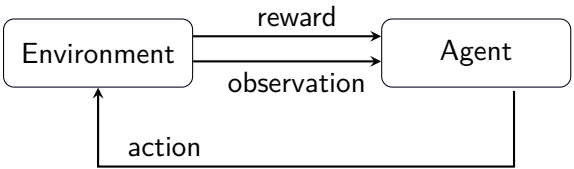
Shielding for infinite state spaces

Shielding for multi-agent systems

Summary

Policies for sequential decision making

- Agent iteratively selects an **action**
- Environment transitions to new **state** (observation) and yields a **reward**
- Goal: find a **policy** π for the agent that maximizes cumulative reward



- **Reinforcement learning (RL)**
  - Assumption: environment is a **black-box** Markov decision process
  - Approach: learn good actions through experience

How trustworthy are learned policies?

- Q-learning **converges to optimal solution**
  - Deep RL generally **does not**
- In practice, one interrupts the training process and hopes to obtain a **close-to-optimal policy**
  - Close-to-optimal policies may still **fail catastrophically**
- Every environment transition must be **observed in training often enough**
  - Some initial conditions may never be sampled
  - Rare events may not be observed
  - Training environment may miss some details

(How) Can one encode (safety) constraints in RL?

- Only way to express constraints in RL: **reward function**
  - No "hard" constraints; everything expressed in a numeric value
- Common approach to "safe RL" is **reward shaping** (punish unsafe actions)
  - Is that a good interface?

(How) Can one encode (safety) constraints in RL?

- Only way to express constraints in RL: **reward function**
  - No "hard" constraints; everything expressed in a numeric value
- Common approach to "safe RL" is **reward shaping** (punish unsafe actions)
  - Is that a good interface? Often no!
- Hard to express what you want
- Can lead to *specification gaming* / *reward hacking*

Researchers from Niels Bohr Institute (1998)

> *"[The] heterogeneous reinforcement functions have to be designed with great care. In our first experiments we rewarded the agent for driving towards the goal but did not punish it for driving away from it. Consequently the agent drove in circles with a radius of 20–50 meters around the starting point."* [1]

[1] J. Randløv and P. Alstrøm. *ICML*. 1998

(How) Can one encode (safety) constraints in RL?

- Only way to express constraints in RL: **reward function**
  - No "hard" constraints; everything expressed in a numeric value
- Common approach to "safe RL" is **reward shaping** (punish unsafe actions)
  - Is that a good interface? Often no!
- Hard to express what you want
- Why not simply assign a very low reward to safety violations?
  - How low is "very low"?
  - Must outweigh all positive alternatives, even for long, low-probability traces
  - Otherwise, even the optimal policy would take the risk

Example ("How to assign rewards here?")

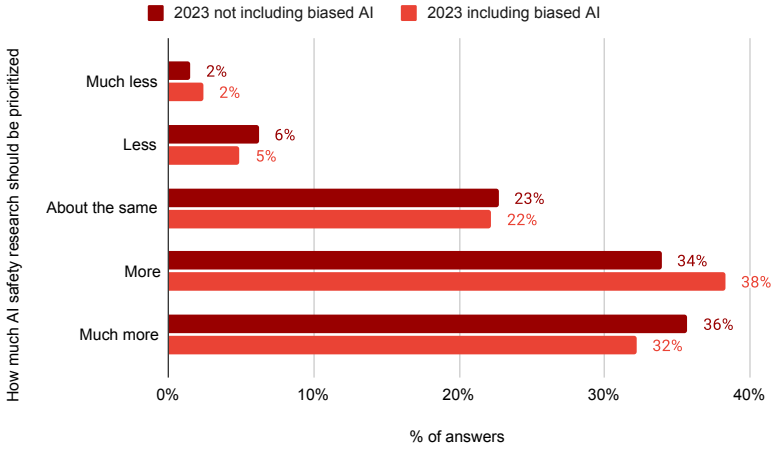
- Action a: get 1 gold bar
- Action b: get 2 gold bars and then throw 100 fair coins
- Safety property: never throw heads 100 times

(How) Can one encode (safety) constraints in RL?

- Only way to express constraints in RL: **reward function**
  - No "hard" constraints; everything expressed in a numeric value
- Common approach to "safe RL" is **reward shaping** (punish unsafe actions)
  - Is that a good interface? Often no!
- Hard to express what you want
- Why not simply assign a very low reward to safety violations?
  - How low is "very low"?
  - Must outweigh all positive alternatives, even for long, low-probability traces
  - Otherwise, even the optimal policy would take the risk
- Empirically, **safety is often in conflict with performance**
  - E.g., drive as fast as possible but do not leave the track

2023 survey among AI researchers about AI safety[1]

"How much should AI safety research be prioritized?"



[1] K. Grace et al. (2023).

Table of contents

Motivation

Shielding

Shielding for finite state spaces

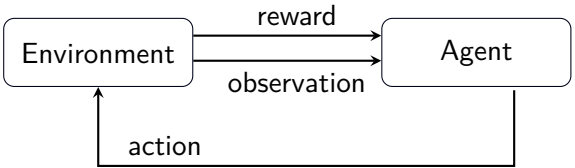
Shielding for infinite state spaces

Shielding for multi-agent systems

Summary

Shielding for safe RL

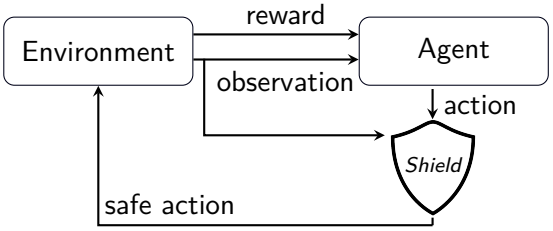
- Add a **shield**[1] for ensuring safety in the RL process
- Idea: shield prevents all unsafe actions



[1] M. Alshiekh et al. *AAAI*. 2018, B. Könighofer et al. *Commun. ACM* (2025).

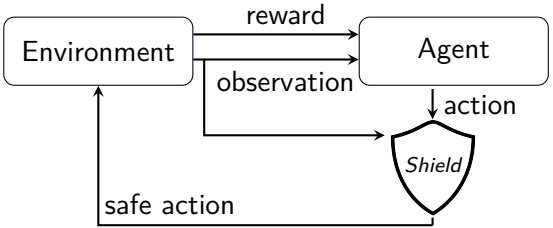
Shielding for safe RL

Idea 1:

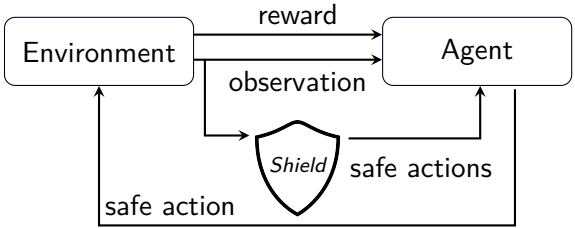


Shielding for safe RL

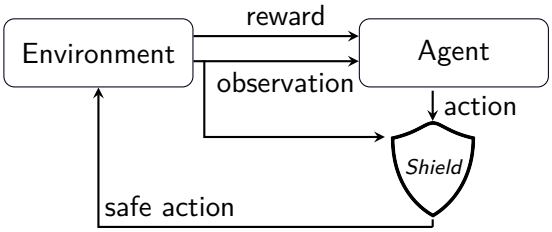
Idea 1:



Idea 2:

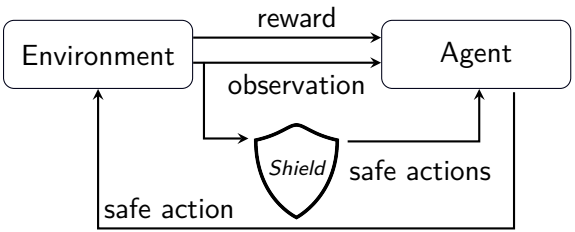


Idea 1: "Post-shielding"



- May **override** agent action with a safe action
  - How to implement? (largest allowed Q-value?)
  - Agent not aware of shield $\rightsquigarrow$ shield must stay in place after training

Idea 2: "Pre-shielding"



- **Only allows safe actions**
  - Conceptually: nondeterministic policy where each allowed action is safe
  - If agent converged, shield can be removed
- From now on, we simply call pre-shielding "**shielding**"

Shield is active during learning

- **No reward shaping** for safety needed
- Not exploring unsafe actions $\rightsquigarrow$ **safe during learning & faster convergence**
- Agent learns to "work with the shield"

How would the shield know which behavior is safe?

- Assumption: **we have a model**
- Then why use RL? We can compute an optimal policy from the model, right?

How would the shield know which behavior is safe?

- Assumption: **we have a model**
- Then why use RL? We can compute an optimal policy from the model, right?
  - Not necessarily, because the model can be **abstract**
  - Safety often does not care about probabilities, only **possibilities**
    - $\rightsquigarrow$ We do not need to know the exact probabilities
      Only need to know whether probability is (non)zero
    - $\rightsquigarrow$ Model need not contain state variables irrelevant to safety
      We still want to use RL to optimize performance

Table of contents

Motivation

Shielding

Shielding for finite state spaces

Shielding for infinite state spaces

Shielding for multi-agent systems

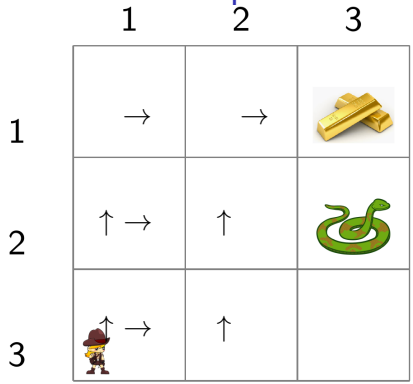
Summary

Example

	1	2	3
1	↑ →	↑ →	
2	↑ →	↑ →	
3	 ↑ →	↑ →	↑ →

- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: **deterministic**, action is always successful
- Safety requirement: do not run into snake or wall

Example



- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: **deterministic**, action is always successful
- Safety requirement: do not run into snake or wall

Algorithm to compute a shield

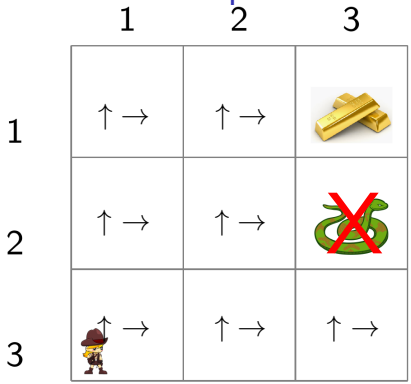
- Recursive definition:
  - A state that violates the safety property is **unsafe**
  - Any action that leads to an **unsafe** state is **forbidden**
  - A state that only has **forbidden** actions is also **unsafe**
- We look for the smallest fixpoint of this description
- Easy to compute for finite state spaces

Example

	1	2	3
1	↑ →	↑ →	
2	↑ →	↑ →	
3	 ↑ →	↑ →	↑ →

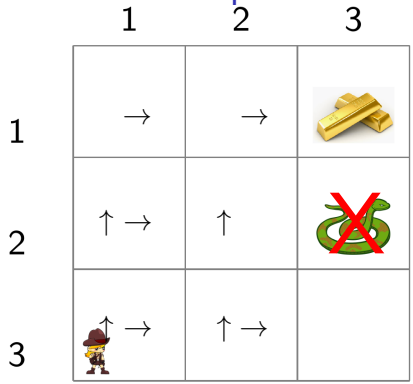
- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: **deterministic**, action is always successful
- Safety requirement: do not run into snake or wall

Example



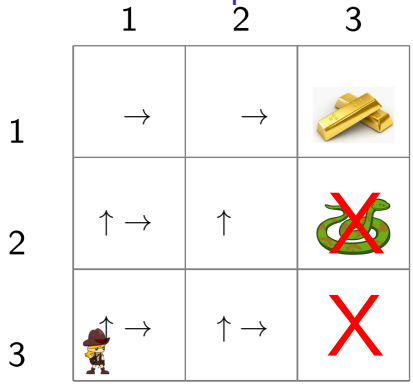
- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: **deterministic**, action is always successful
- Safety requirement: do not run into snake or wall

Example



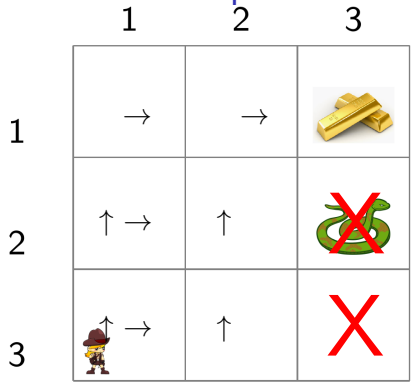
- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: **deterministic**, action is always successful
- Safety requirement: do not run into snake or wall

Example



- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: **deterministic**, action is always successful
- Safety requirement: do not run into snake or wall

Example



- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: **deterministic**, action is always successful
- Safety requirement: do not run into snake or wall

Drawbacks of shielding

- **Requires a model** (but only qualitative, i.e., no probabilities)
  - To be fair, without any model, there cannot be any guarantees

Drawbacks of shielding

- **Requires a model** (but only qualitative, i.e., no probabilities)
  - To be fair, without any model, there cannot be any guarantees
- Complexity: $\mathcal{O}(|S|^2|A|)$ for $|S|$ states and $|A|$ actions
  - But $|S|$ is typically **exponential** in the number of state variables

Drawbacks of shielding

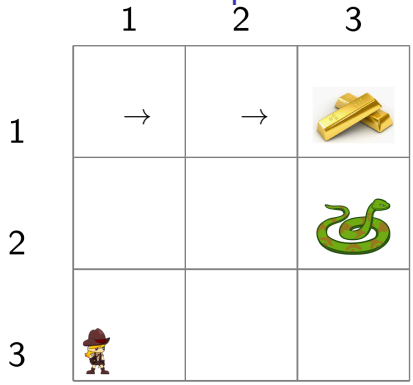
- **Requires a model** (but only qualitative, i.e., no probabilities)
  - To be fair, without any model, there cannot be any guarantees
- Complexity: $\mathcal{O}(|S|^2|A|)$ for $|S|$ states and $|A|$ actions
  - But $|S|$ is typically **exponential** in the number of state variables
- Shield is **qualitative** – an action is either safe or not
  - In some applications, small risks of failure may be acceptable
  - There may be a low failure probability no matter how hard you try

Example

	1	2	3
1	↑ →	↑ →	
2	↑ →	↑ →	
3	 ↑ →	↑ →	↑ →

- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: $\uparrow$: **small probability of moving to the right instead**
- Safety requirement: do not run into snake or wall

Example



- States: grid positions of adventurer plus "dead" state; actions: $\uparrow, \rightarrow$
- Environment: $\uparrow$: **small probability of moving to the right instead**
- Safety requirement: do not run into snake or wall

Table of contents

Motivation

Shielding

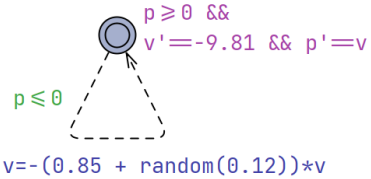
Shielding for finite state spaces

Shielding for infinite state spaces

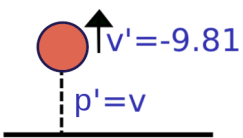
Shielding for multi-agent systems

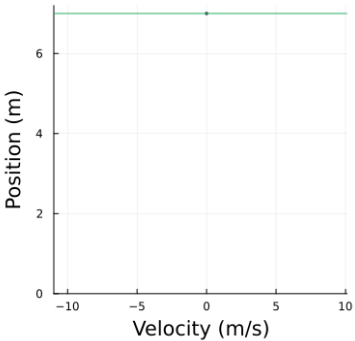
Summary

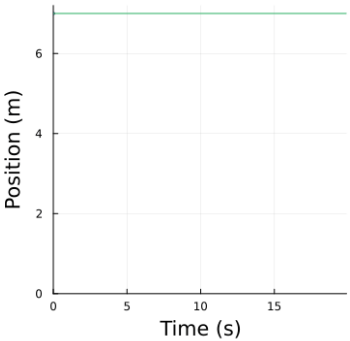
Running example: Bouncing ball



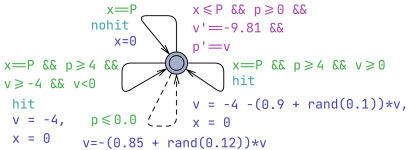
- Euclidean Markov decision process (EMDP)
  - Continuous state space and time
  - Discrete-time events (bounce)
  - Stochastic uncertainty (bounce friction)



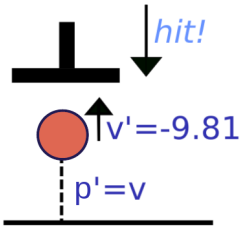


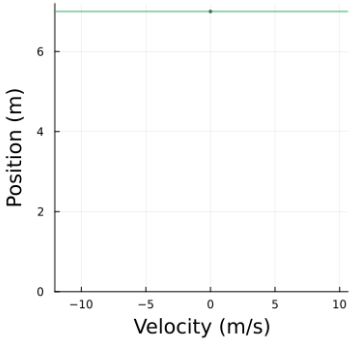


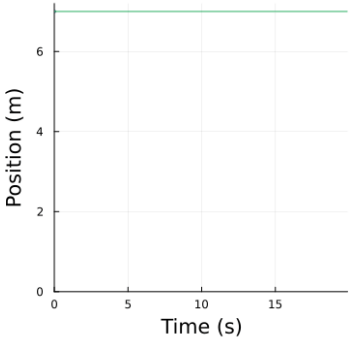
Running example: Bouncing ball



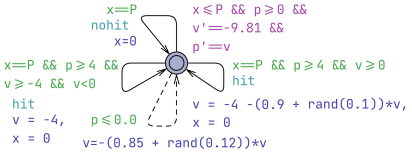
- Add (discrete) actions *hit*/*nohit*
  - *hit*: If $p \geq 4$, ball gets pushed down
  - Agent can choose action periodically
- Goal 1: minimize number of *hit* actions



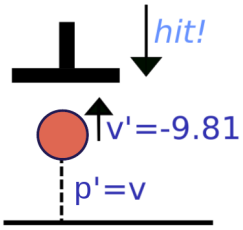


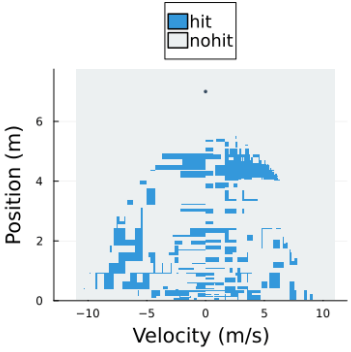


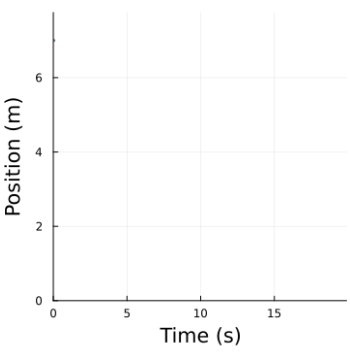
Running example: Bouncing ball



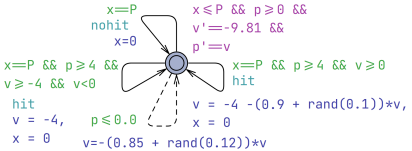
- Goal 2: keep the ball bouncing
- We learned a policy with Uppaal Stratego
  - Uses adaptive state-space partitioning
  - Trained for 12,000 episodes



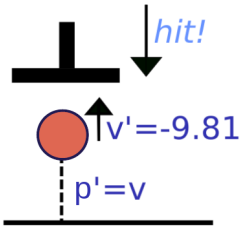


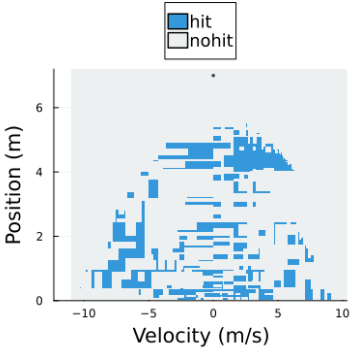


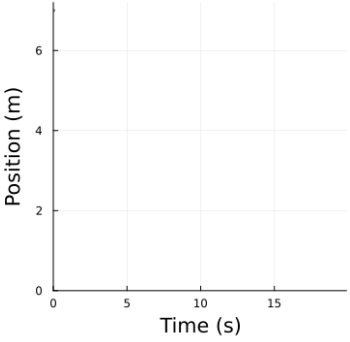
Running example: Bouncing ball



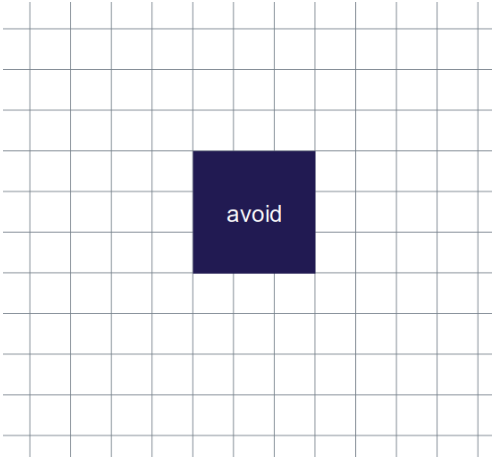
- Safety constraint: $p = 0 \implies |v| > 1$
- Statistical analysis: 2 % of executions unsafe





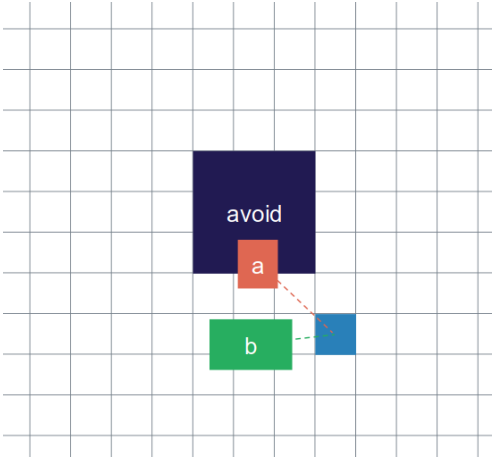


Computing a shield via state-space partitioning



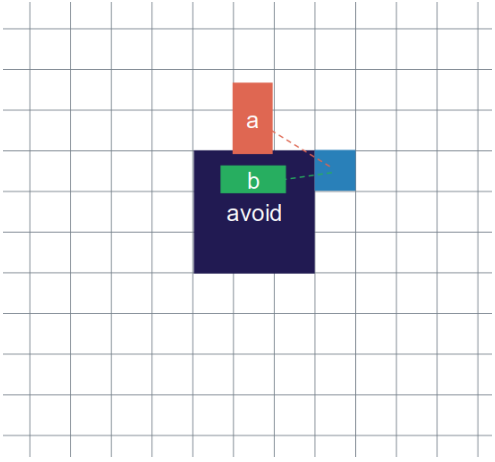
- Define a partition (e.g., a grid)
- Avoid set = cells with unsafe states
- For each cell, compute successor cells under each action
- Then use known algorithm

Computing a shield via state-space partitioning



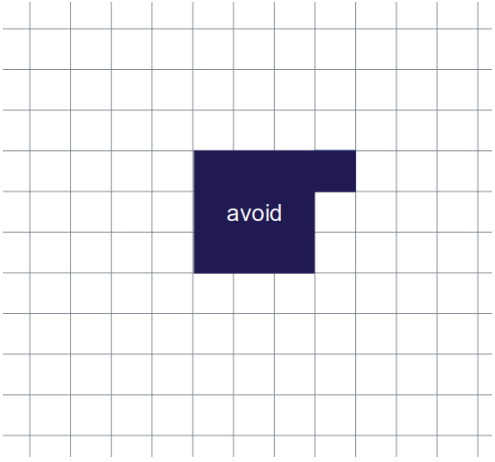
- Define a partition (e.g., a grid)
- Avoid set = cells with unsafe states
- For each cell, compute successor cells under each action
- Then use known algorithm
  - Forbid actions that reach the avoid set

Computing a shield via state-space partitioning



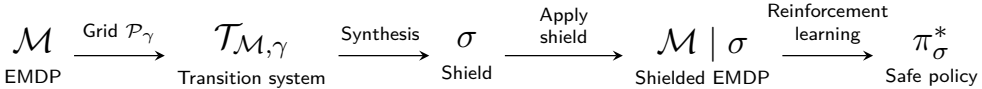
- Define a partition (e.g., a grid)
- Avoid set = cells with unsafe states
- For each cell, compute successor cells under each action
- Then use known algorithm
  - Forbid actions that reach the avoid set
  - If no action is safe,

Computing a shield via state-space partitioning



- Define a partition (e.g., a grid)
- Avoid set = cells with unsafe states
- For each cell, compute successor cells under each action
- Then use known algorithm
  - Forbid actions that reach the avoid set
  - If no action is safe, add the cell to the avoid set

Computing a shield via state-space partitioning

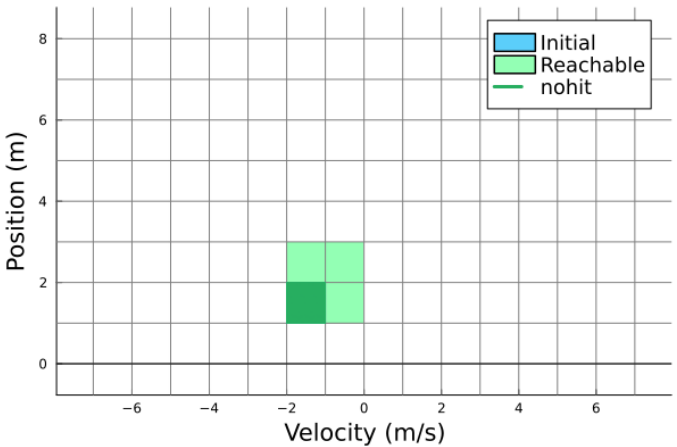


Theorem[1]

Any shield σ for $\mathcal{T}_{\mathcal{M},\gamma}$, when lifted to $\mathcal{M}$, is safe for $\mathcal{M}$

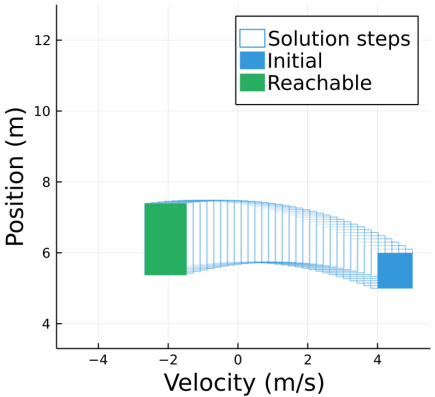
[1] A. H. Brorholt et al. *AISoLA*. 2023.

How to compute reachable cells?



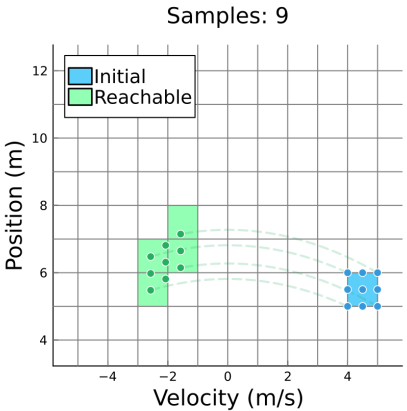
- Reachability problem is **undecidable** (except for very restrictive classes)

Computation via symbolic reachability analysis



- Sound result
- May overapproximate
- Generally expensive

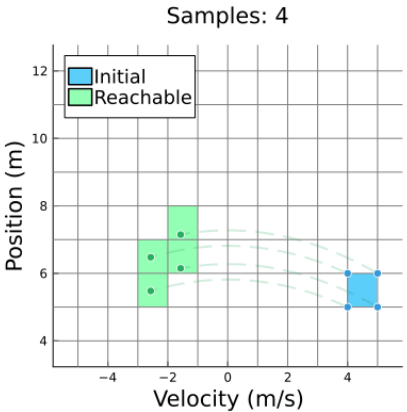
Computation via simulation-based approach[1]



- Cheap
- May miss behavior
- Accuracy improves with more samples
- Black-box access to environment

[1] A. H. Brorholt et al. *AISoLA*. 2023.

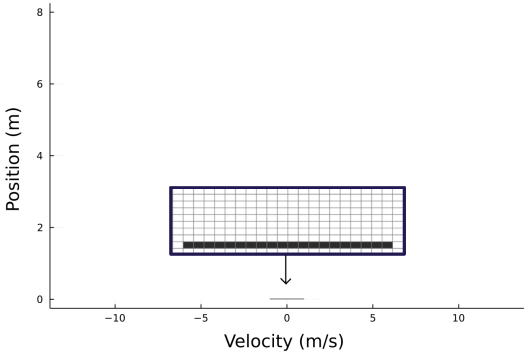
Computation via simulation-based approach[1]



- Cheap
- May miss behavior
- Accuracy improves with more samples
- Black-box access to environment

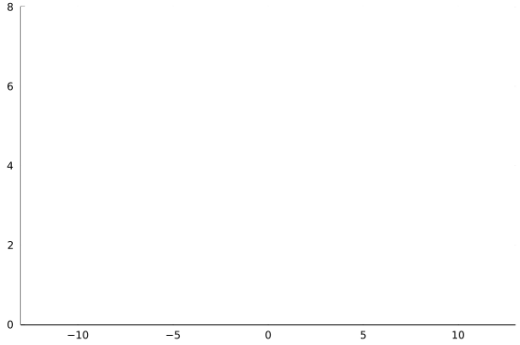
[1] A. H. Brorholt et al. *AISoLA*. 2023.

Fixpoint iteration



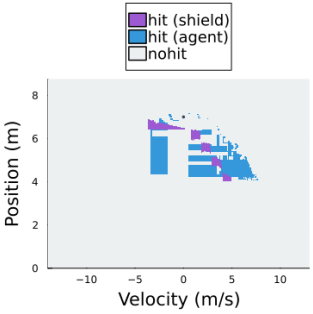
- Cell diameter 0.02
- 520,000 cells

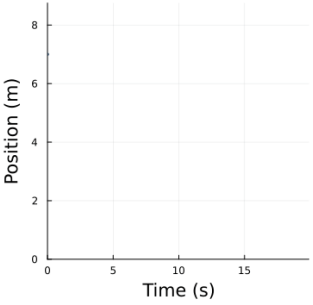
Fixpoint iteration

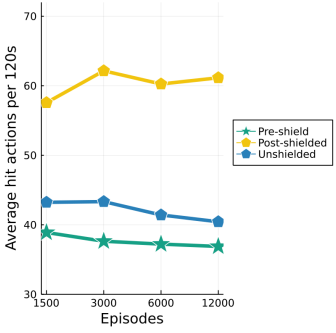


- Cell diameter 0.02
- 520,000 cells
- Computation: 134 sec

Evaluation of policy trained with a shield







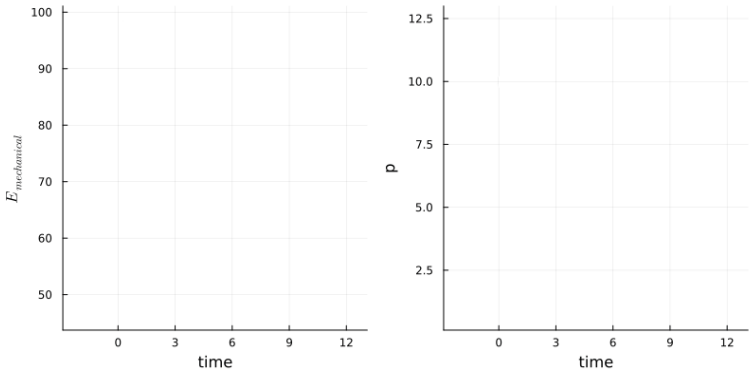
- Integrated in Uppaal[1]

[1] A. H. Brorholt et al. *RP*. 2025.

A different perspective: Mechanical energy

$$E_{mechanical} = mgp + 0.5mv^2$$

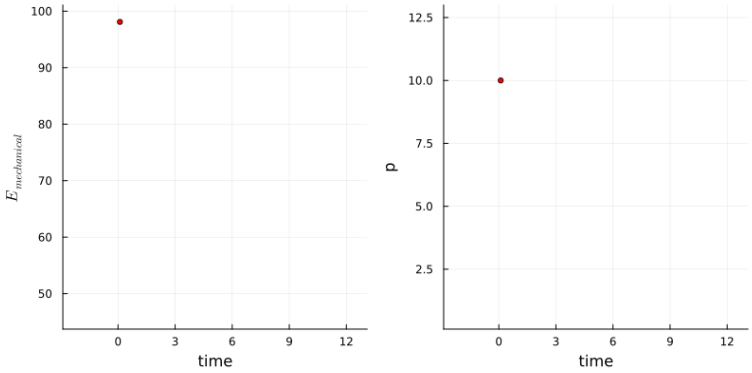
with mass m (1 kg), gravity g (9.81 m/s^2), position p, velocity v



A different perspective: Mechanical energy

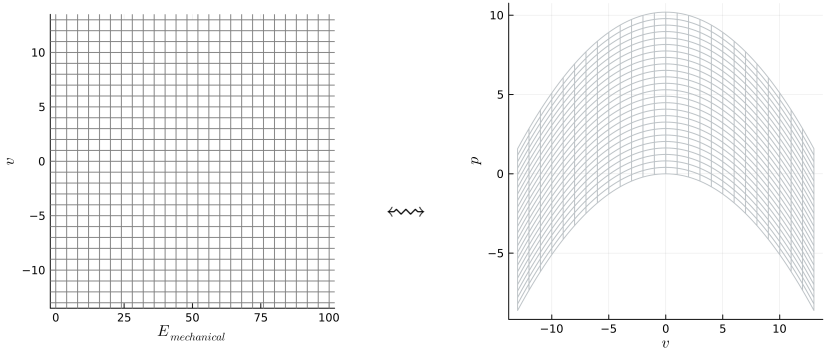
$$E_{mechanical} = mgp + 0.5mv^2$$

with mass m (1 kg), gravity g (9.81 m/s^2), position p, velocity v

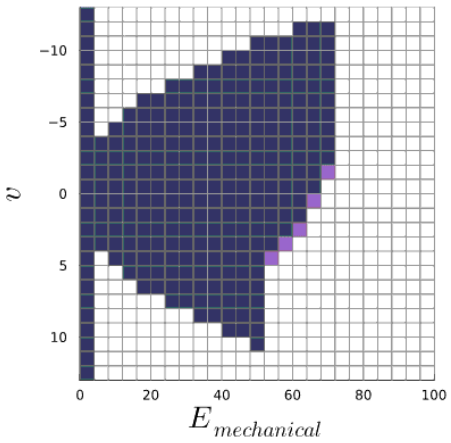


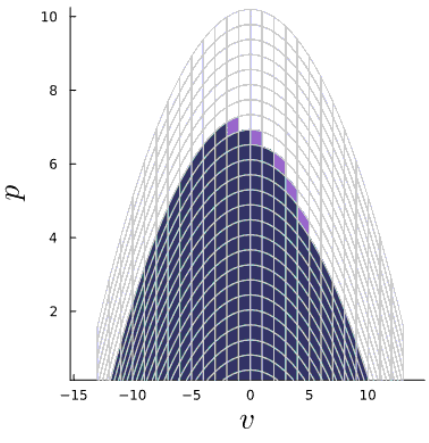
Transformed state space

- New state dimensions ($E_{mechanical}, v$)
- We construct a regular grid again
- Much fewer cells (650) are sufficient
- Requirement: can go back and forth wrt. original state space (v, p)

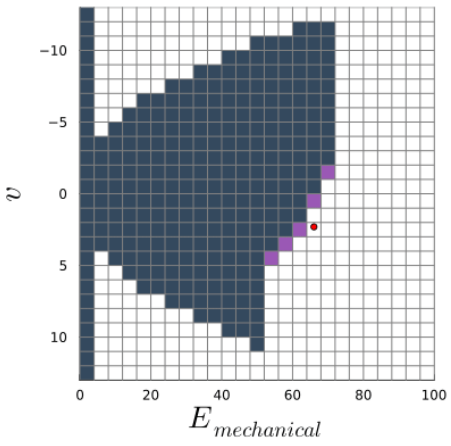


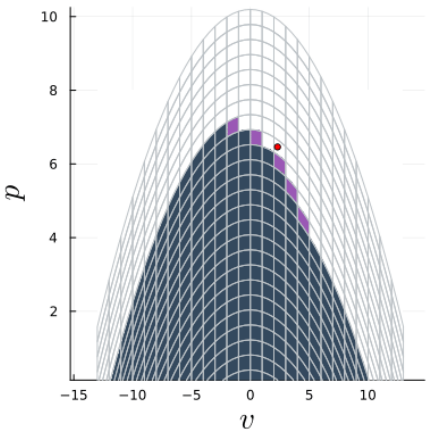
Shield in transformed and original state space



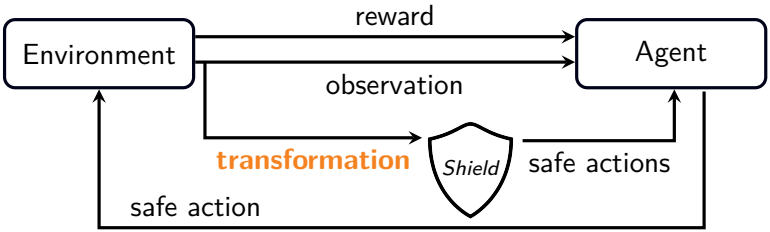


Shield in transformed and original state space

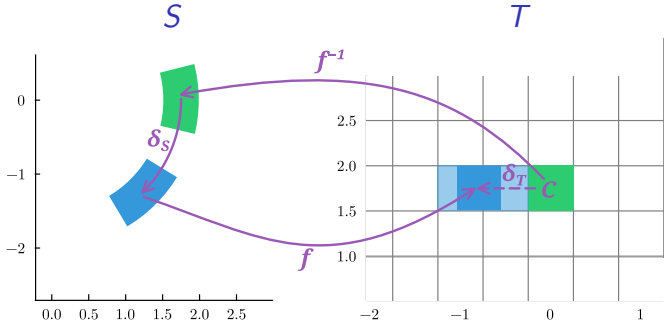




Shielding with state-space transformation



Shield computation with state-space transformation

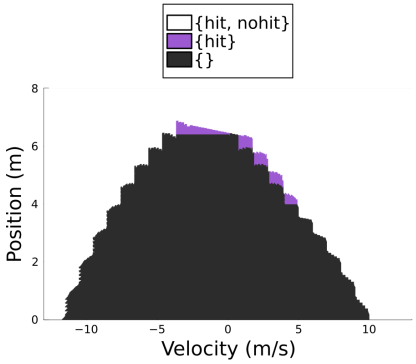


Compute successors δ_T from C (green) in transformed state space T

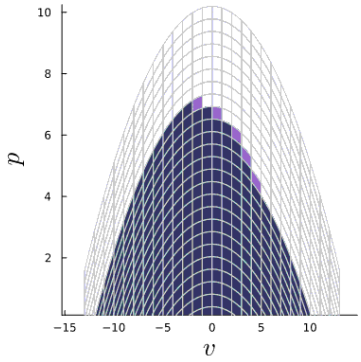
1. Map to original state space S via f^{-1}
2. Compute successors via δ_S
3. Map back to transformed state space T via f (dark blue)

As before, identify all cells intersecting with this set (light blue)

Evaluation

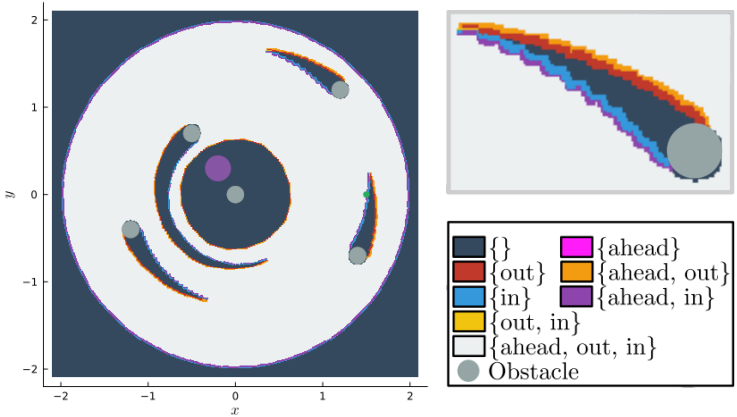


- 520,000 cells
- Computation: 134 sec
- Decision tree: 940 nodes



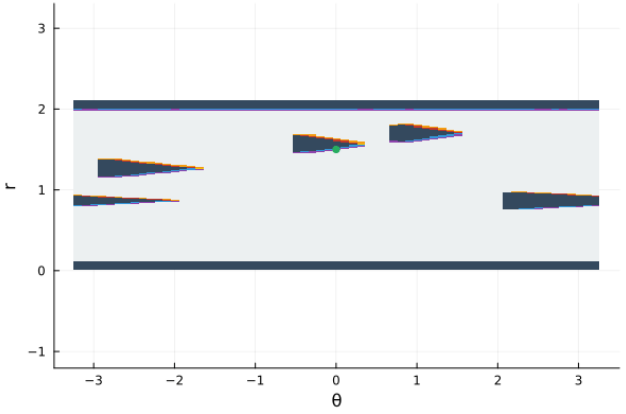
- 650 cells
- Computation: 1.3 sec
- Decision tree: 49 nodes

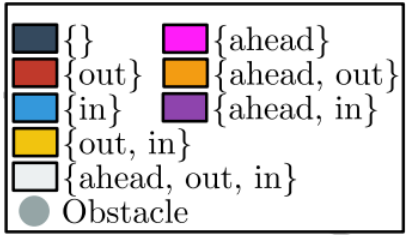
Satellite: Harmonic oscillator + {ahead, out, in}



- Goal: reach a randomly spawning purple area
- Here this is impossible because the shield forbids it

Satellite: Transformed state space

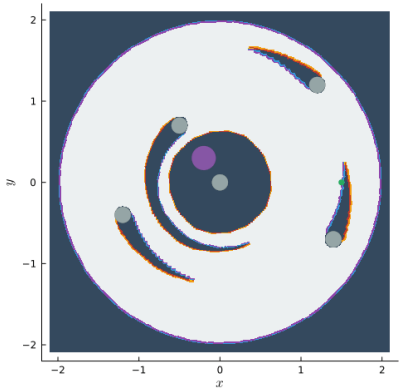




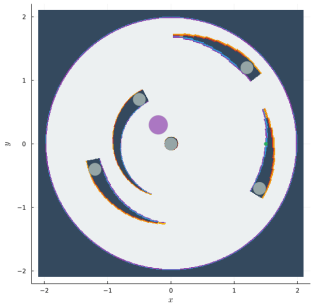
- Transformation to polar coordinates

$$f(x, y) = (\text{atan2}(y, x), \sqrt{x^2 + y^2}) =: (\theta, r)$$

Satellite: Comparison



- 176,400 cells
- Computation: 161 sec
- Decision tree: 4,913 nodes



- 27,300 cells
- Computation: 10 sec
- Decision tree: 544 nodes

Table of contents

Motivation

Shielding

Shielding for finite state spaces

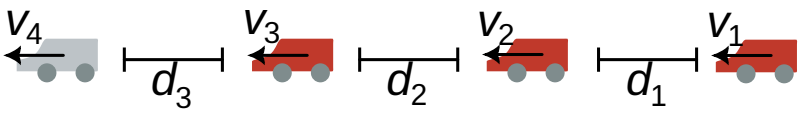
Shielding for infinite state spaces

Shielding for multi-agent systems

Summary

Multi-agent shields[1]

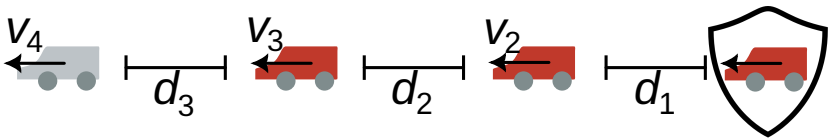
- Shield computation scales exponentially in number of agents
- Having a global shield is often undesired / unrealistic
- However, an individual shield that keeps an agent safe may not exist



[1] A. H. Brorholt, K. G. Larsen, and C. Schilling. *AAMAS*. 2025.

Multi-agent shields[1]

- Shield computation scales exponentially in number of agents
- Having a global shield is often undesired / unrealistic
- However, an individual shield that keeps an agent safe may not exist

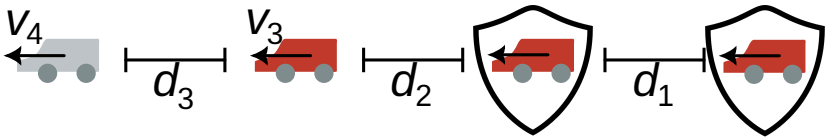


- Key idea: assume-guarantee reasoning for compositional shields

[1] A. H. Brorholt, K. G. Larsen, and C. Schilling. *AAMAS*. 2025.

Multi-agent shields[1]

- Shield computation scales exponentially in number of agents
- Having a global shield is often undesired / unrealistic
- However, an individual shield that keeps an agent safe may not exist

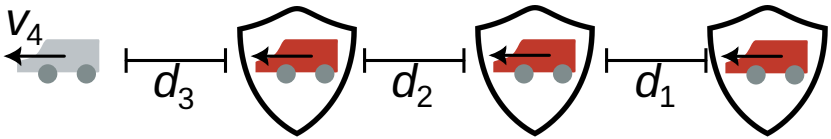


- Key idea: assume-guarantee reasoning for compositional shields

[1] A. H. Brorholt, K. G. Larsen, and C. Schilling. *AAMAS*. 2025.

Multi-agent shields[1]

- Shield computation scales exponentially in number of agents
- Having a global shield is often undesired / unrealistic
- However, an individual shield that keeps an agent safe may not exist



- Key idea: assume-guarantee reasoning for compositional shields

[1] A. H. Brorholt, K. G. Larsen, and C. Schilling. *AAMAS*. 2025.

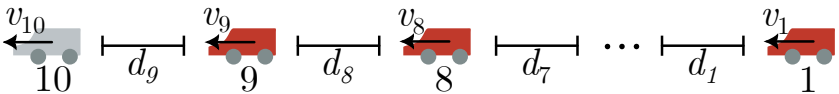
Distributed shielding[1]

- We want shields $\triangledown_i$ that ensure agent properties φ_i
- Let $\triangledown_1 \sqcap \triangledown_2$ denote the composition of shields $\triangledown_1$ and $\triangledown_2$
- Let $\langle\varphi_1\rangle\triangledown\langle\varphi_2\rangle$ denote "assuming φ_1 holds, $\triangledown$ will guarantee φ_2"
- Proof rule:

$$\frac{\langle\top\rangle\triangledown_1\langle\varphi_1\rangle \ \wedge \ \langle\varphi_1\rangle\triangledown_2\langle\varphi_2\rangle \ \wedge \ \ldots \ \wedge \ \langle\varphi_{n-1}\rangle\triangledown_n\langle\varphi_n\rangle}{\langle\top\rangle\triangledown_1 \sqcap \triangledown_2 \sqcap \ldots \sqcap \triangledown_n\langle\bigwedge_j \varphi_j\rangle}$$

[1] A. H. Brorholt, K. G. Larsen, and C. Schilling. *AAMAS*. 2025.

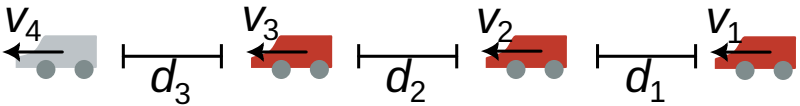
Evaluation – Shield computation



- Centralized shield: Timeout after 12 hours (2 controlled cars)
- Distributed shield: 6.5 seconds (9 controlled cars)

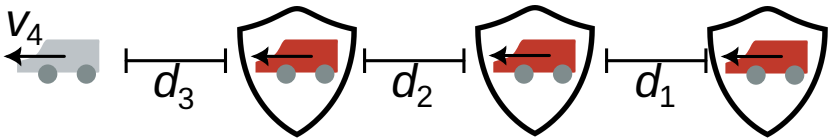
Multi-agent reinforcement learning (MARL)

- MARL: iterative update of all policies $\rightsquigarrow$ often diverges in practice



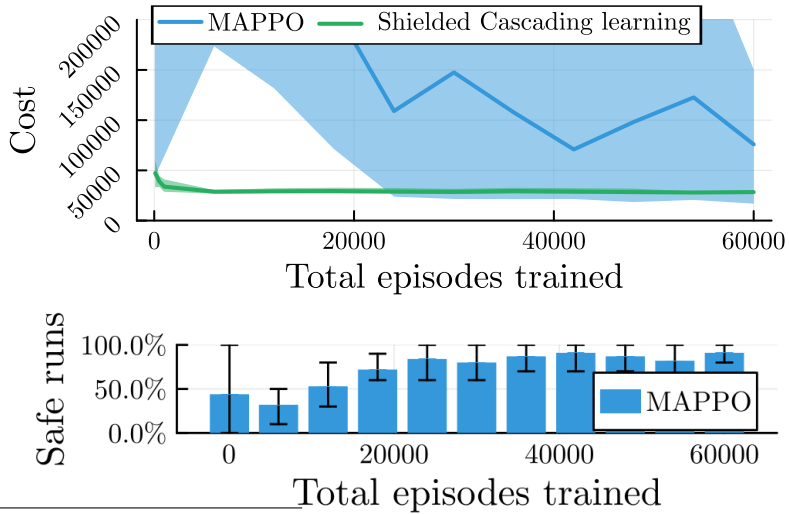
Multi-agent reinforcement learning (MARL)

- MARL: iterative update of all policies $\rightsquigarrow$ often diverges in practice



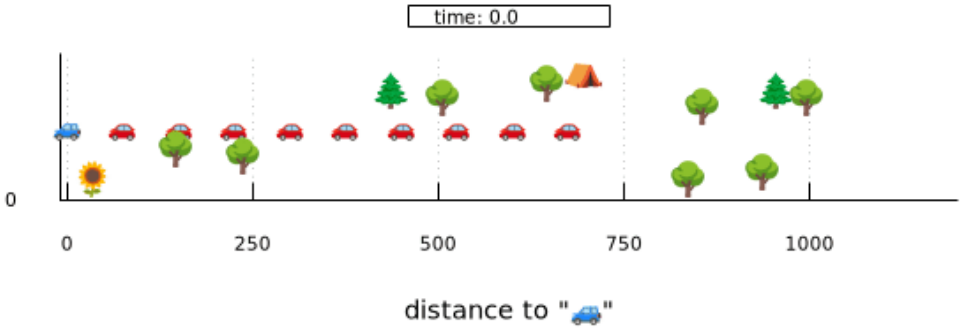
- Observation: distributed shields may break mutual dependencies
- Can now train agents individually $\rightsquigarrow$ may train faster and converge

Evaluation – Comparison to MAPPO[1]



[1] C. Yu et al. *NeurIPS*. 2022.

Learned shielded policies



Ongoing work – Circular dependencies

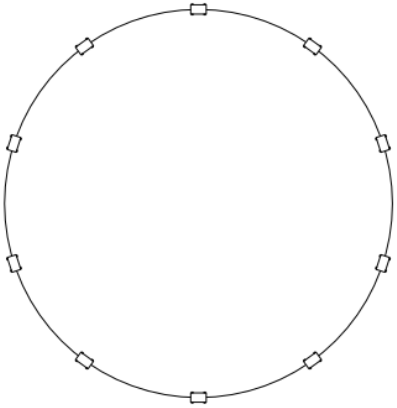


Table of contents

Motivation

Shielding

Shielding for finite state spaces

Shielding for infinite state spaces

Shielding for multi-agent systems

Summary

Conclusion

- Shielding for infinite-state systems with complex dynamics
  - Replace hard steps with simulation[1]
  - State-space transformation: more precise and efficient[2]
  - Integrated in the tool Uppaal Coshy[3]
- Multi-agent shielding: assume-guarantee reasoning[4]

Future work

- Cyclic dependencies
- Probabilistic shielding and online model learning
- Learning of suitable transformations and assumptions
- Combination with symbolic approach


Astrid


Kim


Florian


Peter


Andreas


Andrzej


Marius

[1] A. H. Brorholt et al. *AISoLA*. 2023.
[2] A. H. Brorholt et al. *AISoLA*. 2024.
[3] A. H. Brorholt et al. *RP*. 2025.
[4] A. H. Brorholt, K. G. Larsen, and C. Schilling. *AAMAS*. 2025.

Table of contents

Backup

Scalability

Grid size	Samples	Time
0.02	4	2m 14s
0.02	8	4m 02s
0.02	16	11m 03s
0.01	4	19m 00s
0.01	8	27m 21s
0.01	16	56m 32s

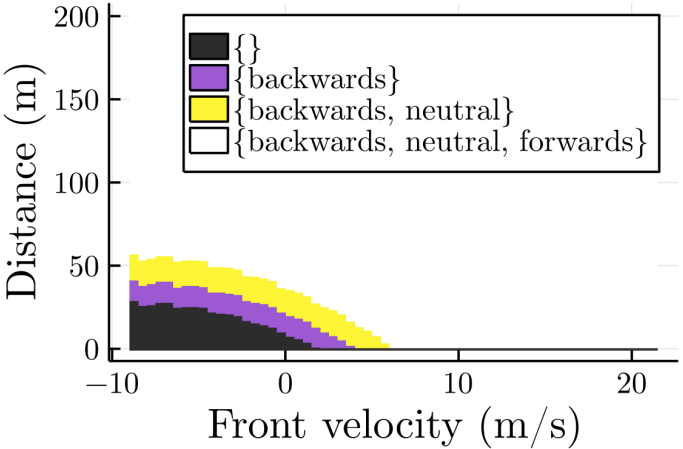
← Sampling-based

- Statistically safe ($\geq 99.99\,\%$ with confidence $99\,\%$)

Grid size	Time
0.01	41h 05m

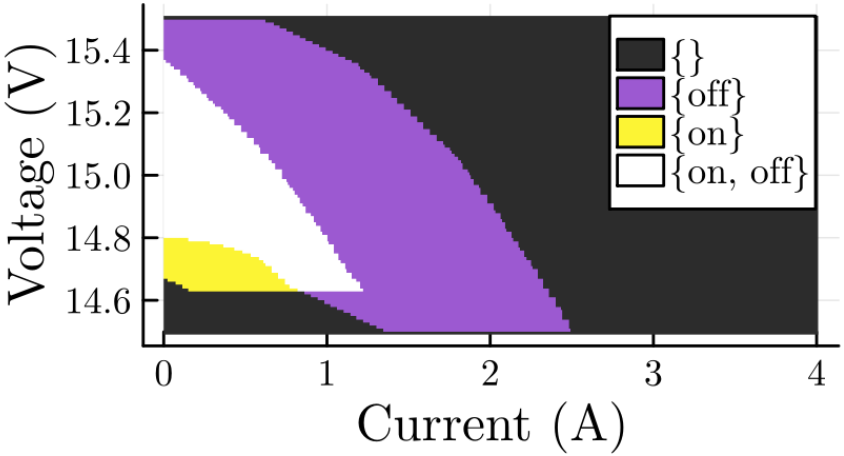
← Reachability-based

Cruise control (car behind another car)



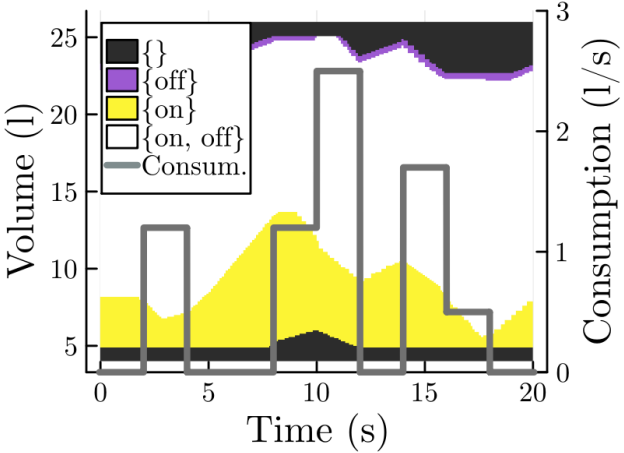
Synthesis time: 36 min

DC-to-DC boost converter



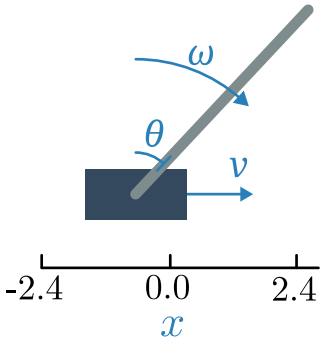
Synthesis time: 1 h 19 min

Oil pump (plot: pump is *on*)

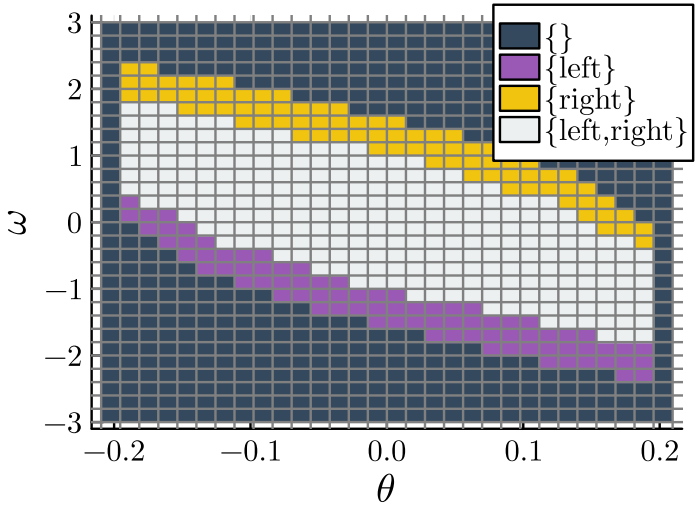


Synthesis time: 5 h 23 min

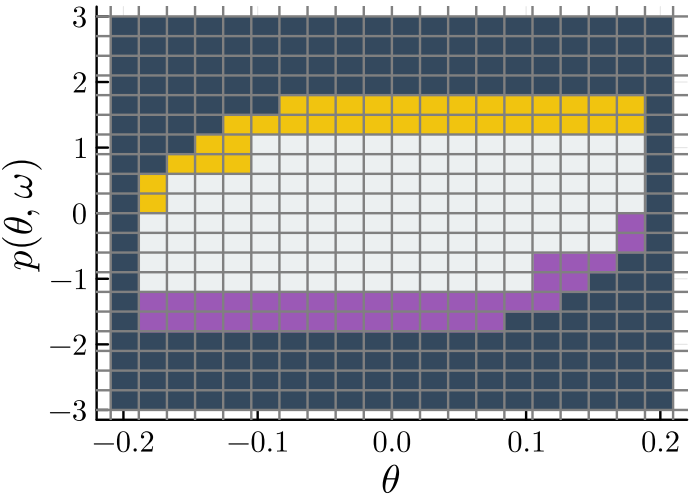
Cart-pole model



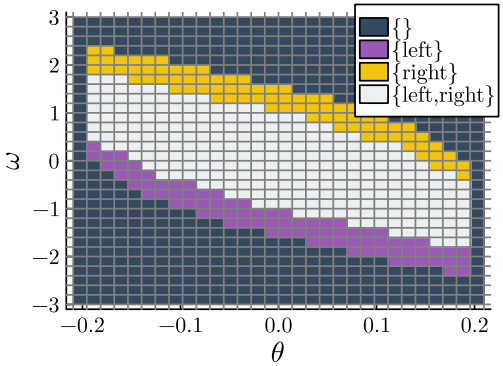
Cart-pole model: Shield



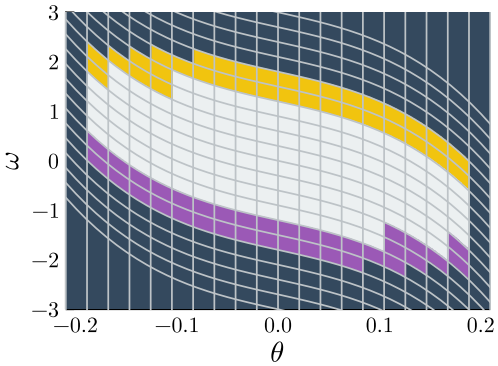
Cart-pole model: Transformed state space



Cart-pole model: Comparison



- 900 cells
- Computation: 0.2 sec
- Decision tree: 99 nodes



- 400 cells
- Computation: 0.5 sec
- Decision tree: 32 nodes

How we derived a transformation

- Goal: Grid in new state space captures decision boundaries, i.e., new boundaries are roughly axis-aligned
- Problem: We do not know the decision boundaries

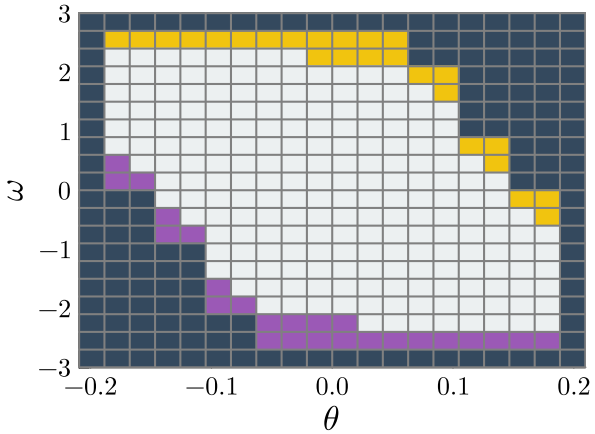
How we derived a transformation

- Goal: Grid in new state space captures decision boundaries, i.e., new boundaries are roughly axis-aligned
- Problem: We do not know the decision boundaries
- Step 1: Approximate the shape of the decision boundaries
- Step 2: Craft a suitable transformation

How we approximated the decision boundaries

- Use a coarse grid (here: 20×20 cells)
- Fixpoint: All cells in the avoid set
- Consequence of the abstraction and spurious transitions
- Spuriousness of transitions grows with number of steps
- Idea: Perform fixpoint iteration for only a few (here: 3) steps

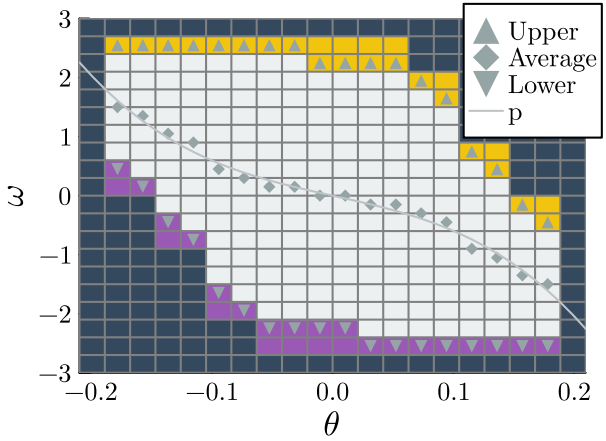
How we approximated the decision boundaries



How we approximated the decision boundaries

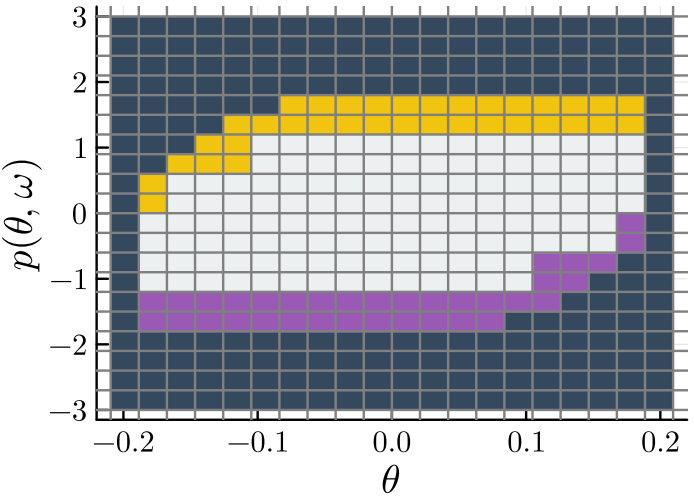
- Dimension θ (x-axis) is already well-aligned
- Goal: Replace dimension ω (y-axis) by a new expression
- We fit a third-degree polynomial to the "average" of the upper and lower bounds

How we approximated the decision boundaries

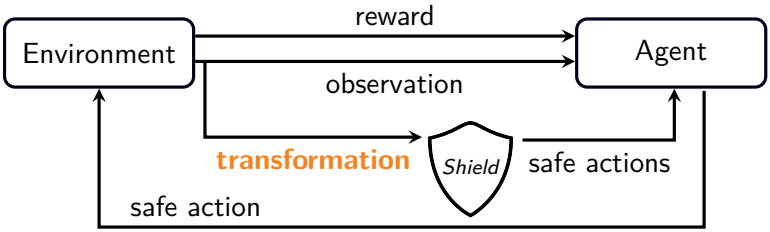


$$p(\theta) = -141.6953 \cdot \theta^3 - 4.5508 \cdot \theta$$

How we approximated the decision boundaries



Impact on learning



- We evaluate the cumulative return in six scenarios
- Three shield variants: no shield, shield in S, shield in T
- Two learning variants: learn in S, learn in T
- (The diagram shows the variant *shield in T, learn in S)*

Impact on learning

Learning	**Satellite** (↗)			**Bouncing ball** (↘)		
	None	S	T	None	S	T
S	1.123	0.786	**1.499**	39.897	37.607	**36.593**
T	0.917	0.889	**1.176**	39.128	40.024	**39.099**

Learning	**Cart/pole** (↘)		
	None	S	T
S	0.007	0.019	**0.001**
T	**0.000**	**0.000**	**0.000**