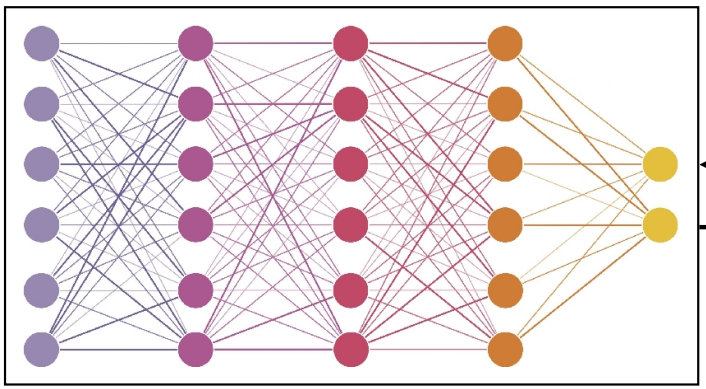
To safe AI control systems in three steps

Christian Schilling

DANMARKS FRIE FORSKNINGSFOND
INDEPENDENT RESEARCH FUND DENMARK

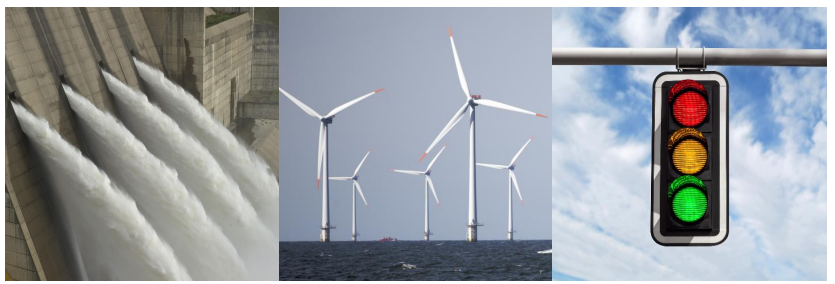
DIREC
Digital Research Centre Denmark

AALBORG UNIVERSITY



system state

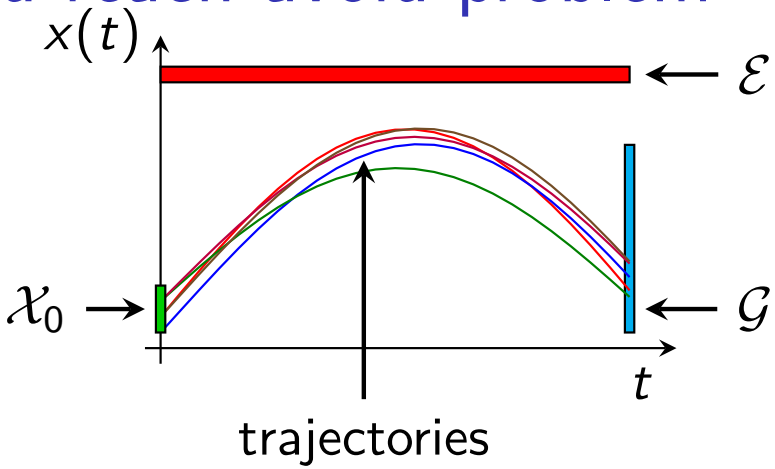
control action



Neural-network controller

Environment

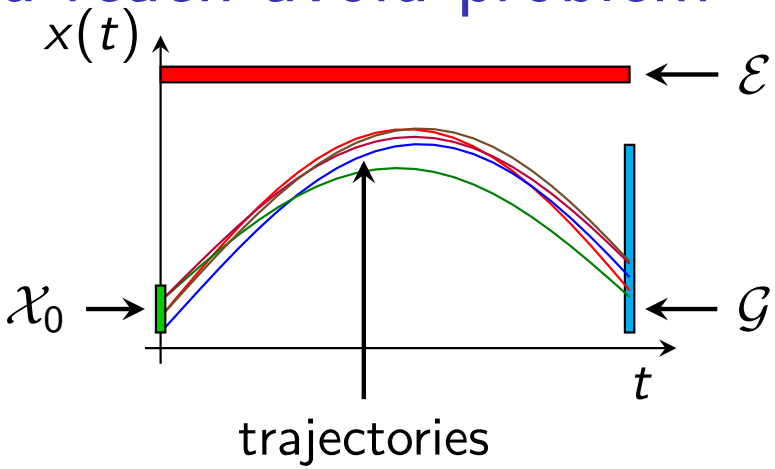
Reach-avoid specification and reach-avoid problem



Reach-avoid specification:

- Given:
  - Set of **initial states** $\mathcal{X}_0 \subseteq \mathbb{R}^n$
  - Set of **goal states** $\mathcal{G} \subseteq \mathbb{R}^n$
  - Set of **error states** $\mathcal{E} \subseteq \mathbb{R}^n$
  - Time bound T
- Aim: starting at $\mathcal{X}_0$, reach $\mathcal{G}$ within time T while avoiding $\mathcal{E}$
- Covers many real-world scientific and engineering problems

Reach-avoid specification and reach-avoid problem



Reach-avoid specification:

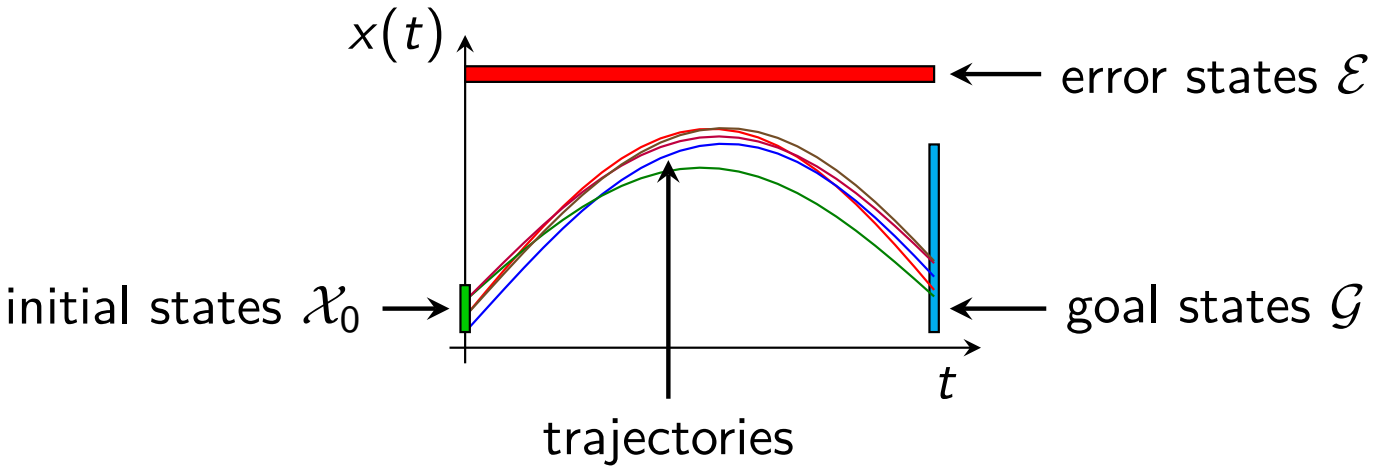
- Given:
  - Set of **initial states** $\mathcal{X}_0 \subseteq \mathbb{R}^n$
  - Set of **goal states** $\mathcal{G} \subseteq \mathbb{R}^n$
  - Set of **error states** $\mathcal{E} \subseteq \mathbb{R}^n$
  - Time bound T
- Aim: starting at $\mathcal{X}_0$, reach $\mathcal{G}$ within time T while avoiding $\mathcal{E}$
- Covers many real-world scientific and engineering problems

Reach-avoid problem:

- Does a given system satisfy a reach-avoid specification?
- **Undecidable** for nonlinear dynamics

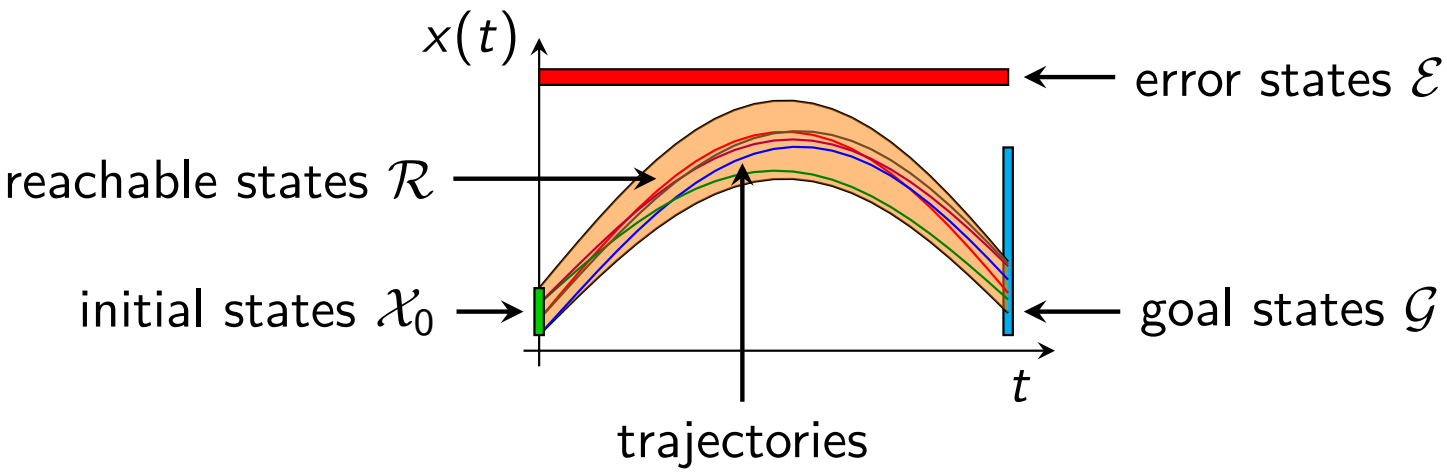
Solution strategy for the reach-avoid problem

- Verify: all trajectories lead to **goal states** avoiding **error states**



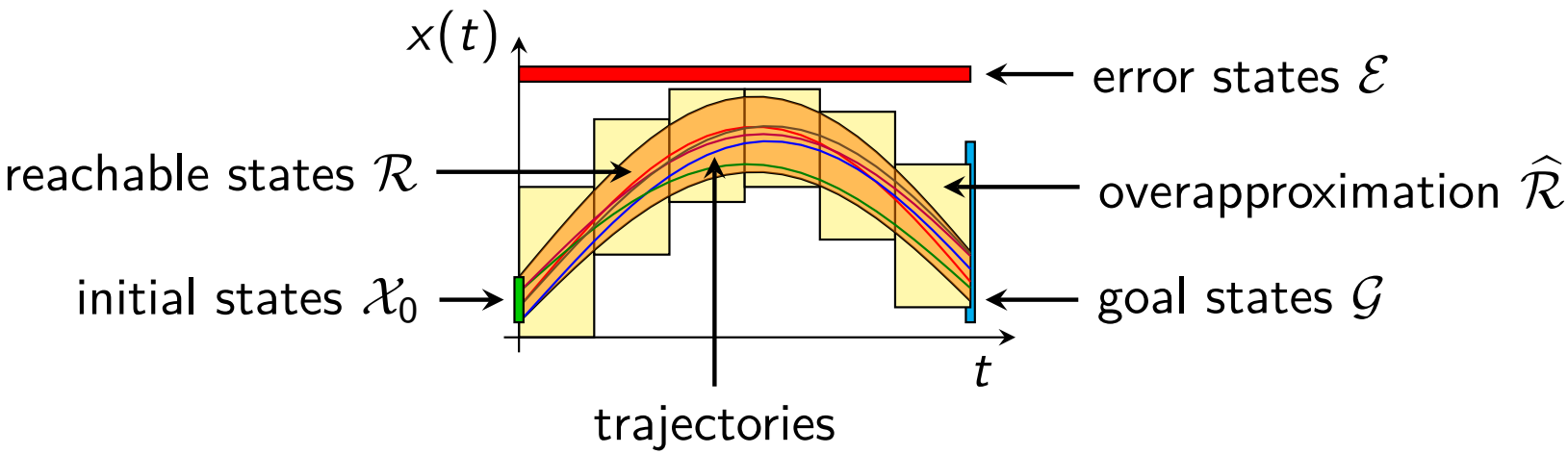
Solution strategy for the reach-avoid problem

- Verify: all trajectories lead to **goal states** avoiding **error states**
- Equivalent to computing the reachable states $\mathcal{R}$ and proving $\mathcal{R} \cap \mathcal{E} = \emptyset \wedge \mathcal{R}_T \subseteq \mathcal{G}$
- $\mathcal{R}$ only computable under strong restrictions



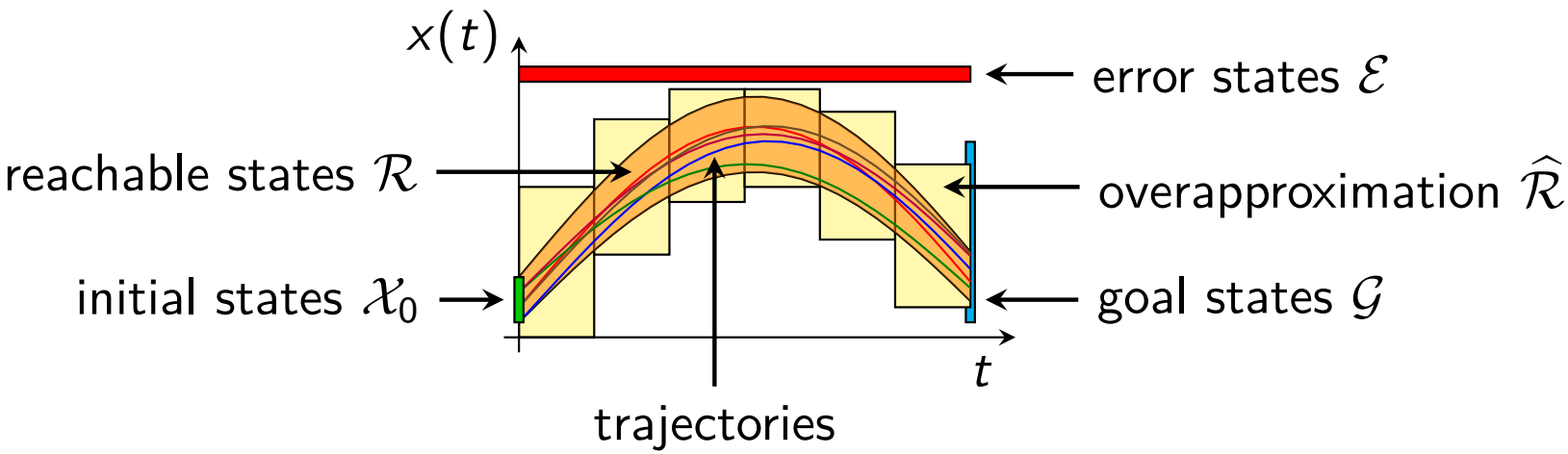
Solution strategy for the reach-avoid problem

- Verify: all trajectories lead to **goal states** avoiding **error states**
- Equivalent to computing the reachable states $\mathcal{R}$ and proving $\mathcal{R} \cap \mathcal{E} = \emptyset \wedge \mathcal{R}_T \subseteq \mathcal{G}$
- $\mathcal{R}$ only computable under strong restrictions
- Proving $\widehat{\mathcal{R}} \cap \mathcal{E} = \emptyset \wedge \widehat{\mathcal{R}}_T \subseteq \mathcal{G}$ is sufficient, where $\widehat{\mathcal{R}}$ is an overapproximation of $\mathcal{R}$

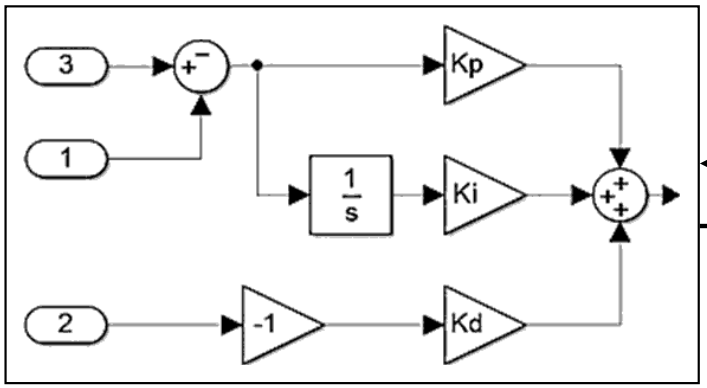


Solution strategy for the reach-avoid problem

- Verify: all trajectories lead to **goal states** avoiding **error states**
- Equivalent to computing the reachable states $\mathcal{R}$ and proving $\mathcal{R} \cap \mathcal{E} = \emptyset \wedge \mathcal{R}_T \subseteq \mathcal{G}$
- $\mathcal{R}$ only computable under strong restrictions
- Proving $\widehat{\mathcal{R}} \cap \mathcal{E} = \emptyset \wedge \widehat{\mathcal{R}}_T \subseteq \mathcal{G}$ is sufficient, where $\widehat{\mathcal{R}}$ is an overapproximation of $\mathcal{R}$
- Challenge in practice: trade-off between **precision** and **scalability**



Research problems for control systems



system state

control action

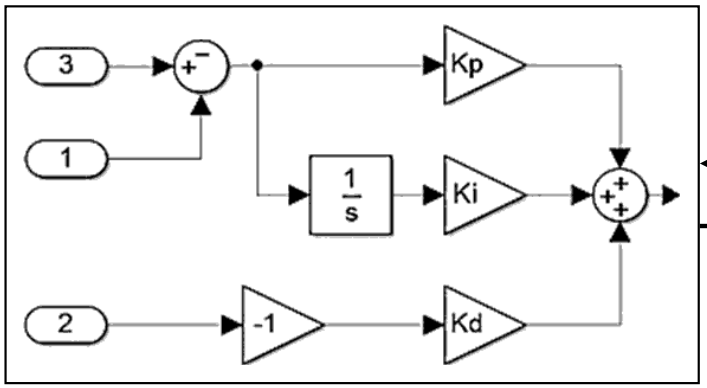
Controller

Environment

Reach-avoid problem:

- Does a given controller satisfy a reach-avoid specification?
- **Undecidable** for nonlinear dynamics
- Need ways to compute the reachable states $\mathcal{R}$ (resp. an overapproximation $\widehat{\mathcal{R}}$) for both the controller and the environment

Research problems for control systems



system state

control action

Controller

Environment

Reach-avoid problem:

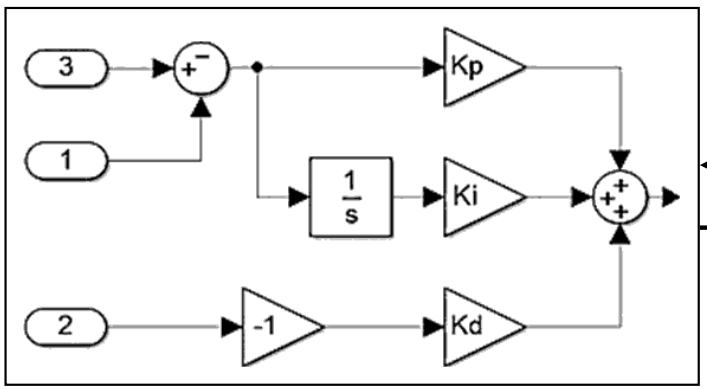
- Does a given controller satisfy a reach-avoid specification?
- **Undecidable** for nonlinear dynamics
- Need ways to compute the reachable states $\mathcal{R}$ (resp. an overapproximation $\widehat{\mathcal{R}}$) for both the controller and the environment

Synthesis problem:

- Find a controller that satisfies a reach-avoid specification
- **Even harder** problem

AI control systems

- Traditionally, the controller is designed by engineers



Controller

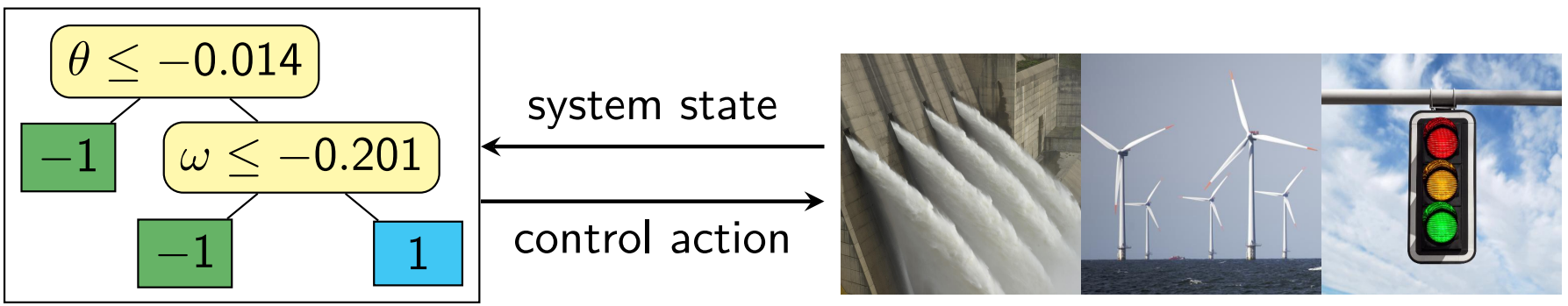
system state

control action

Environment

AI control systems

- Traditionally, the controller is designed by engineers
- Now we can machine-learn **high performance controllers**

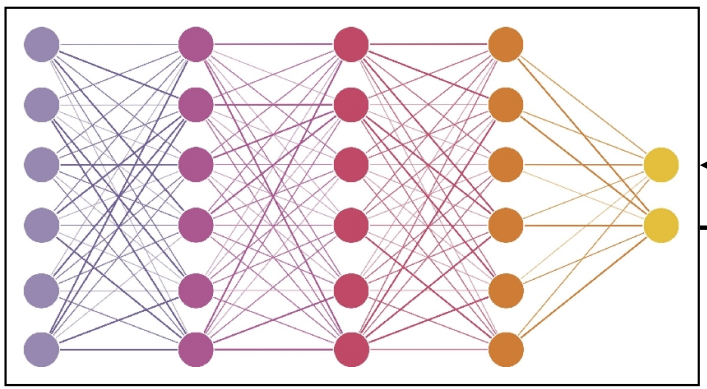


Decision-tree controller

Environment

AI control systems

- Traditionally, the controller is designed by engineers
- Now we can machine-learn **high performance controllers**



Neural-network controller

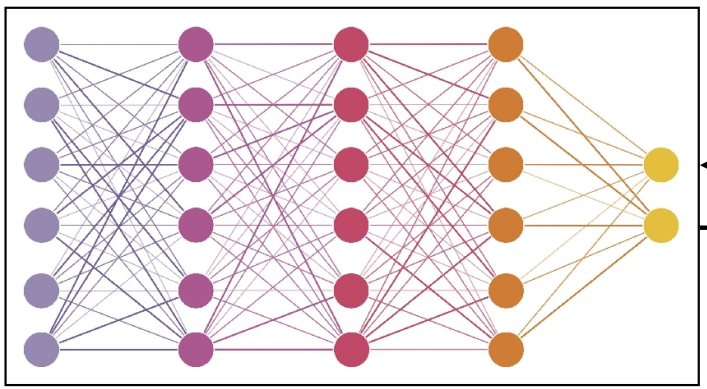
system state

control action

Environment

AI control systems

- Traditionally, the controller is designed by engineers
- Now we can machine-learn **high performance controllers**
- **No safety guarantee** and often **intransparent** ("black box")



Neural-network controller

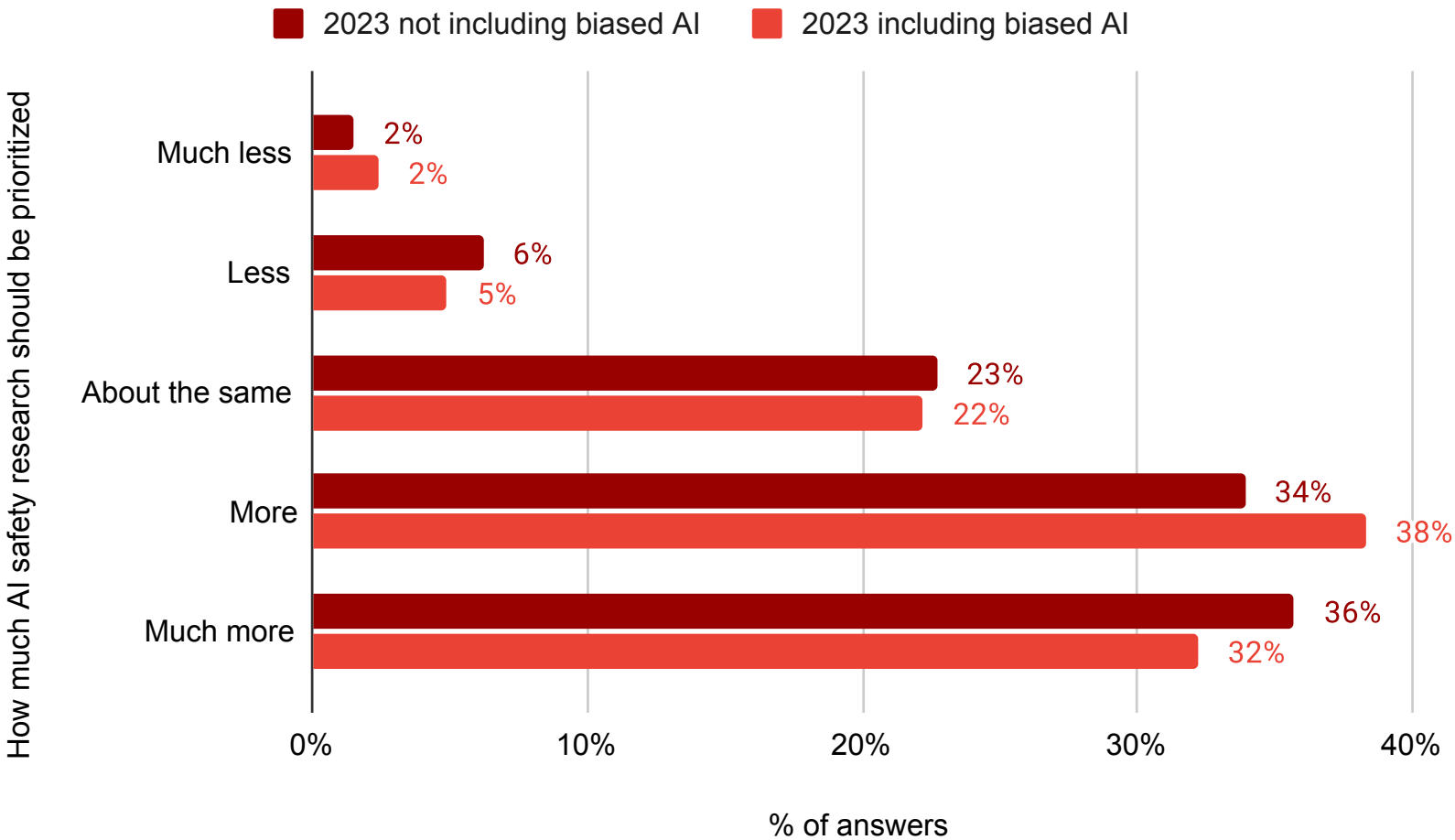
system state

control action

Environment

AI safety[1]

"How much should AI safety research be prioritized?"



[1] Grace et al. (2023). Thousands of AI authors on the future of AI.

Example: Unicycle model

Environment:

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{v} = u_1 + w$$
$$\dot{\theta} = u_2$$

Example: Unicycle model

Environment:

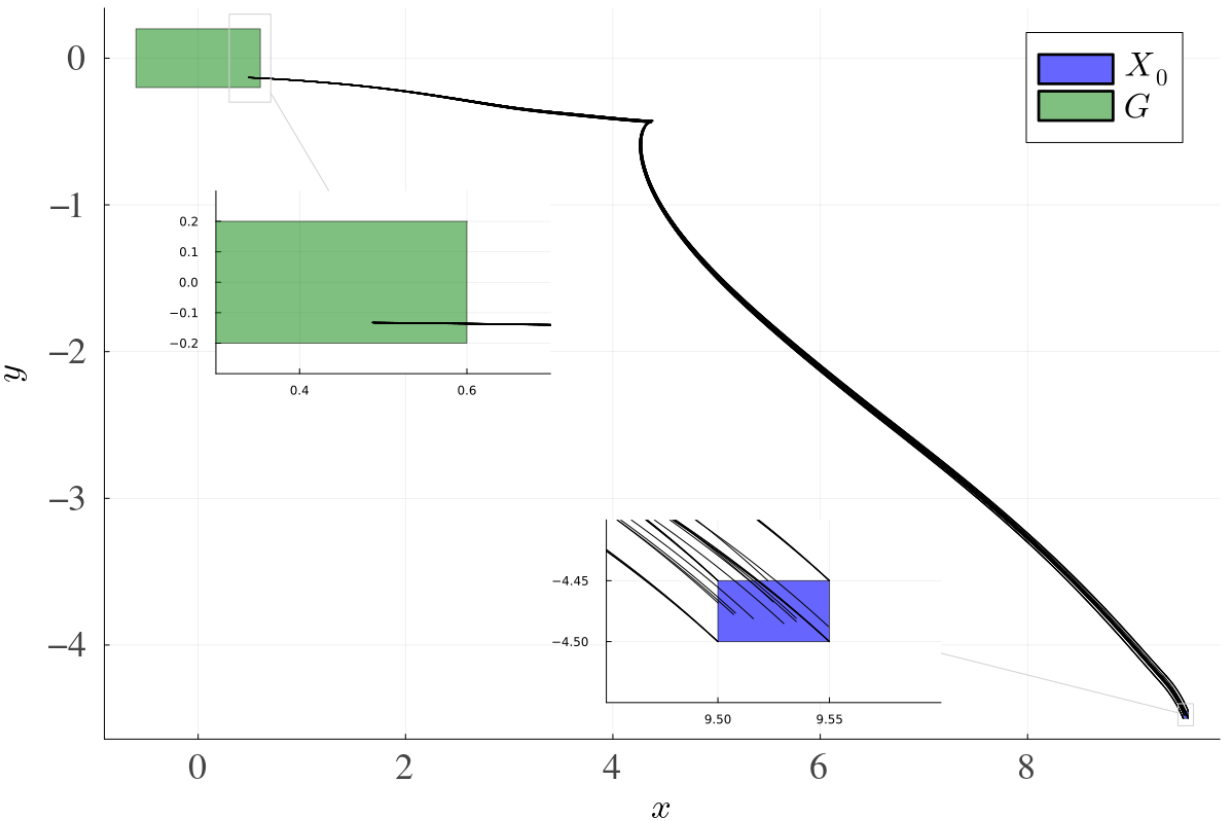
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{v} = u_1 + w$
$\dot{\theta} = u_2$

Specification:

$x(0) \in \mathcal{X}_0$
$x(10) \overset{!}{\in} \mathcal{G}$

Controller:

neural network
500 hidden units
$\tau = 0.2$



42 simulations

Example: Unicycle model

Environment:

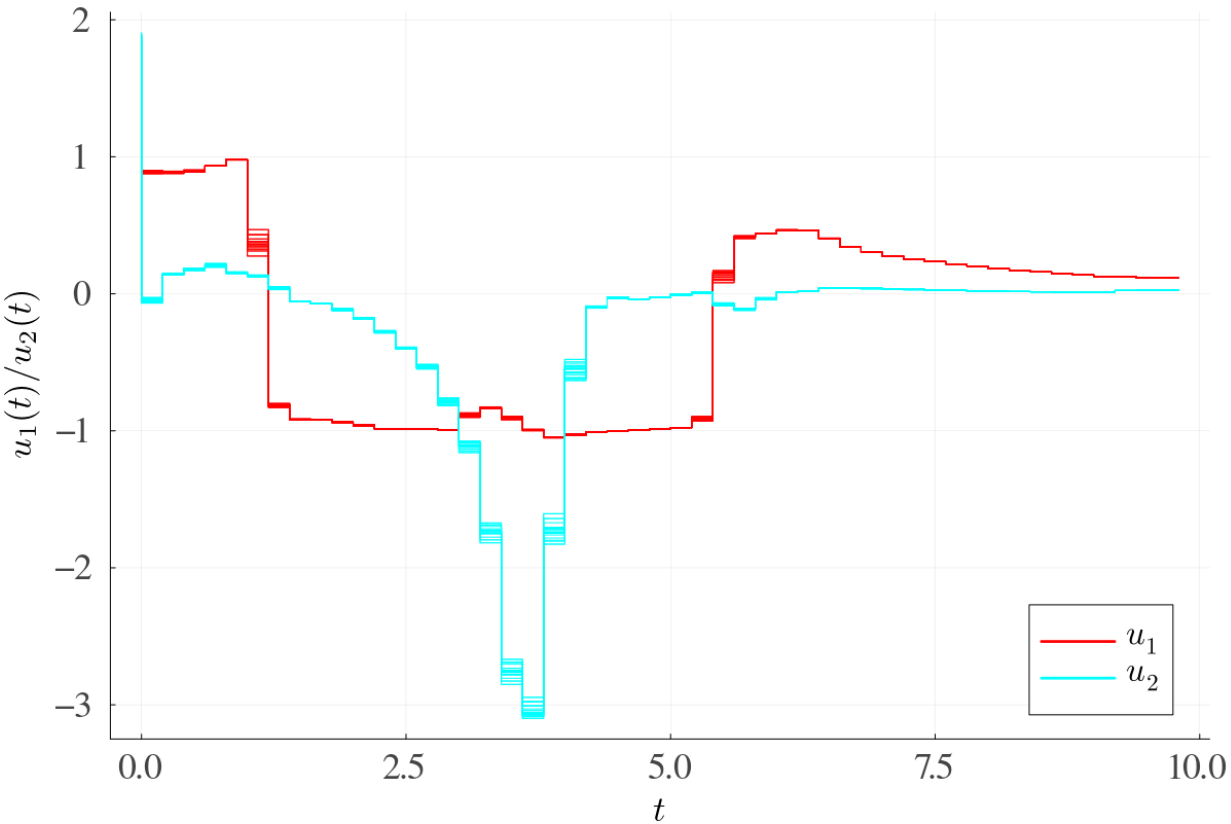
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{v} = u_1 + w$
$\dot{\theta} = u_2$

Specification:

$x(0) \in \mathcal{X}_0$
$x(10) \overset{!}{\in} \mathcal{G}$

Controller:

neural network
500 hidden units
$\tau = 0.2$



control signals (42 simulations)

Example: Unicycle model

Environment:

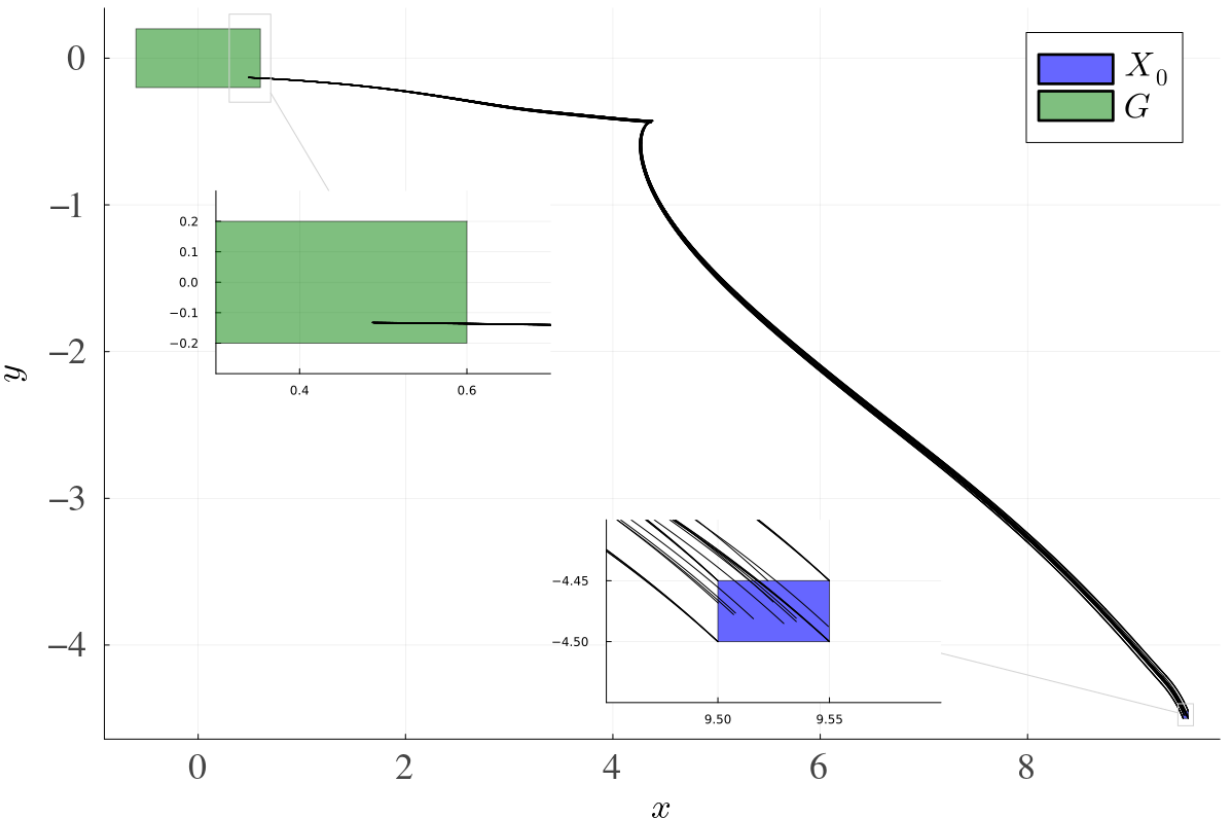
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{v} = u_1 + w$
$\dot{\theta} = u_2$

Specification:

$x(0) \in \mathcal{X}_0$
$x(10) \overset{!}{\in} \mathcal{G}$

Controller:

neural network
500 hidden units
$\tau = 0.2$



42 simulations

Example: Unicycle model

Environment:

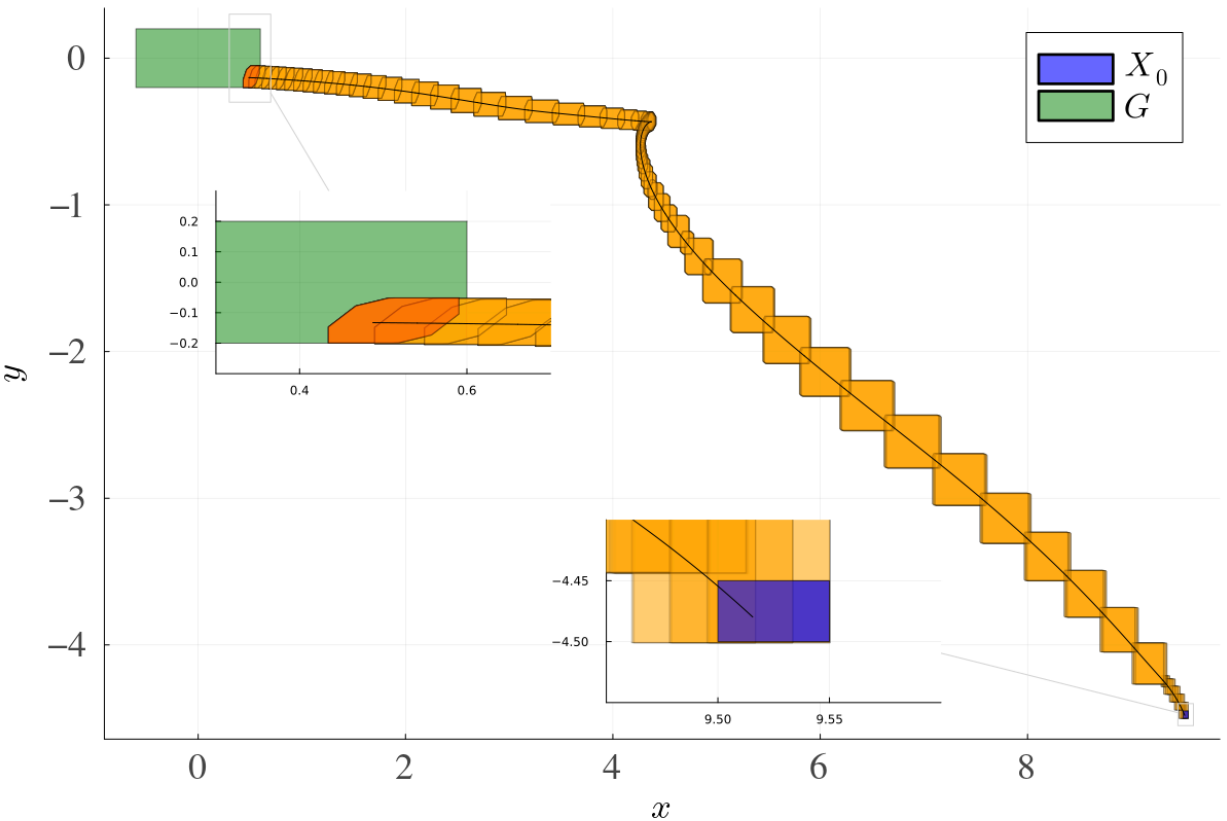
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{v} = u_1 + w$
$\dot{\theta} = u_2$

Specification:

$x(0) \in \mathcal{X}_0$
$x(10) \overset{!}{\in} \mathcal{G}$

Controller:

neural network
500 hidden units
$\tau = 0.2$

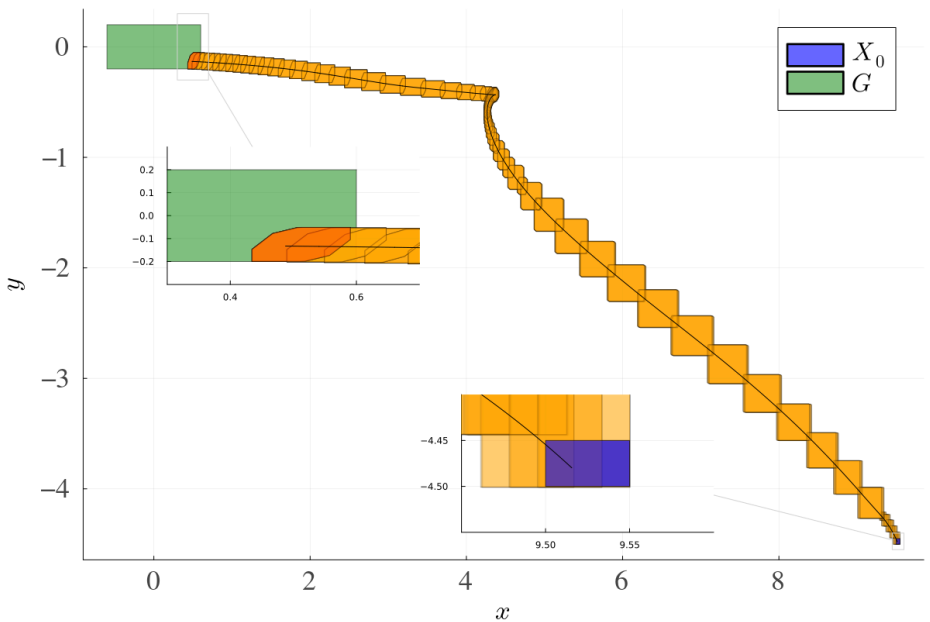


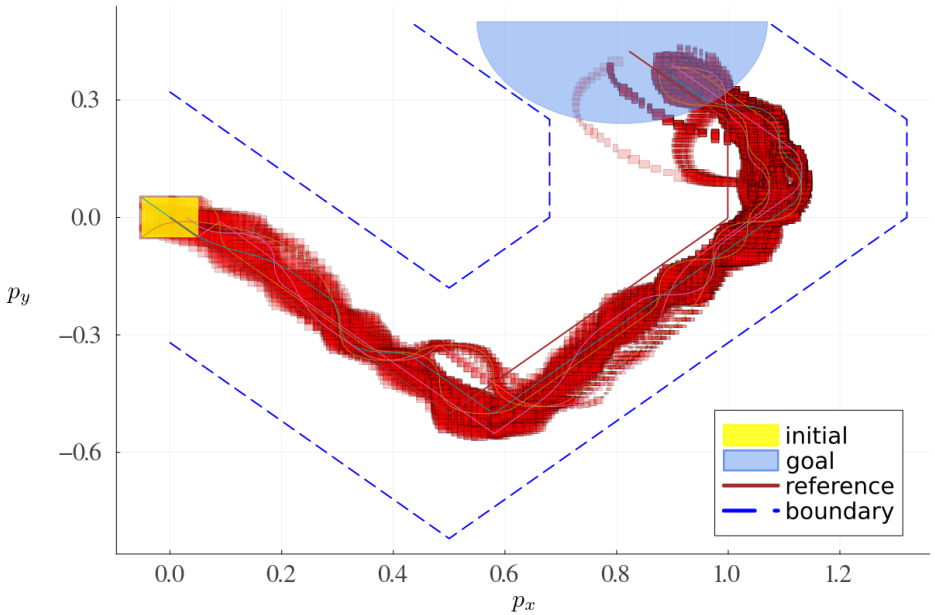
Overapproximated reachable states

To safe AI control systems in three steps

Step 1: Verification

- **Goal**: Given a controller, show that it **satisfies** the specification
- Focus: Precise, scalable, and quantitative **verification** techniques

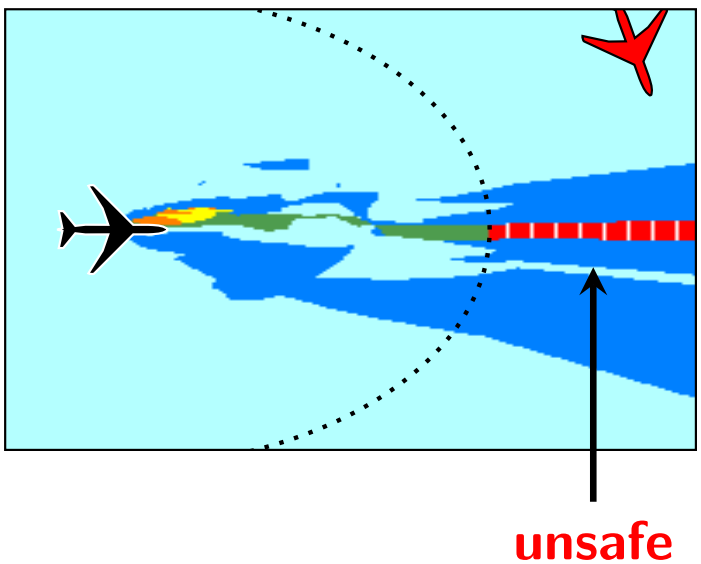




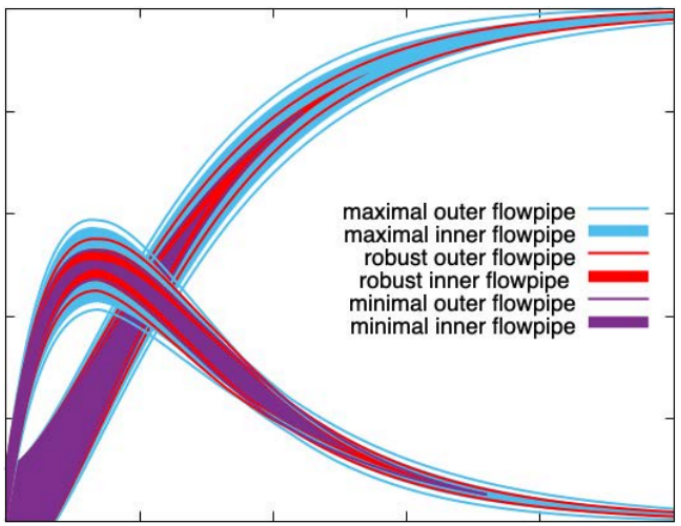
S, Forets, and Guadalupe. *AAAI*. 2022.
Kochdumper, S, Althoff, and Bak. *NASA Formal Methods*. 2023.
S, Lukina, Demirović, and Larsen. *NeurIPS*. 2023.

Step 2: Falsification

- **Goal**: Given a controller, show that it **violates** the specification
- Focus: Targeted **falsification** techniques



Constrained optimization



Underapproximated reach tube

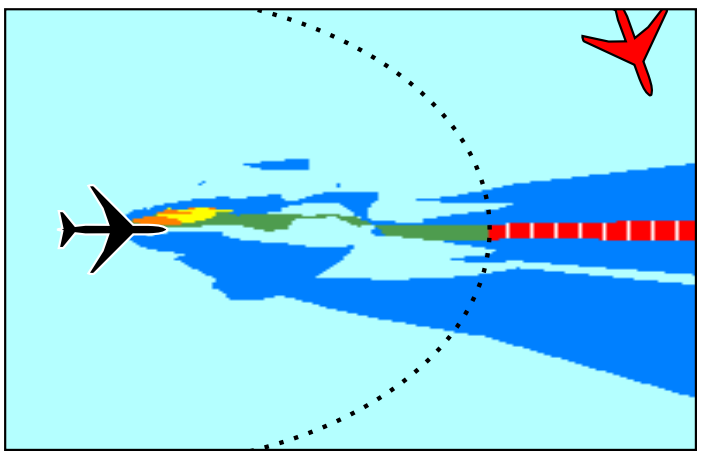
Bauer-Marquart, Boetius, Leue, and S. *SPIN*. 2022.
Goubault and Putot. *HSCC*. 2019.

Step 3: Repair / Synthesis

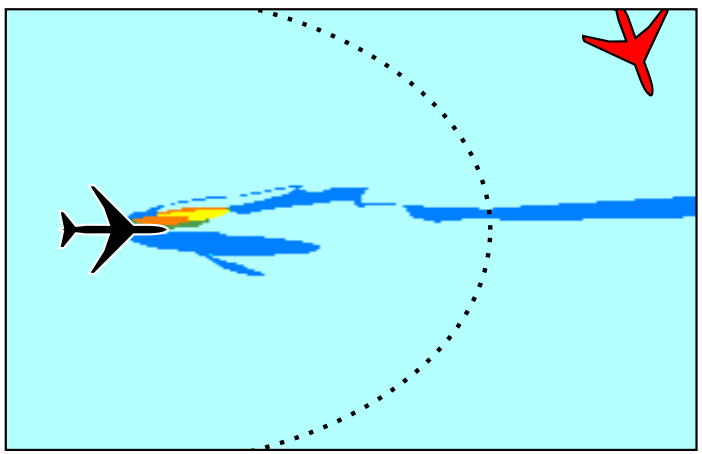
- **Goal**: Given a **violating** controller, make it **satisfy** the specification

  falsify $\rightsquigarrow$ **repair** $\rightsquigarrow$ **verify**

- Focus: **Repair** techniques for high controller performance



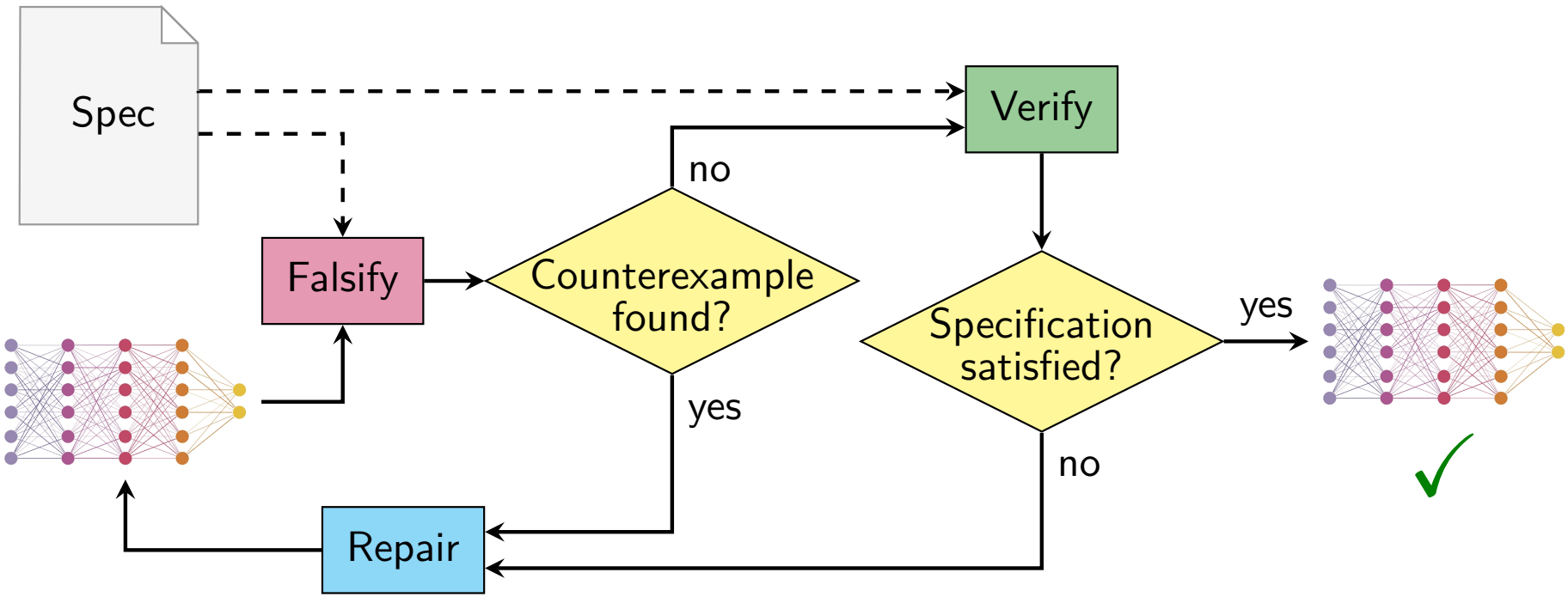
Before repair



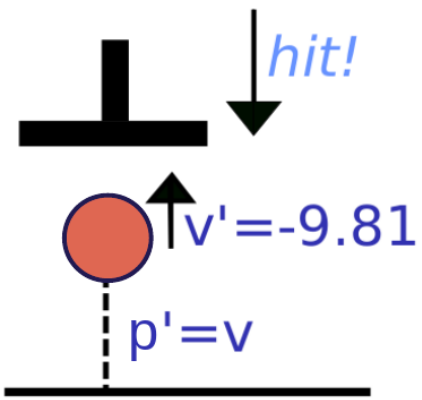
After repair

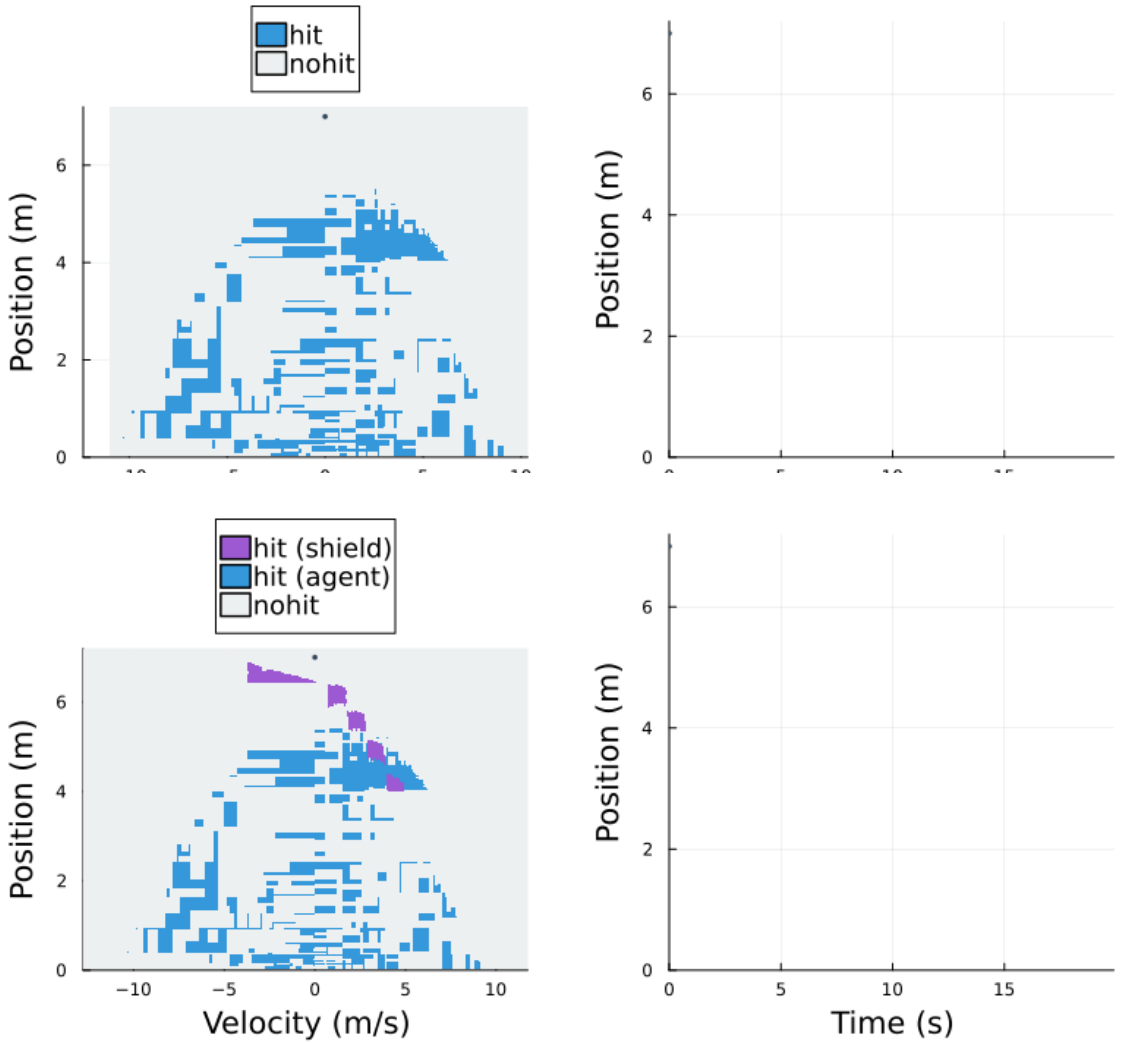
Bauer-Marquart, Boetius, Leue, and S. *SPIN*. 2022.
Vergari, Choi, Liu, Teso, and Van den Broeck. *NeurIPS*. 2021.

Repair loop guided by counterexamples



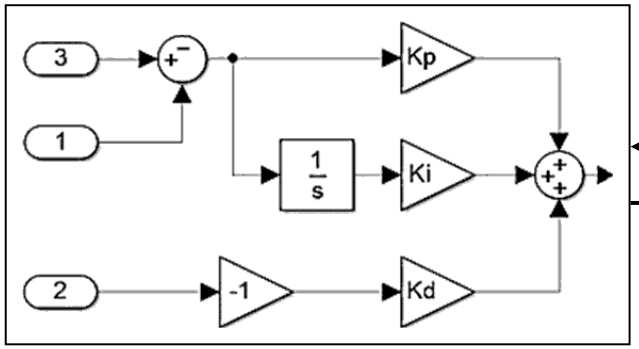
Learned controller guarded by synthesized shield

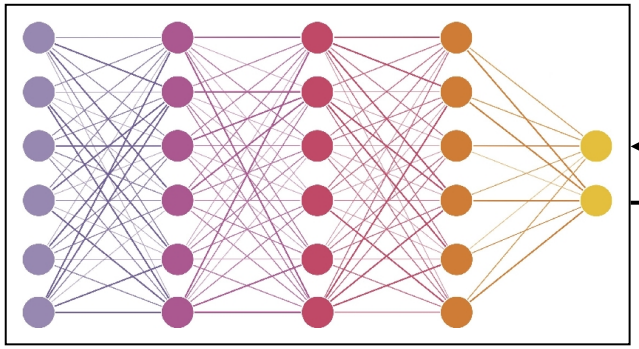




Brorholt, Jensen, Larsen, Lorber, and S. *AISoLA*. 2023.

Vision



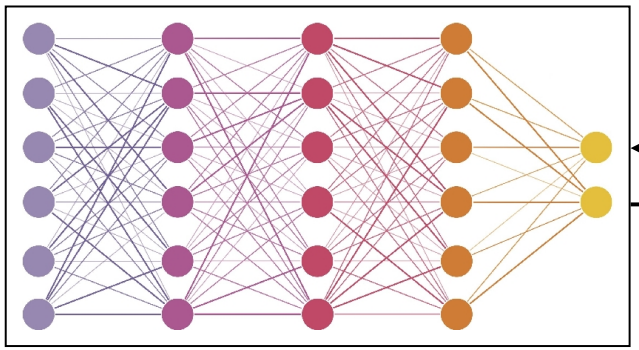


Performance ↯

Correctness ✓

Performance ✓

Correctness ↯



Performance ✓

Correctness ✓