Verification of AI-controlled systems

Christian Schilling

Workshop on Verifiable and Robust AI 2023

AALBORG
UNIVERSITY

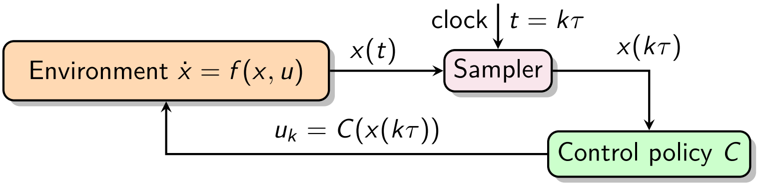
Overview

Overview

Control system

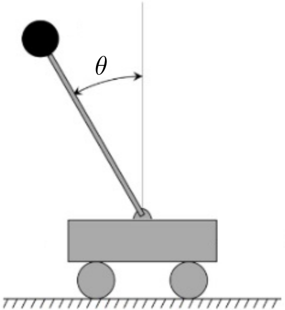
A (continuous-time) **control system** is a triple (f, C, τ) with

- f: **environment** $\dot{x} = f(x, u) : \mathbb{R}^{n+m} \to \mathbb{R}^n$
- C: **control policy** $C : \mathbb{R}^n \to \mathbb{R}^m$
- τ: **control period** $\tau \in \mathbb{R}^+$

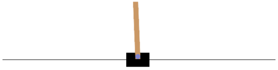


Example: Stabilizing a pole on a cart

- Continuous time



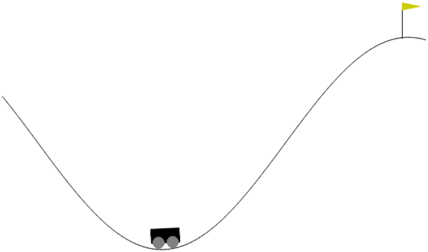
$$\dot{p} = v$$
$$\dot{\theta} = \omega$$
$$\dot{\omega} = \phi$$
$$\dot{v} = \psi - \frac{1}{22}\phi\cos(\theta)$$
$$\phi = \frac{9.8\sin(\theta) - \cos(\theta)\psi}{2/3 + 5/11\cos(\theta)^2}$$
$$\psi = \frac{10\boldsymbol{u} + 0.05\omega^2\sin(\theta)}{1.1}$$
$$\tau = 0.02$$



Example: Car reaching the top of a mountain

- Discrete time

$$v_{k+1} = v_k + (\mathbf{u} - 1)F - \cos(3p_k)g$$
$$p_{k+1} = p_k + v_{k+1}$$



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
- "Starting at $\mathcal{X}_0$, reach $\mathcal{G}$ before T while avoiding $\mathcal{E}$"

$$\mathcal{X}_0 \to \neg\mathcal{E} \;\; \mathcal{U}^T \; \mathcal{G}$$

Reach-avoid problem:

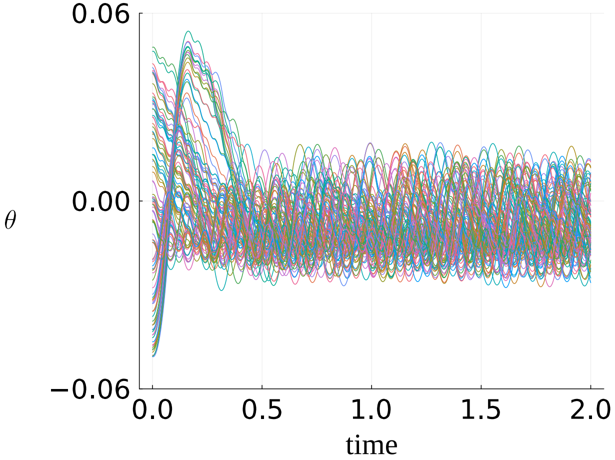
- Does a given control policy C satisfy a reach-avoid specification?
- Reduces to computing **reachable states** of control system (f, C, τ)
- **Undecidable** for nonlinear environments $f(x, u)$, even with fixed u

Synthesis problem:

- Find a control policy C that satisfies a reach-avoid specification

Example: Cart/pole system

Example: $p_0, v_0, \theta_0, \omega_0 \in [-0.05, 0.05]$ and $T = 2$



Overview

Problem

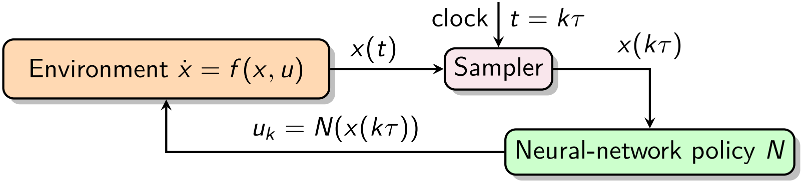
Neural-network controllers

Decision-tree controllers

Conclusion

Neural-network control system[1][2]

A **neural-network control system** (DTCS) uses a neural-network policy $N : \mathbb{R}^n \to \mathbb{R}^m$



[1] Schilling, Forets, and Guadalupe. *AAAI*. 2022.
[2] Kochdumper, Schilling, Althoff, and Bak. *NASA Formal Methods*. 2023.

Unicycle model

Environment:

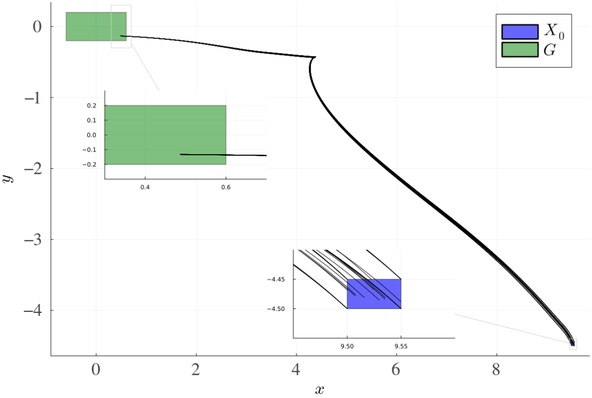
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{\theta} = u_2$
$\dot{v} = u_1 + w$

Specification:

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

Controller:

500 hidden units
$\tau = 0.2$



42 simulations

Unicycle model

Environment:

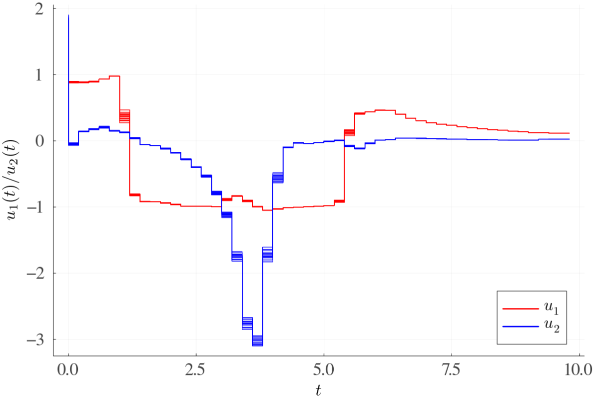
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{\theta} = \boldsymbol{u_2}$
$\dot{v} = \boldsymbol{u_1} + w$

Specification:

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

Controller:

500 hidden units
$\tau = 0.2$



control signals (42 simulations)

Unicycle model

Environment:

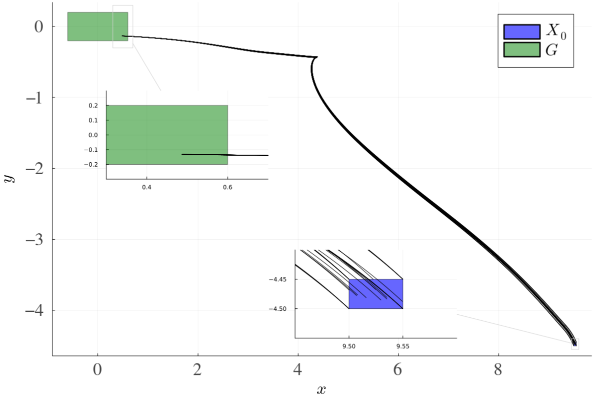
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{\theta} = \mathbf{u_2}$
$\dot{v} = \mathbf{u_1} + w$

Specification:

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

Controller:

500 hidden units
$\tau = 0.2$



42 simulations

Unicycle model

Environment:

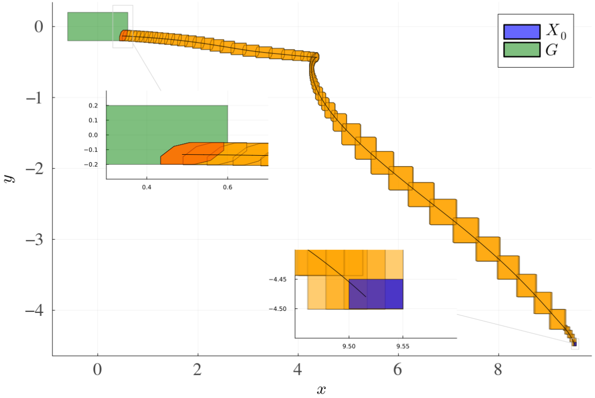
$\dot{x} = v\cos(\theta)$
$\dot{y} = v\sin(\theta)$
$\dot{\theta} = u_2$
$\dot{v} = u_1 + w$

Specification:

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

Controller:

500 hidden units
$\tau = 0.2$



Set-based simulations/reachability analysis

Two reachability algorithms

Reachability algorithm 1[1]

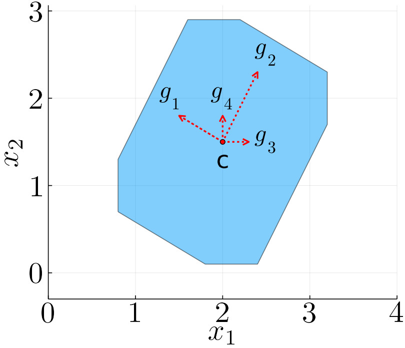
- Combination of **Taylor models**[2] and **zonotopes**[3]
- Implemented in **JuliaReach**[4]

Reachability algorithm 2[5]

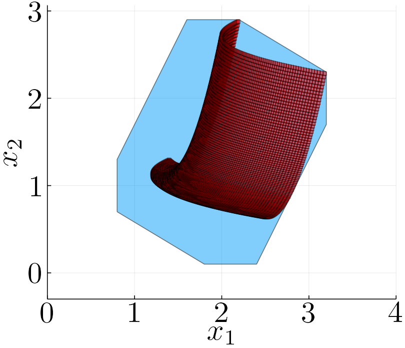
- **Polynomial zonotopes**[6]
- Implemented in **CORA**[7]

[1] Schilling, Forets, and Guadalupe. *AAAI*. 2022.
[2] Makino and Berz. *Int. J. Pure Appl. Math* (2003).
[3] Singh, Gehr, Mirman, Püschel, and Vechev. *NeurIPS*. 2018.
[4] Bogomolov, Forets, Frehse, Potomkin, and Schilling. *HSCC*. 2019.
[5] Kochdumper, Schilling, Althoff, and Bak. *NASA Formal Methods*. 2023.
[6] Kochdumper and Althoff. *IEEE Trans. Autom. Control*. (2021).
[7] Althoff. *ARCH*. 2015.

Taylor model and structured zonotope

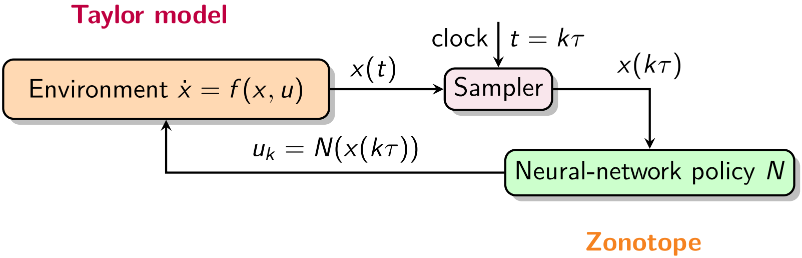


Structured zonotope

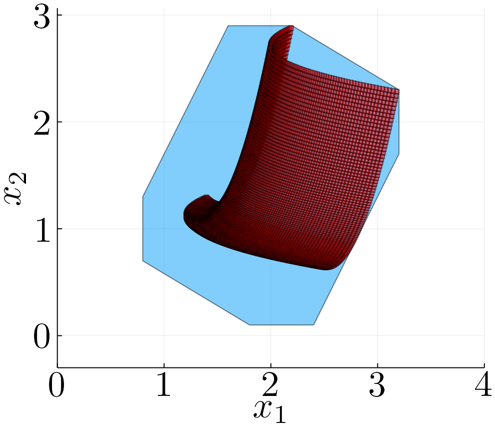


Taylor model enclosed by **structured zonotope**

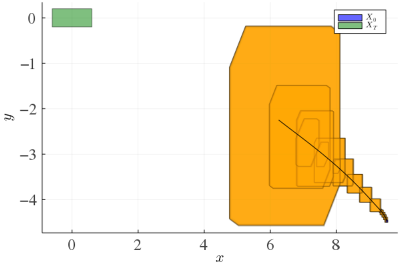
Combining Taylor models with zonotopes



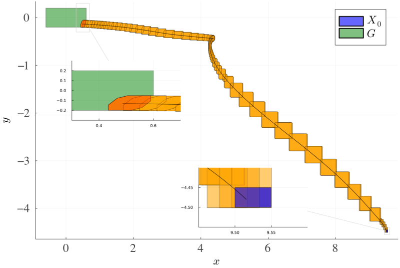
Combining Taylor models with zonotopes



Combining Taylor models with zonotopes



Naive combination



Careful combination

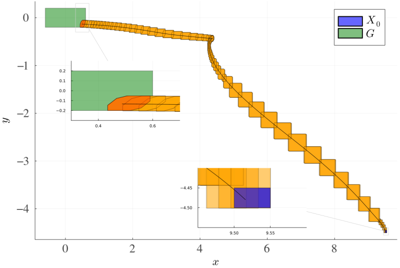
Evaluation

- Benchmark problems from **ARCH-COMP**[1]
  - **JuliaReach** was the first tool that solved all problems
- Comparison with **Sherlock**[2]
  - Taylor-model techniques for environment
  - Quadratic approximation for neural network
  - No set conversion (Taylor model end-to-end)

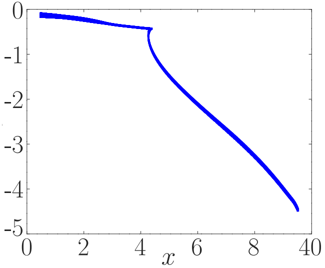
[1] Johnson et al. *ARCH*. 2021.

[2] Dutta, Chen, and Sankaranarayanan. *HSCC*. 2019.

Unicycle car



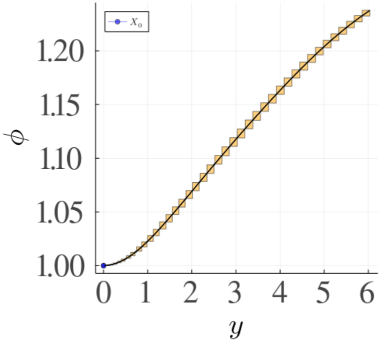
JuliaReach, 93 seconds



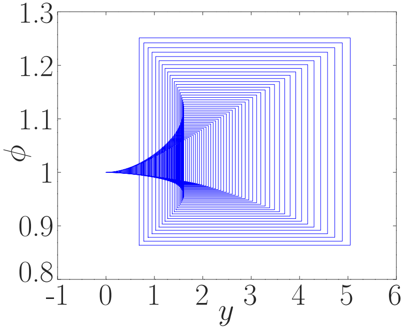
Sherlock, 526 seconds

- 4 state dimensions, 2 control dimensions
- Comparable precision

Airplane



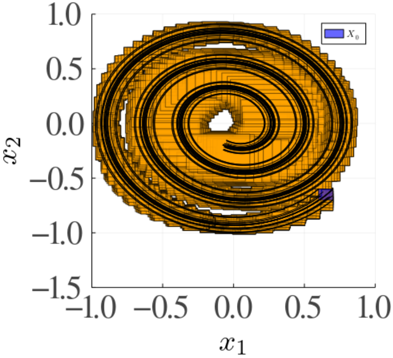
JuliaReach, 29 seconds



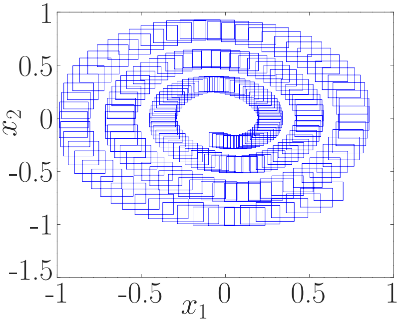
Sherlock, stopped after 169 seconds

- 12 state dimensions, 6 control dimensions
- **Sherlock** diverges

Translational oscillations by a rotational actuator (TORA)



JuliaReach, 2040 seconds



Sherlock, 30 seconds

- 4 state dimensions, 1 control dimension
- **JuliaReach** needs to split the initial states

Two reachability algorithms

Reachability algorithm 1[1]

- Combination of **Taylor models**[2] and **zonotopes**[3]
- Implemented in **JuliaReach**[4]

Reachability algorithm 2[5]

- **Polynomial zonotopes**[6]
- Implemented in **CORA**[7]

[1] Schilling, Forets, and Guadalupe. *AAAI*. 2022.
[2] Makino and Berz. *Int. J. Pure Appl. Math* (2003).
[3] Singh, Gehr, Mirman, Püschel, and Vechev. *NeurIPS*. 2018.
[4] Bogomolov, Forets, Frehse, Potomkin, and Schilling. *HSCC*. 2019.
[5] Kochdumper, Schilling, Althoff, and Bak. *NASA Formal Methods*. 2023.
[6] Kochdumper and Althoff. *IEEE Trans. Autom. Control*. (2021).
[7] Althoff. *ARCH*. 2015.

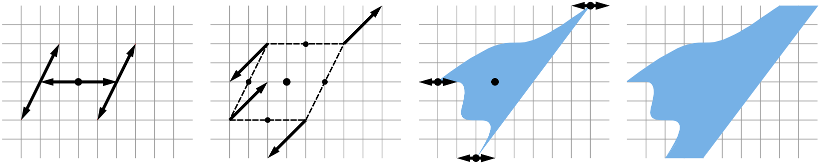
Polynomial zonotope - Example

The polynomial zonotope

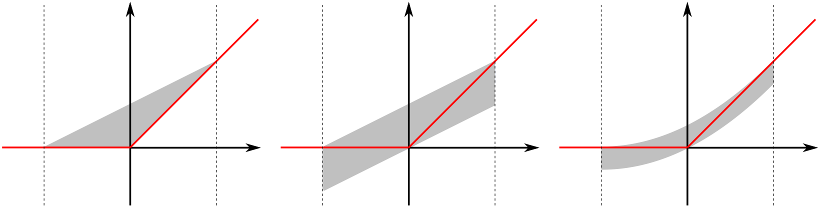
$$\left\langle \begin{bmatrix} 4 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 & 1 & 2 \\ 0 & 2 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \end{bmatrix} \right\rangle$$

defines the set

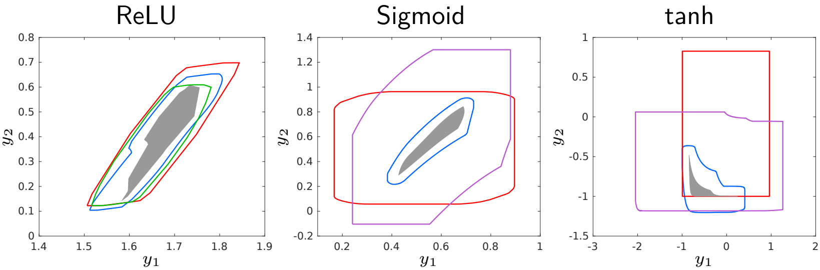
$$\left\{ \begin{bmatrix} 4 \\ 4 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} \alpha_1 + \begin{bmatrix} 1 \\ 2 \end{bmatrix} \alpha_2 + \begin{bmatrix} 2 \\ 2 \end{bmatrix} \alpha_1^3 \alpha_2 + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \beta_1 \;\middle|\; \alpha_1, \alpha_2, \beta_1 \in [-1, 1] \right\}$$



Approximation of ReLU activation

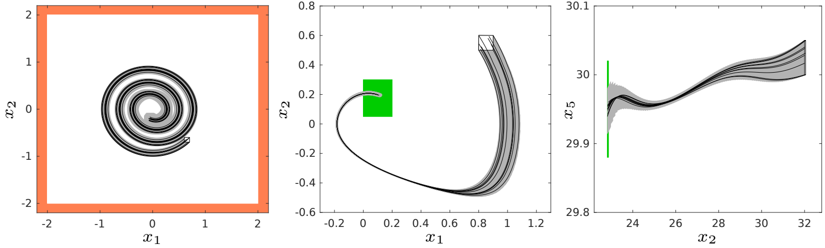


Approximation quality on random neural networks



- **Polynomial zonotope**
- **Zonotope**
- **Star set** (only applicable to ReLU)
- **Taylor model** (only applicable to Sigmoid and tanh)
- **Exact solution**

Approximation quality on control problems



Comparison

	ReLU			sigmoid					hyp. tangent				
	Sherlock	JuliaReach	Poly. zono.	Verisig	Verisig 2.0	ReachNN*	POLAR	Poly. zono.	Verisig	Verisig 2.0	ReachNN*	POLAR	Poly. zono.
B1 $(2, 2, 20)$				-	49	69	23	**2**	-	48	-	25	**8**
B2 $(2, 2, 20)$				12	8	32	10	**1**	-	-	-	**3**	-
B3 $(2, 2, 20)$				98	47	130	37	**3**	98	43	128	38	**3**
B4 $(3, 2, 20)$				24	12	20	4	**1**	23	11	20	4	**1**
B5 $(3, 3, 100)$				196	1063	31	25	**2**	-	168	-	31	**2**
TORA $(4, 3, a)$	30	2040	**13**	136	83	13402		**1**	134	70	2524		**1**
ACC $(6, b, 20)$	4	**1**	2						-	1512	-	312	**2**
Unic. $(3, 1, 500)$	526	93	**3**										
Airp. $(12, 3, 100)$	-	29	**7**										
SPen. $(2, 2, 25)$	1	1	**1**										

Overview

Problem

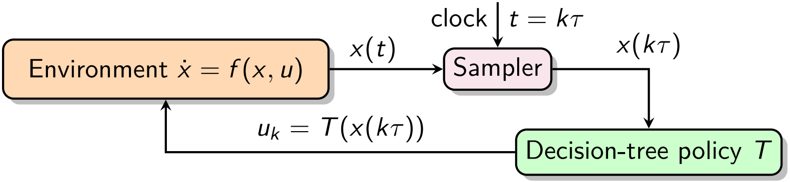
Neural-network controllers

Decision-tree controllers

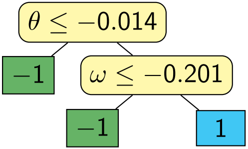
Conclusion

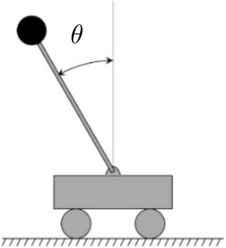
Decision-tree control system[1]

A **decision-tree control system** (DTCS) uses a decision-tree policy $T : \mathbb{R}^n \to U$ where $U \subseteq \mathbb{R}^m$



Example: Cart/pole system





[1] Schilling, Lukina, Demirović, and Larsen. *NeurIPS*. Spotlight. 2023.

Reach-avoid problem for DTCS

- We consider axis-aligned decisions ("$x \leq c$"), used in various tools such as Uppaal Stratego[1] and dtControl[2]
- If $f(x, u) = u$, we call f **state independent**

Theorem

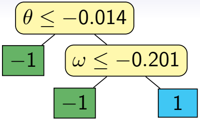
The reach-avoid problem for **state-independent** DTCS is

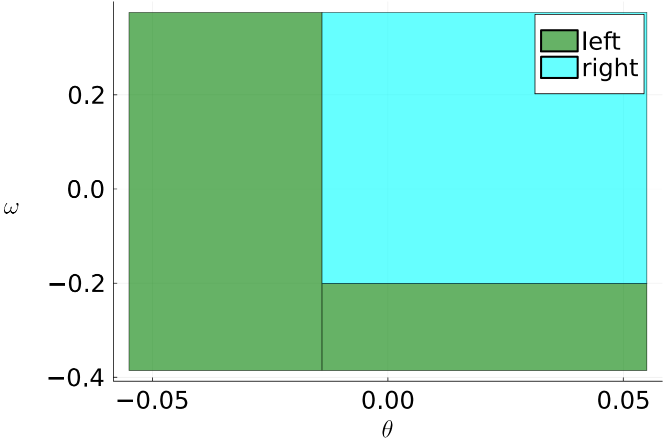
1. undecidable in unbounded time
2. PSPACE-complete in bounded time

[1] David, Jensen, Larsen, Mikucionis, and Taankvist. *TACAS*. 2015.

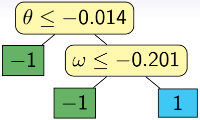
[2] Ashok, Jackermeier, Jagtap, Kretínský, Weininger, and Zamani. *HSCC*. 2020.

Algorithm sketch

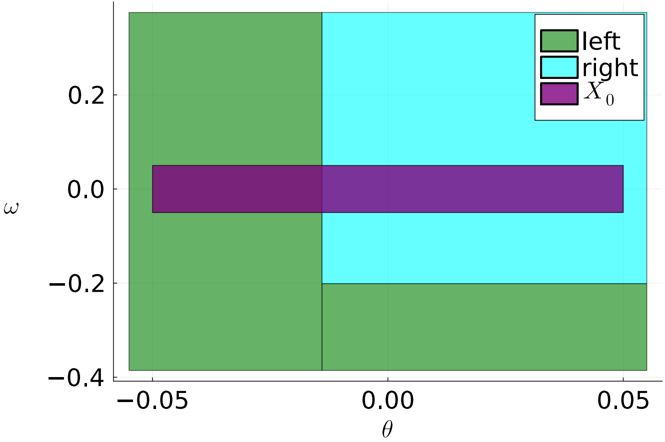




Algorithm sketch



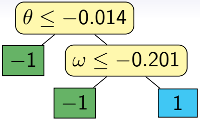
$\mathcal{X}_0$: $p_0, v_0, \theta_0, \omega_0 \in [-0.05, 0.05]$

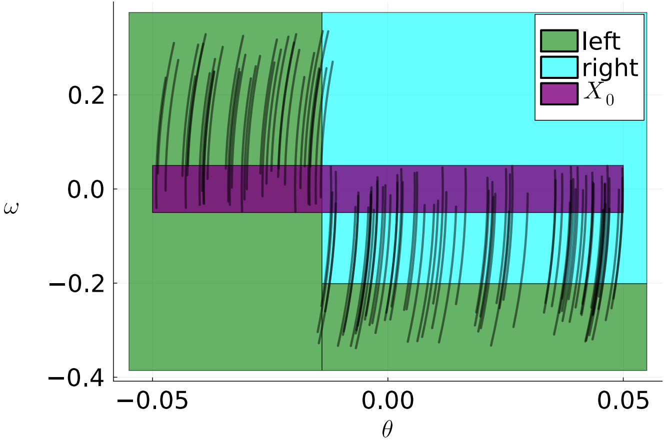


Algorithm sketch

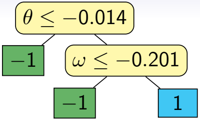
$\mathcal{X}_0$: $p_0, v_0, \theta_0, \omega_0 \in [-0.05, 0.05]$

τ: 0.02



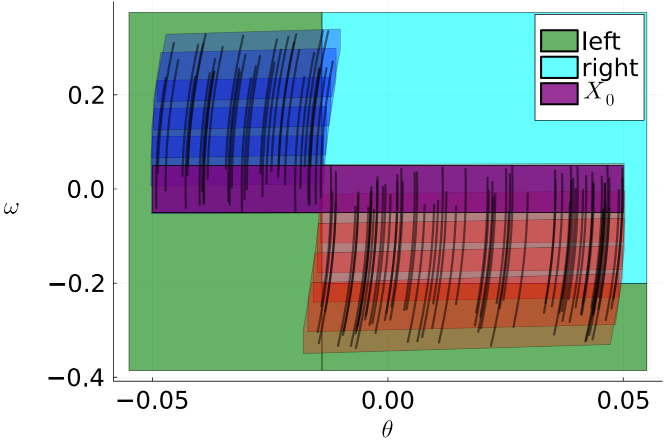


Algorithm sketch

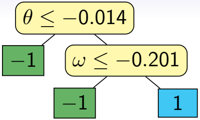


$\mathcal{X}_0$: $p_0, v_0, \theta_0, \omega_0 \in [-0.05, 0.05]$

τ: 0.02

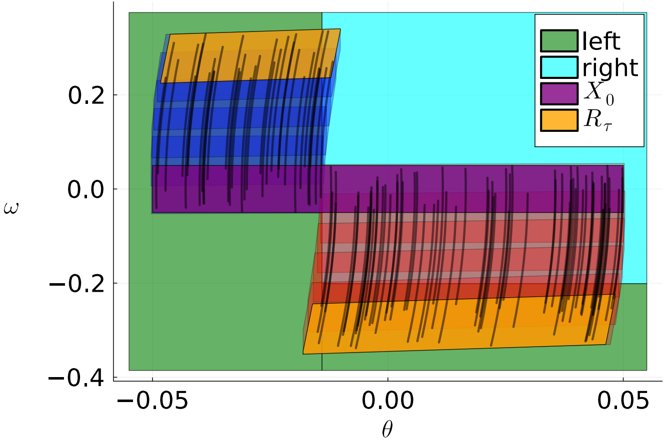


Algorithm sketch



$\mathcal{X}_0$: $p_0, v_0, \theta_0, \omega_0 \in [-0.05, 0.05]$

τ: 0.02

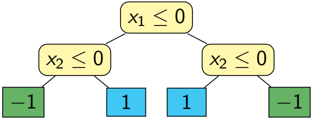


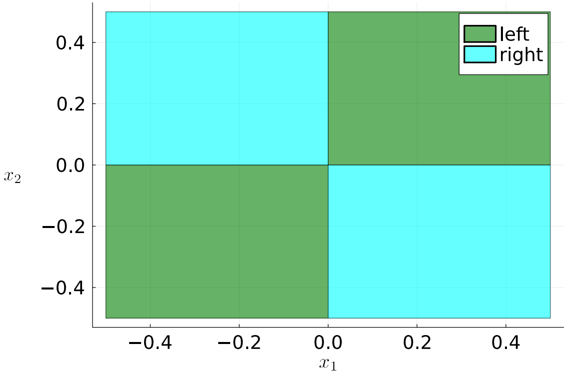
Reachability algorithm

1. Start from set of **initial states** $\mathcal{X}_0$
2. For each control action $u \in U$
   2.1 Compute subset $\mathcal{X}_0^u$
   2.2 Compute set of **reachable states** $\mathcal{R}_{[0,\tau]}^u$ under $f(x, u)$
   2.3 Compute set of **final reachable states** $\mathcal{R}_\tau^u$
3. Obtain set of **reachable states** $\mathcal{R}_{[0,\tau]} = \bigcup_u \mathcal{R}_{[0,\tau]}^u$
4. Obtain set of **final reachable states** $\mathcal{R}_\tau = \bigcup_u \mathcal{R}_\tau^u$

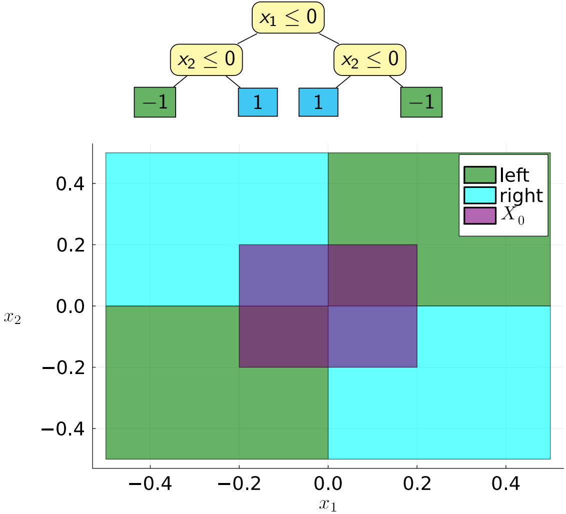
- We use **Taylor models** to represent the sets $\mathcal{R}_{[0,\tau]}^u$ and $\mathcal{R}_\tau^u$
- We can repeat from $\mathcal{R}_\tau$ for time interval $[\tau, 2\tau]$, etc.
- Unions of sets: generally must be treated individually
  - Step 4 creates unions
  - Step 2.1 also creates unions

Example for union creation when selecting subsets $\mathcal{X}_0^u$

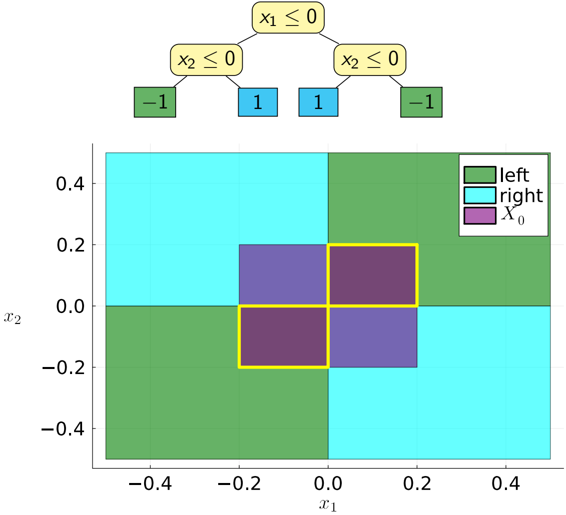




Example for union creation when selecting subsets $\mathcal{X}_0^u$



Example for union creation when selecting subsets $\mathcal{X}_0^u$

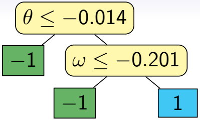


Comparison to generic algorithm

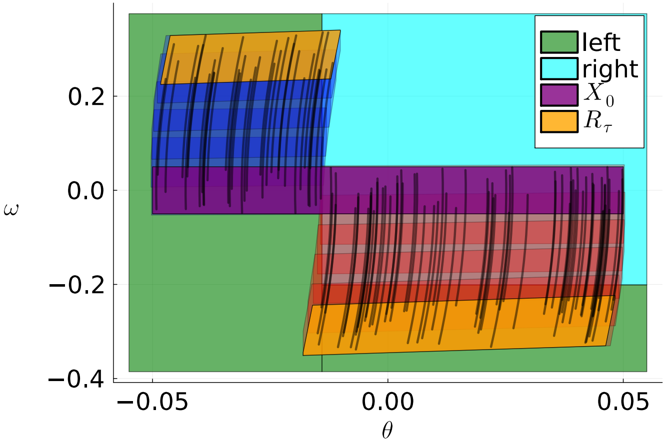
- A DTCS is a special case of a hybrid automaton
- Generic reachability algorithm failed in all case studies
- Our algorithm exploits problem structure in several ways

- **Periodic controller** $\rightsquigarrow$ **only check conditions occasionally**
- T **partitions** a set $\mathcal{X}$ in a hierarchical way
  - Each predicate either splits $\mathcal{X}$ in two or keeps it **intact**
  - If **intact** $\rightsquigarrow$ **no need to explore complement branch**
- No split for **leaves with same action** $\rightsquigarrow$ **no precision loss**
- **Axis-aligned predicates** $\rightsquigarrow$ **cheap interval abstractions**

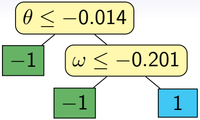
Dealing with set splits



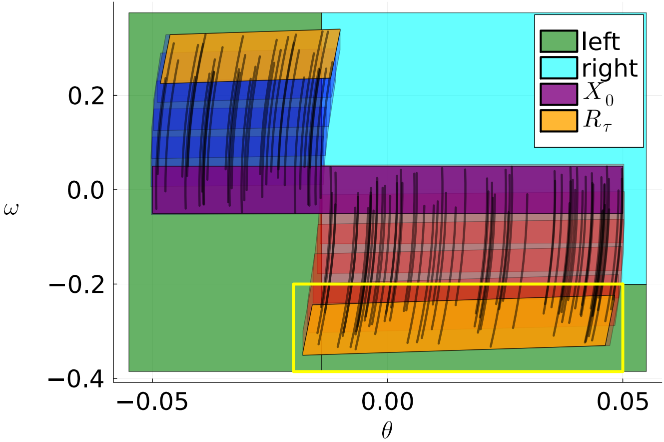
- Only set $\mathcal{R}_\tau$ after control period is relevant



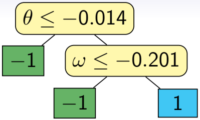
Dealing with set splits



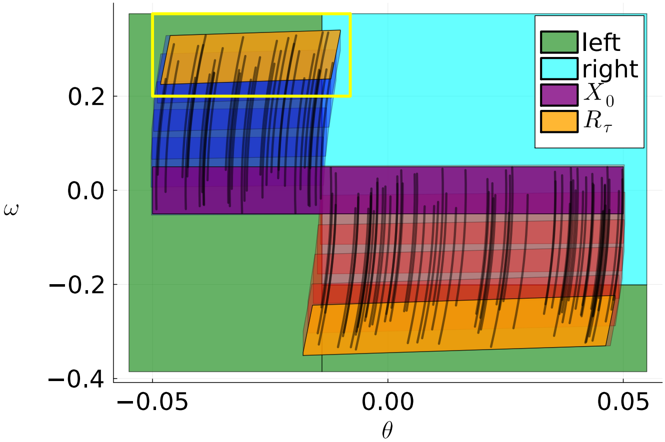
- Bottom set covered by green sets ✓



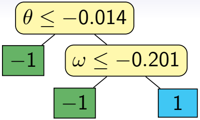
Dealing with set splits

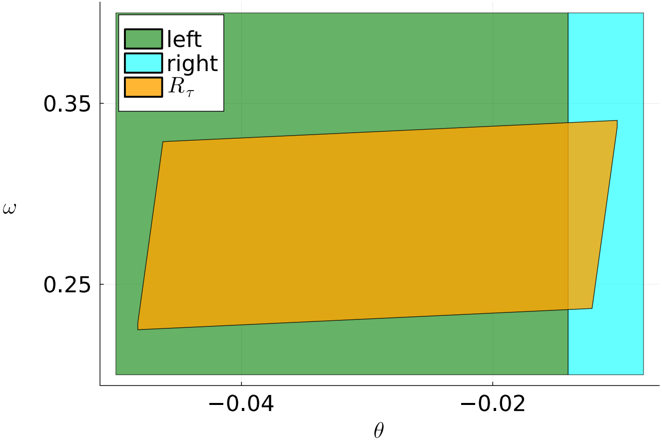


- Top set: zoom in

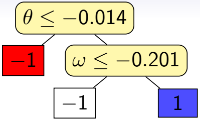


Dealing with set splits

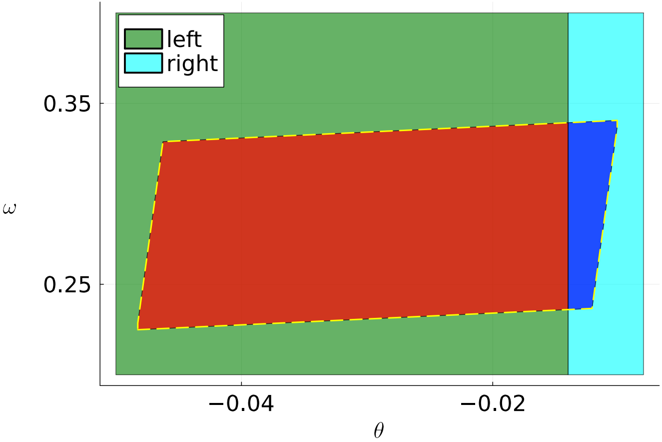




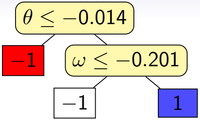
Dealing with set splits



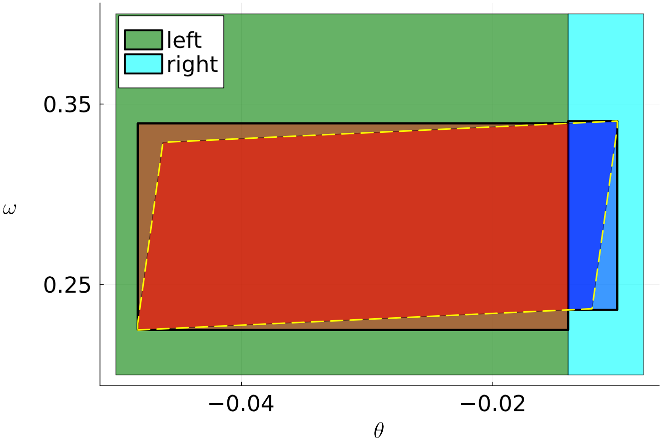
- Complexity of set representation grows



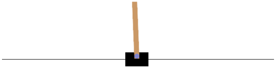
Dealing with set splits



- Complexity of set representation grows
- Interval approximations to tame complexity



Cart policy to stabilize a pole



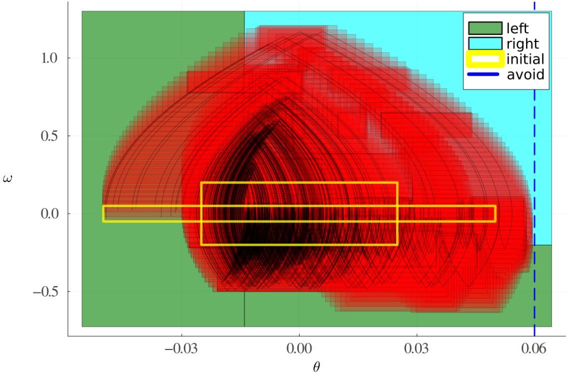
$$\dot{\theta} = \omega$$

$$\dot{\omega} = \phi$$

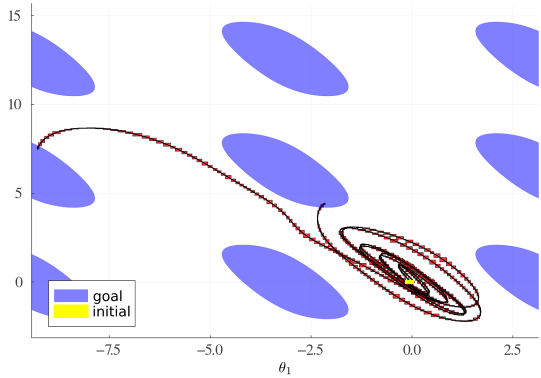
$$\dot{v} = \psi - \frac{1}{22}\phi\cos(\theta)$$

$$\phi = \frac{9.8\sin(\theta) - \cos(\theta)\psi}{2/3 + 5/11\cos(\theta)^2}$$

$$\psi = \frac{10u + 0.05\omega^2\sin(\theta)}{1.1}$$



Acrobot policy to swing up to a goal height



$$\ddot{\theta}_1 = -\frac{d_2\psi + \phi_1}{d_1}, \quad \ddot{\theta}_2 = \psi$$

$$d_1 = I_1 + I_2 + m_1 lc_1^2 + m_2(l_1^2 + lc_2^2 + 2 l_1 lc_2 \cos(\theta_2))$$

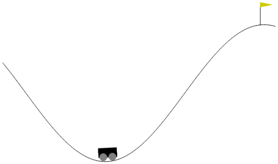
$$d_2 = m_2(lc_2^2 + l_1 lc_2 \cos(\theta_2)) + I_2$$

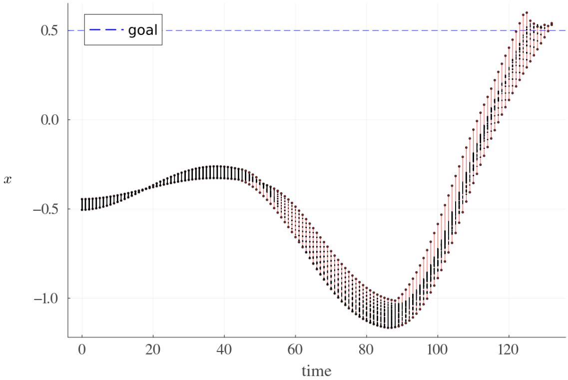
$$\phi_2 = m_2 lc_2 g \cos\left(\theta_1 + \theta_2 - \frac{\pi}{2}\right)$$

$$\phi_1 = -m_2 l_1 lc_2 \dot{\theta}_2^2 \sin(\theta_2) - 2 m_2 l_1 lc_2 \dot{\theta}_2 \dot{\theta}_1 \sin(\theta_2) + (m_1 lc_1 + m_2 l_1) g \cos\left(\theta_1 - \frac{\pi}{2}\right) + \phi_2$$

$$\psi = \left(u + \frac{d_2}{d_1}\phi_1 - m_2 l_1 lc_2 \dot{\theta}_1^2 \sin(\theta_2) - \phi_2\right)\left(m_2 lc_2^2 + I_2 - \frac{d_2^2}{d_1}\right)^{-1}$$

Car policy to reach the top of a mountain





$$v_{k+1} = v_k + (u-1)F - \cos(3x_k)g$$
$$x_{k+1} = x_k + v_{k+1}$$

Quadrotor policy to follow a reference trajectory to a goal

T: 177 nodes

$|U| = 8$

$\dot{p}_x = v_x$

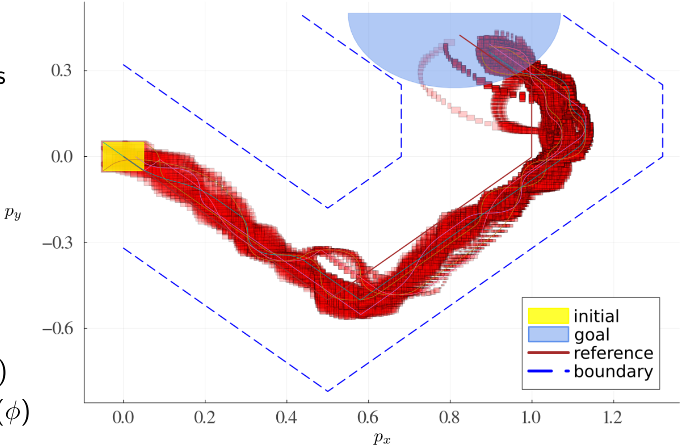
$\dot{p}_y = v_y$

$\dot{p}_z = v_z$

$\dot{v}_x = g\tan(\theta)$

$\dot{v}_y = -g\tan(\phi)$

$\dot{v}_z = \alpha - g$



Size of policies and verification times

System	Policy T			Verification
	#nodes	depth	#actions	time
Cart/pole	5	2	2	15 sec
Acrobot	7	2	2	101 sec
	9	3	2	113 sec
Mountain/car	9	3	3	7 sec
Quadrotor	177	10	8	84 sec

Overview

Problem

Neural-network controllers

Decision-tree controllers

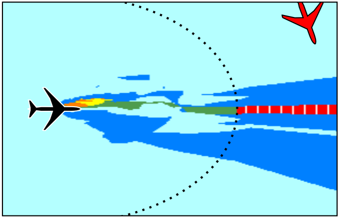
Conclusion

Conclusion

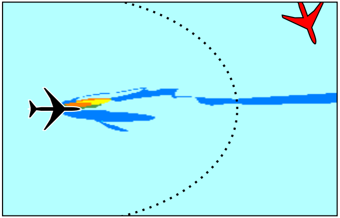
- Reachability approaches for **neural-network control systems**[1][2] and **decision-tree control systems**[3]
- Orthogonal challenges
  - **Neural networks**: infinitely many continuous control actions
  - **Decision trees**: finitely many discontinuous control actions
- Common challenge: repeated set conversion incurs precision loss
  - Mitigated by **Taylor models** and **structured zonotopes**
  - Avoided by **polynomial zonotopes**
  - **Intervals** suitable for axis-aligned decisions
- Future work:
  - Constrained polynomial zonotopes for decision trees
  - Learning safe control policies

[1] Schilling, Forets, and Guadalupe. *AAAI*. 2022.
[2] Kochdumper, Schilling, Althoff, and Bak. *NASA Formal Methods*. 2023.
[3] Schilling, Lukina, Demirović, and Larsen. *NeurIPS*. Spotlight. 2023.

Learning safe control policies[1][2]



Before repair



After repair

[1] Bauer-Marquart, Boetius, Leue, and Schilling. *SPIN*. 2022.
[2] Boetius, Leue, and Sutter. *ICML*. 2023.