Safety verification of decision-tree policies in continuous time

AISoLA 2023

accepted as NeurIPS 2023 spotlight

Christian Schilling

Anna Lukina

Emir Demirović

Kim Guldstrand Larsen

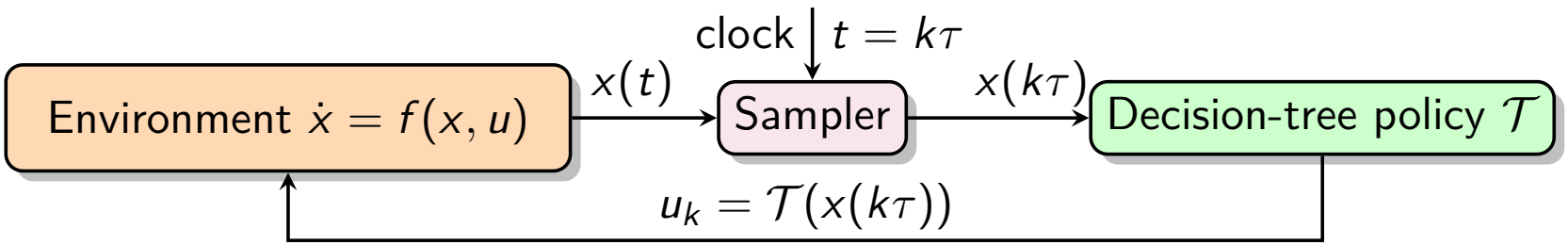
AALBORG UNIVERSITY

TU Delft

Decision-tree control systems

A **decision-tree control system** (DTCS) is a triple $(f, \mathcal{T}, \tau)$ with

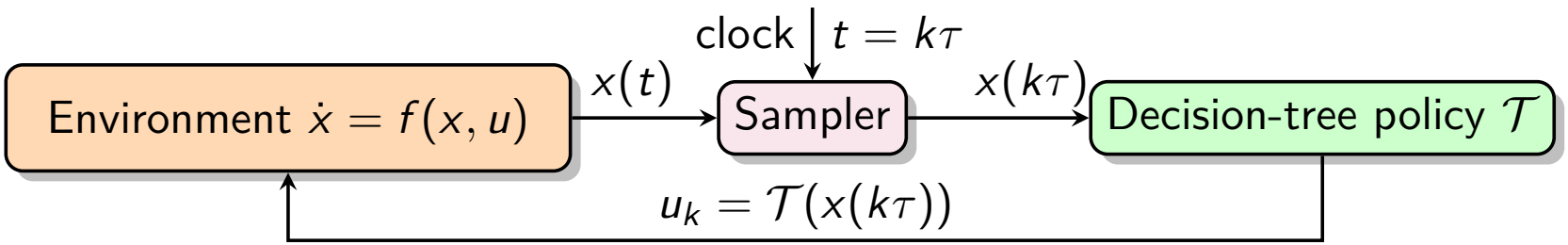
- f: continuous-time environment $\dot{x} = f(x, u) : \mathbb{R}^{n+m} \to \mathbb{R}^n$
- $\mathcal{T}$: decision-tree policy $\mathcal{T} : \mathbb{R}^n \to U$ where $U \subseteq \mathbb{R}^m$
- τ: control period $\tau \in \mathbb{R}^+$



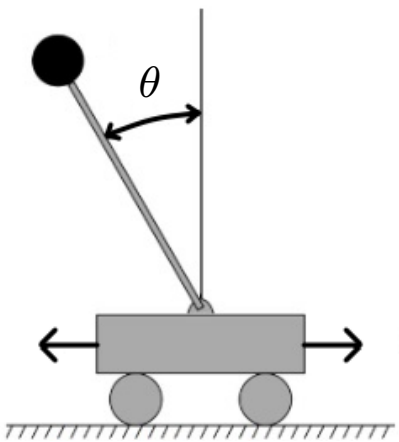
Decision-tree control systems

A **decision-tree control system** (DTCS) is a triple $(f, \mathcal{T}, \tau)$ with

- f: continuous-time environment $\dot{x} = f(x, u) : \mathbb{R}^{n+m} \to \mathbb{R}^n$
- $\mathcal{T}$: decision-tree policy $\mathcal{T} : \mathbb{R}^n \to U$ where $U \subseteq \mathbb{R}^m$
- τ: control period $\tau \in \mathbb{R}^+$



Example: Cart/pole system

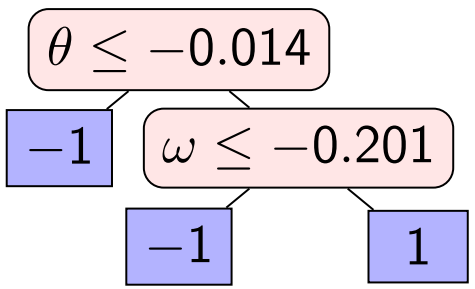


$$\dot{p} = v \qquad \phi = \frac{9.8\sin(\theta) - \cos(\theta)\psi}{2/3 + 5/11\cos(\theta)^2}$$

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

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

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

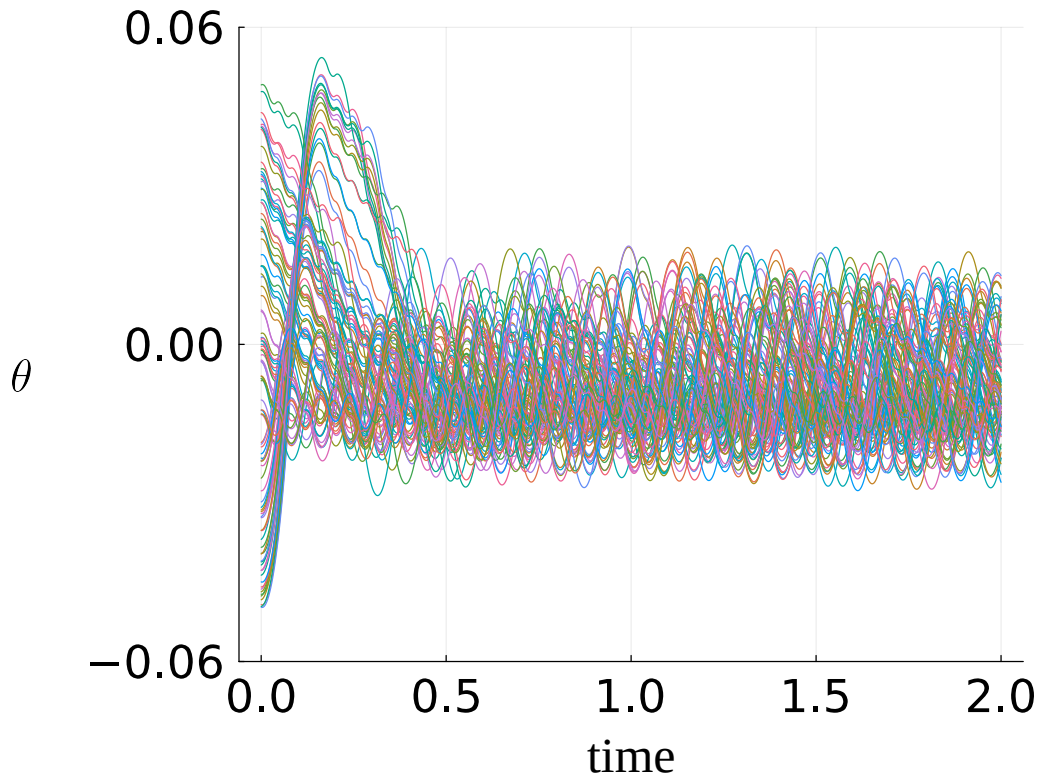


$\tau = 0.02$

Reachable states

Given a DTCS $(f, \mathcal{T}, \tau)$ with axis-aligned decisions ("$x \leq c$"), a set of initial states $\mathcal{X}_0 \subseteq \mathbb{R}^n$, and an iteration bound $k_{\max}$, compute the **set of reachable states**

Example: $p_0, v_0, \theta_0, \omega_0 \in [-0.05, 0.05]$ and $k_{\max} = 100$



Reach-avoid problem

- By computing the **reachable states** we can analyze **reach-avoid specifications**: "avoid A and reach R at time t"

$$\neg A \, \mathcal{U}^{\leq t} \, \neg A \wedge R$$

- **Undecidable** for nonlinear environments $f(x, u)$
- If $f(x, u) = u$, we call f **state independent**

Theorem

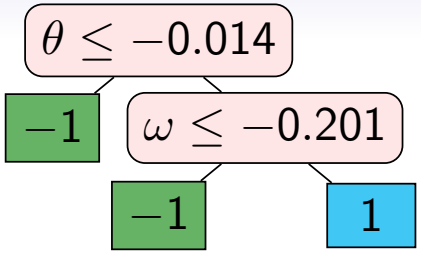
The **reach-avoid problem** for **state-independent** DTCS is

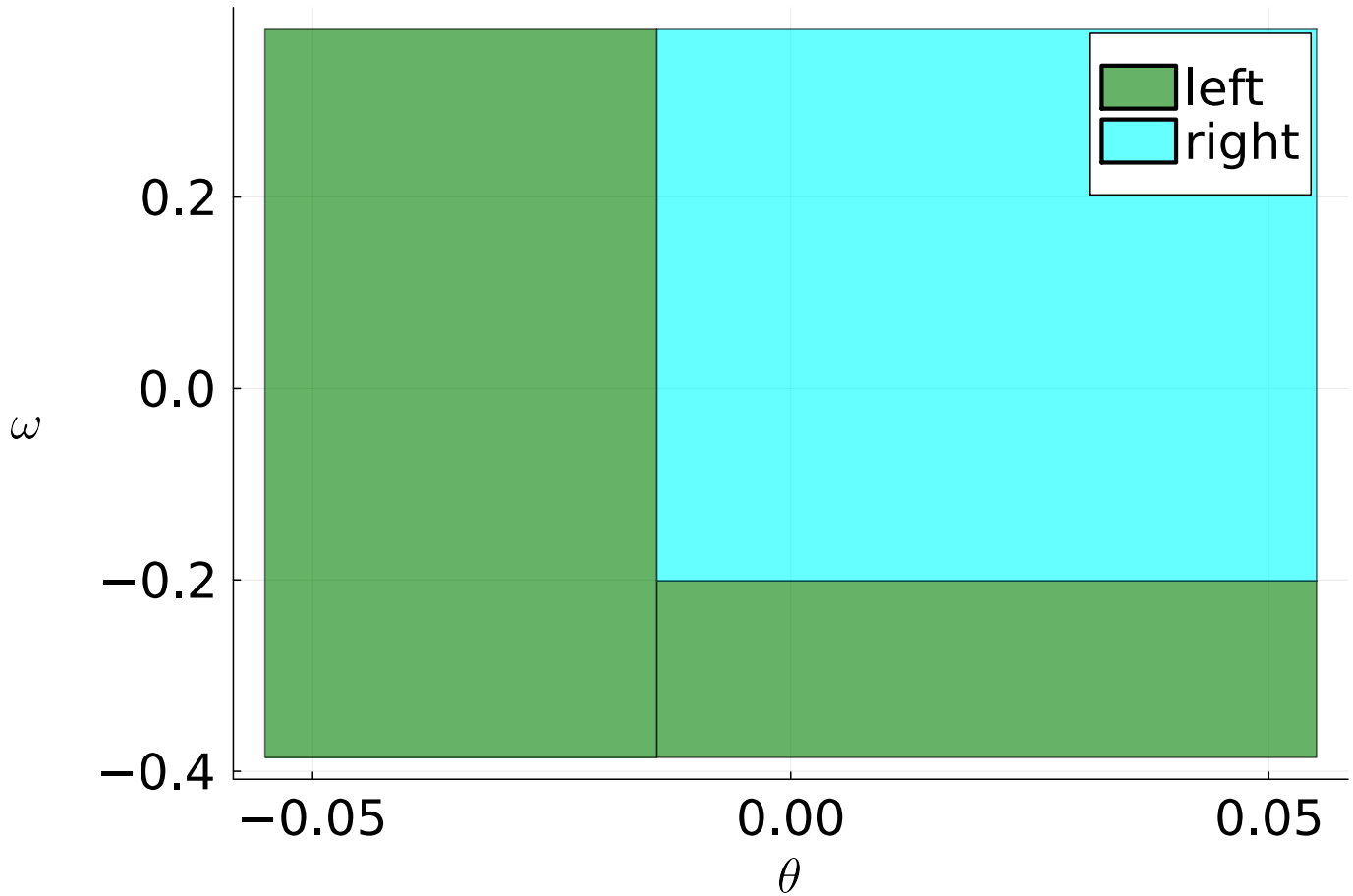
1. **undecidable** for unbounded time
2. **PSPACE-complete** for bounded time

This even holds for $A = \emptyset$ (i.e., the **reachability problem**)

Overview

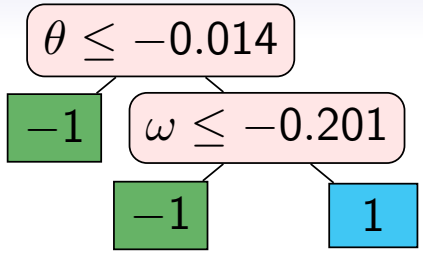
Algorithm sketch

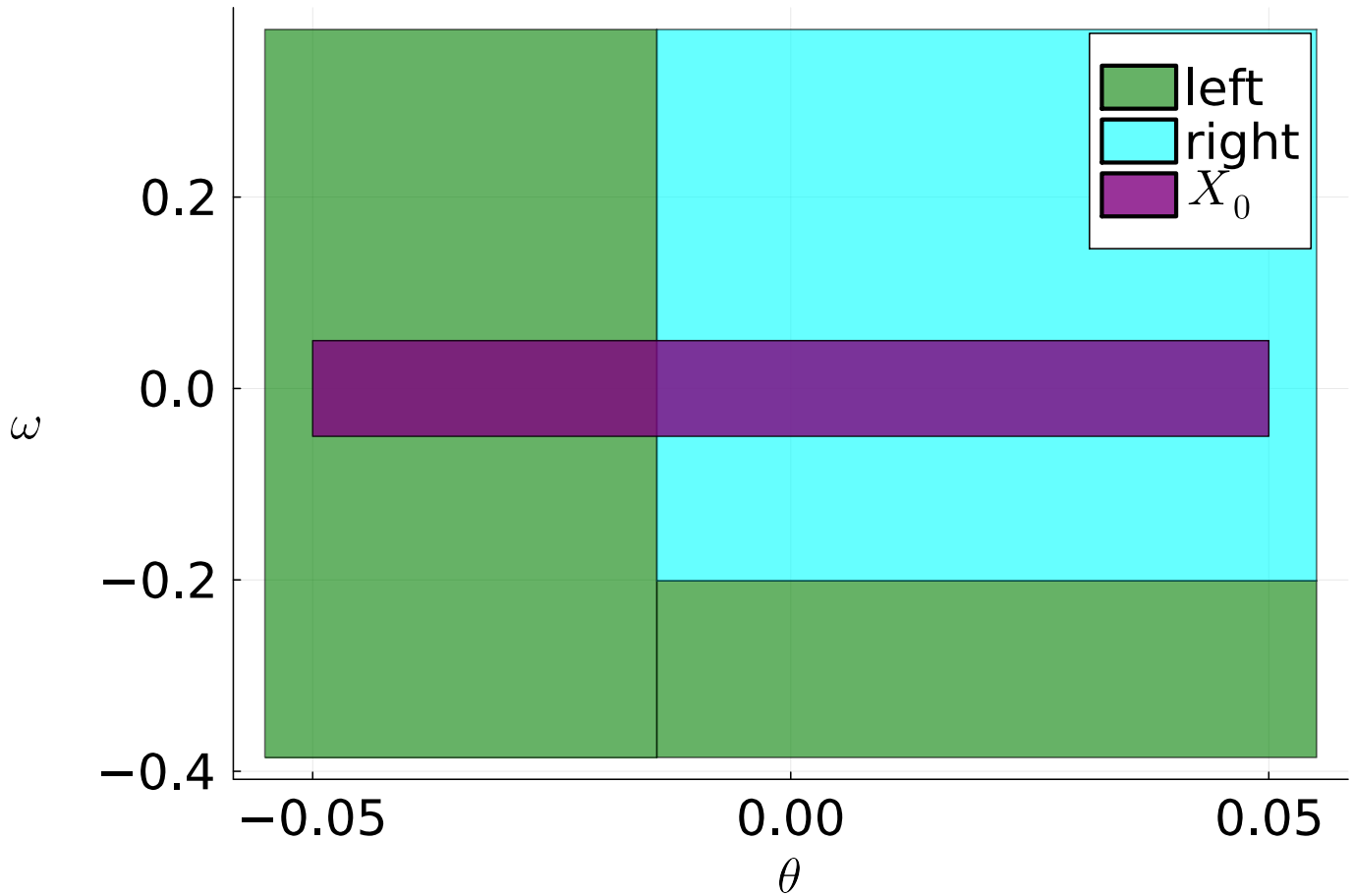




Algorithm sketch

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

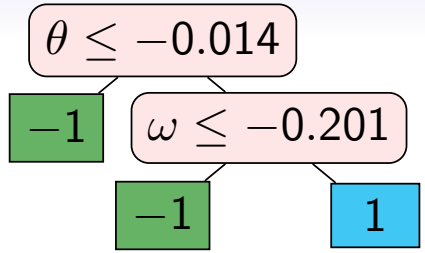


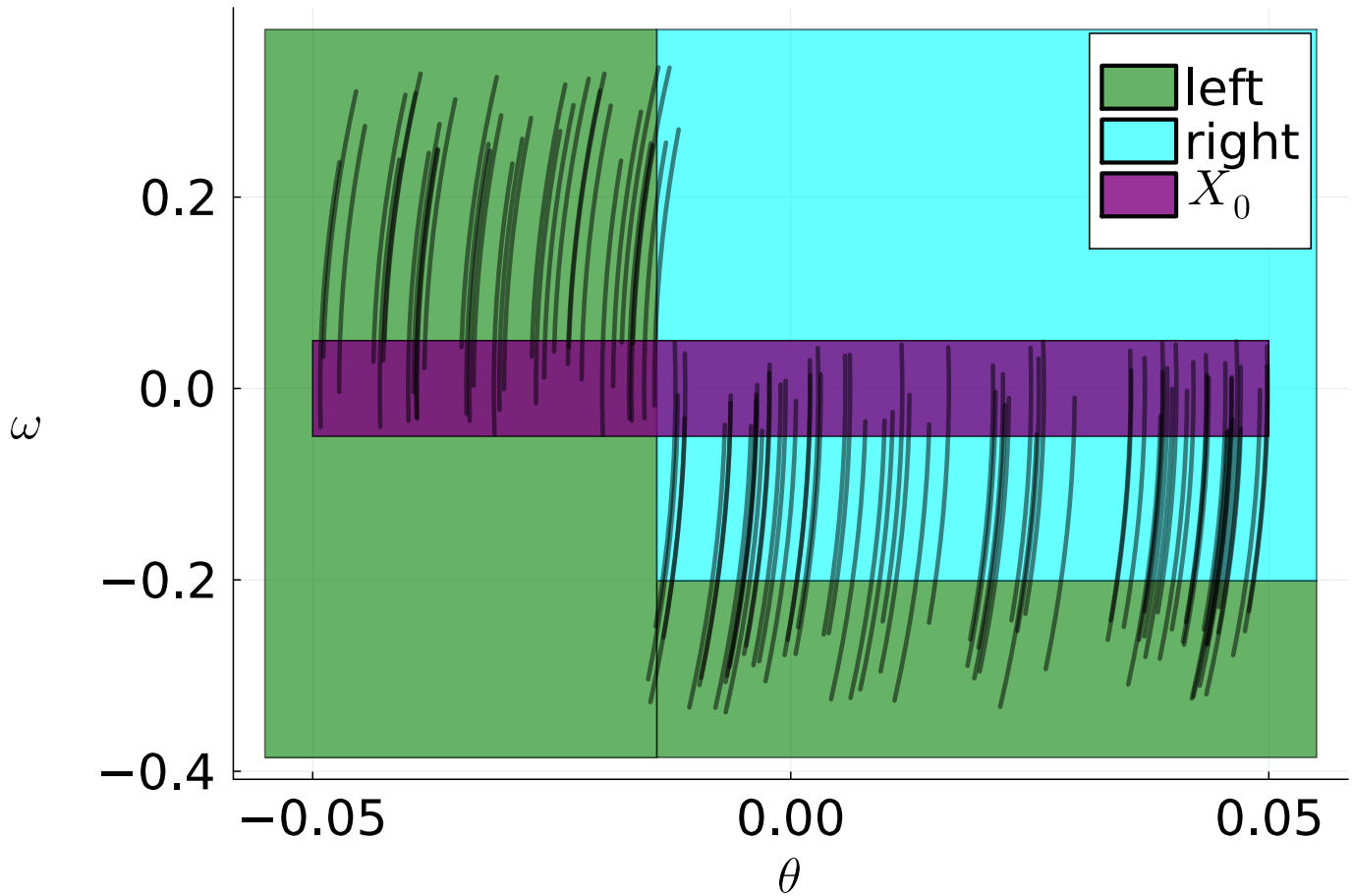


Algorithm sketch

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

τ: 0.02

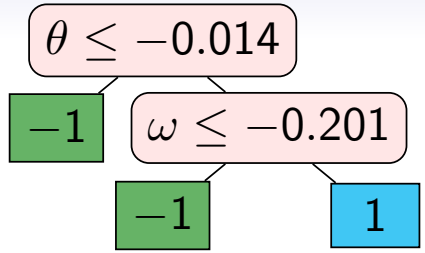


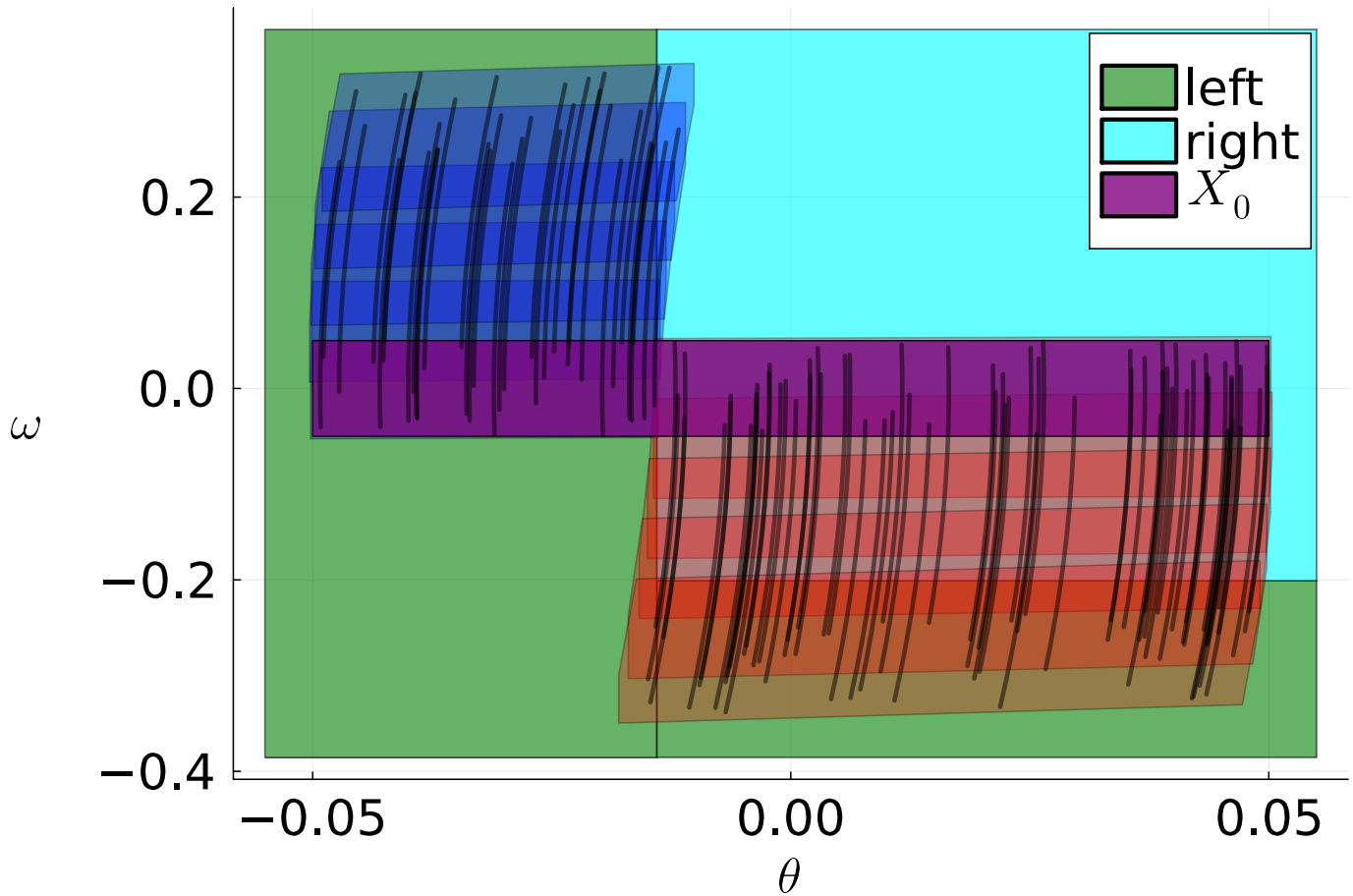


Algorithm sketch

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

τ: 0.02

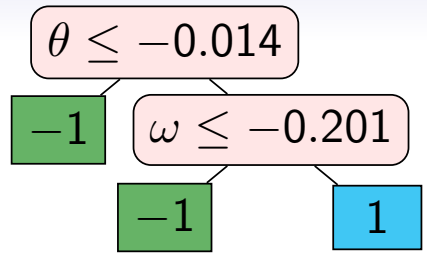


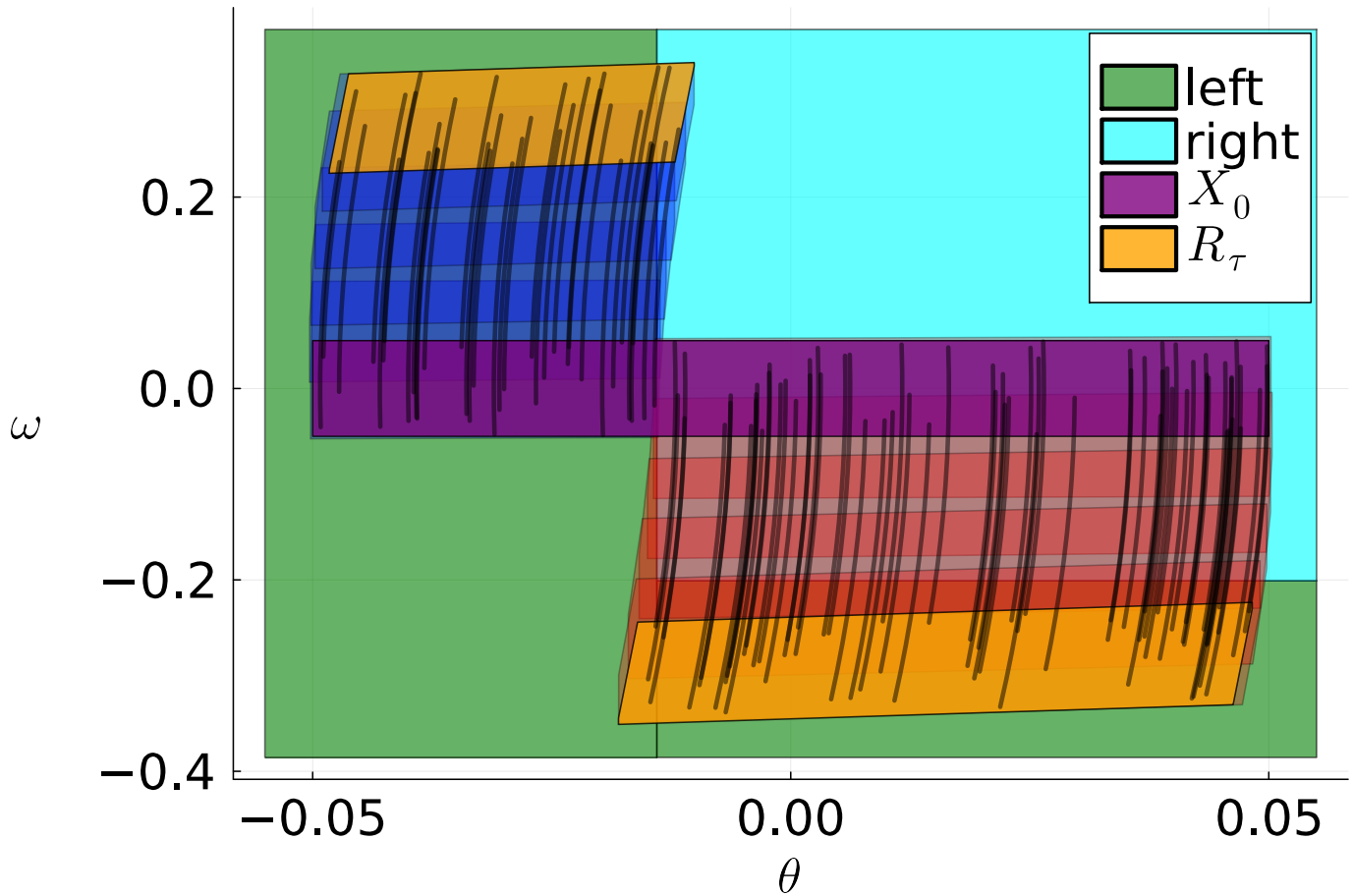


Algorithm sketch

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

τ: 0.02





Reachability algorithm for first time interval $[0, \tau]$

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 $\mathcal{R}_\tau^u$

General 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** $\mathcal{R}^u_{[0,\tau]}$ under $f(x, u)$
   - 2.3 Compute set of **final reachable states** $\mathcal{R}^u_\tau$
3. Obtain set of **reachable states** $\mathcal{R}_{[0,\tau]} = \bigcup_u \mathcal{R}^u_{[0,\tau]}$
4. Obtain set of **final reachable states** $\mathcal{R}_\tau = \bigcup_u \mathcal{R}^u_\tau$

- We use **Taylor models** to represent the sets $\mathcal{R}^u_{[0,\tau]}$ and $\mathcal{R}^u_\tau$
- **We can repeat from $\mathcal{R}_\tau$ for time interval $[\tau, 2\tau]$, etc.**

General 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** $\mathcal{R}^u_{[0,\tau]}$ under $f(x, u)$
   - 2.3 Compute set of **final reachable states** $\mathcal{R}^u_\tau$
3. Obtain set of **reachable states** $\mathcal{R}_{[0,\tau]} = \bigcup_u \mathcal{R}^u_{[0,\tau]}$
4. Obtain set of **final reachable states** $\mathcal{R}_\tau = \bigcup_u \mathcal{R}^u_\tau$

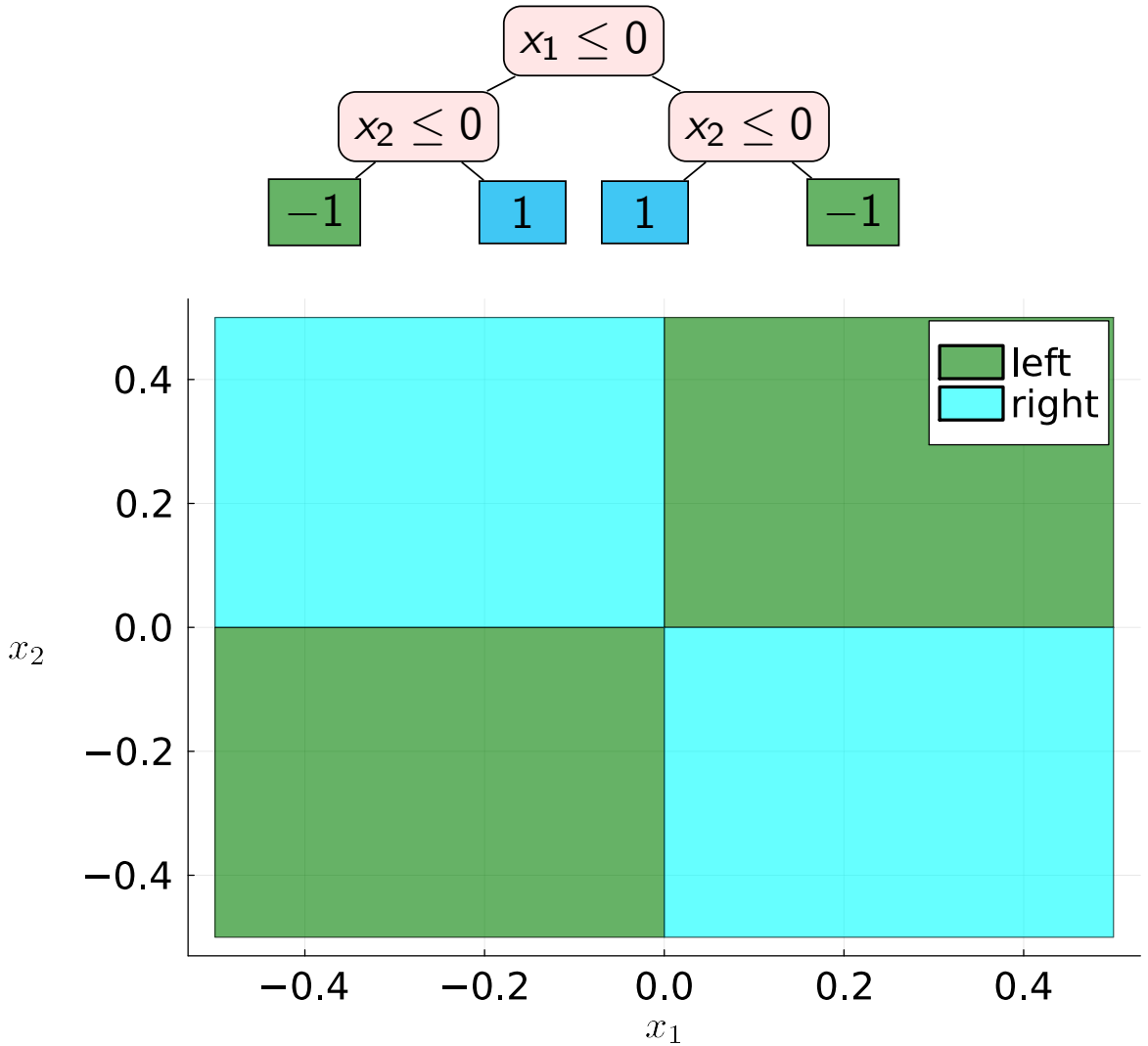
- We use **Taylor models** to represent the sets $\mathcal{R}^u_{[0,\tau]}$ and $\mathcal{R}^u_\tau$
- **We can repeat from $\mathcal{R}_\tau$ for time interval $[\tau, 2\tau]$, etc.**
- **Unions of sets: generally must be treated individually**
  - Step 4 creates unions

General reachability algorithm

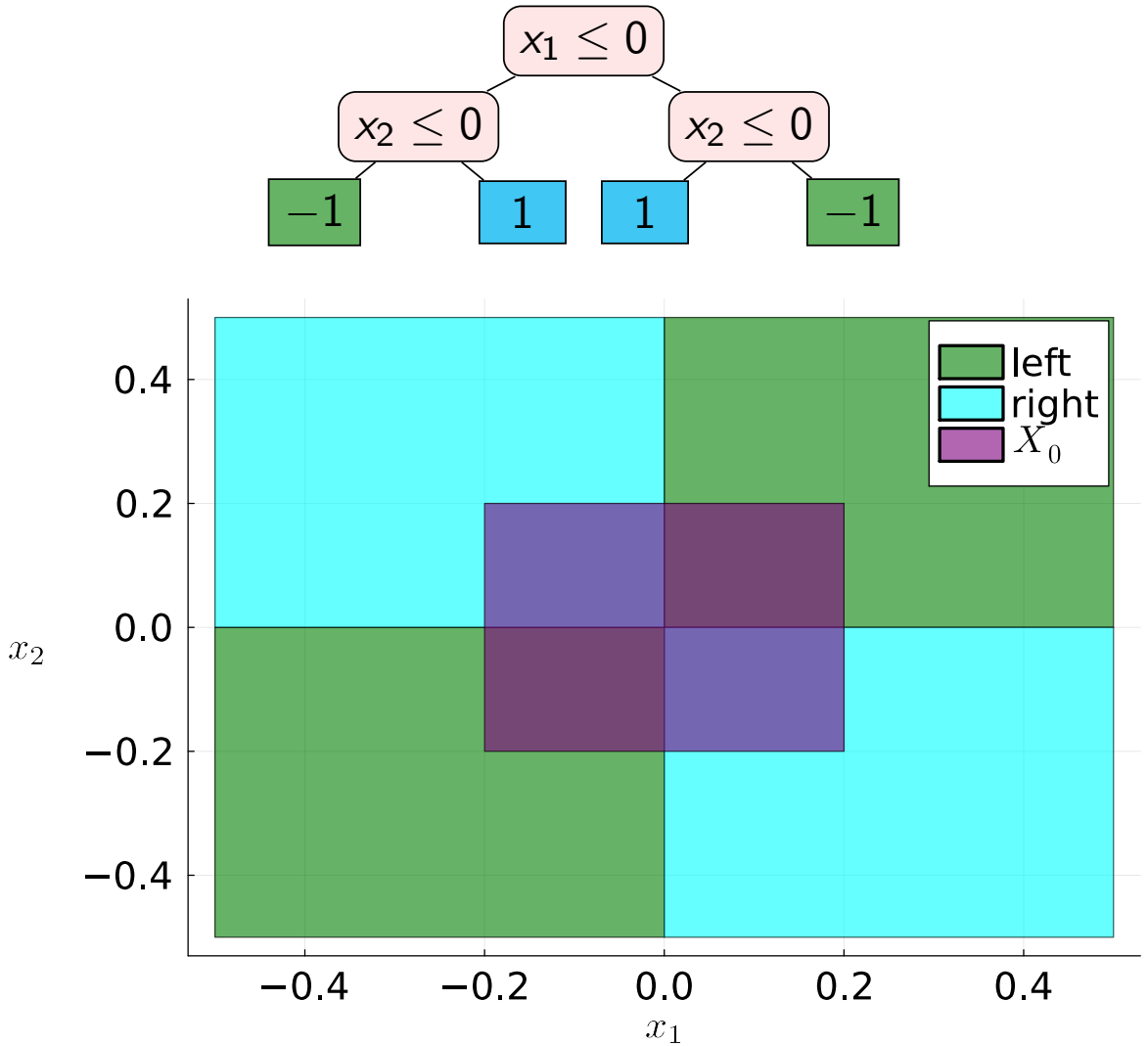
1. Start from set of **initial states** $\mathcal{X}_0$
2. For each control action $u \in U$
   - 2.1 Compute subset $\mathcal{X}_0^u$
   - 2.2 Compute set of **reachable states** $\mathcal{R}^u_{[0,\tau]}$ under $f(x, u)$
   - 2.3 Compute set of **final reachable states** $\mathcal{R}^u_\tau$
3. Obtain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

- We use **Taylor models** to represent the sets $\mathcal{R}^u_{[0,\tau]}$ and $\mathcal{R}^u_\tau$
- **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

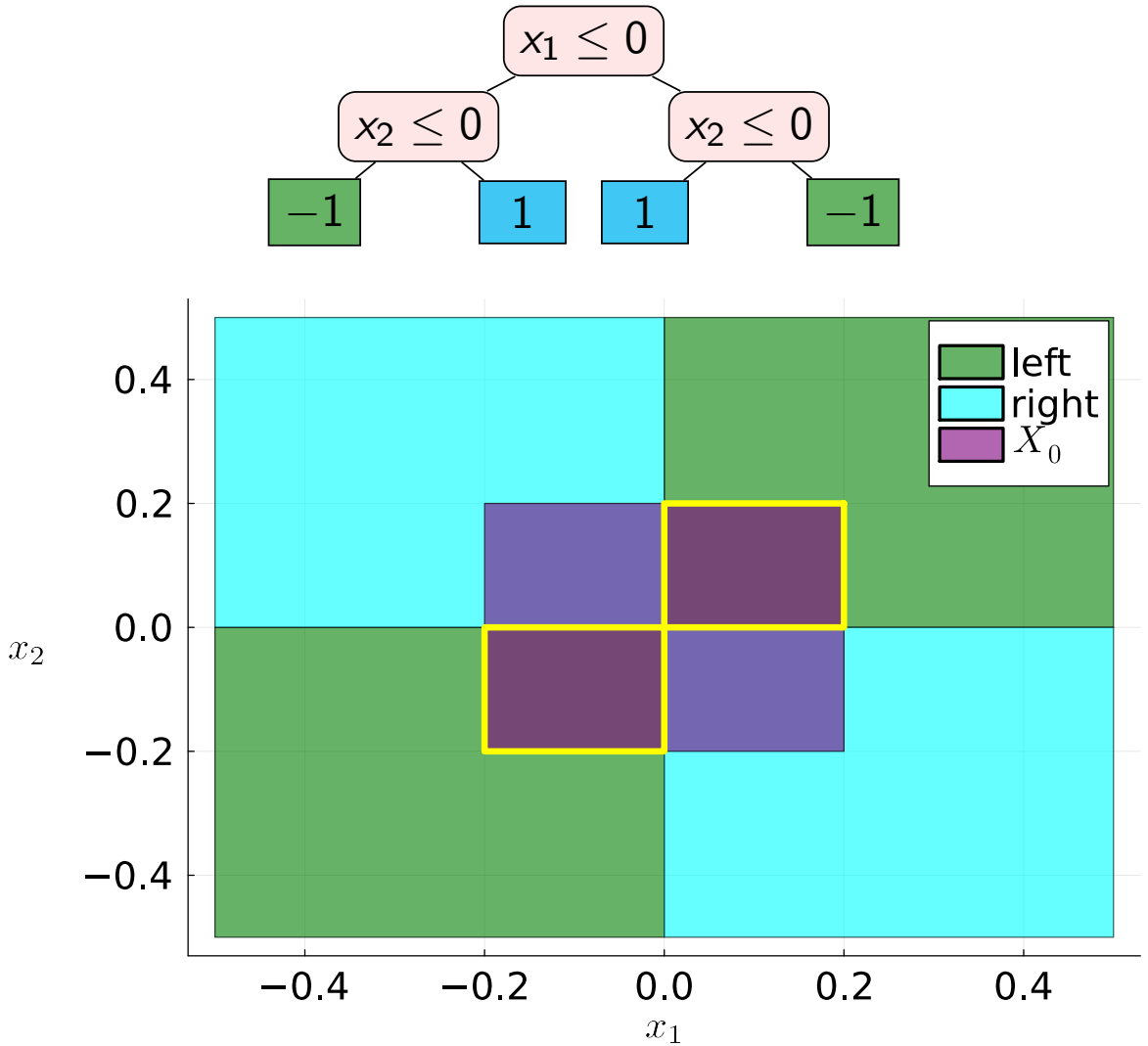
Example for union creation when selecting subsets $\mathcal{X}_0^u$



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



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



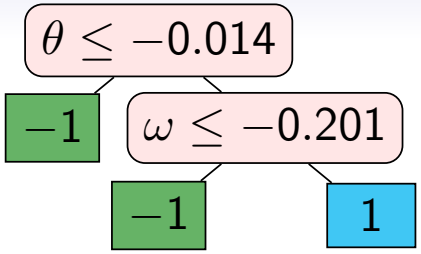
Comparison to generic algorithm

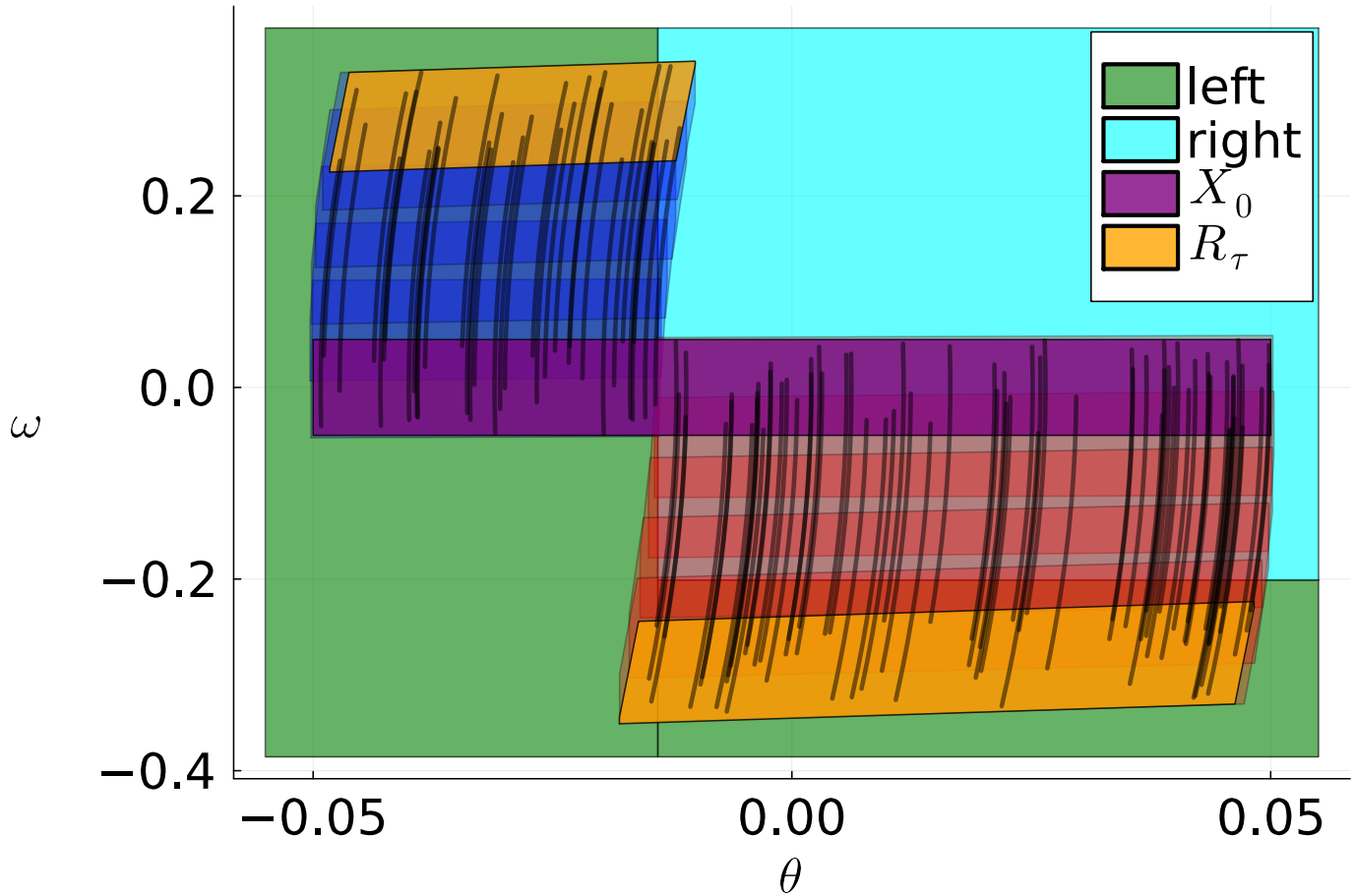
- DTCS is a special case of a **hybrid automaton**
- Generic reachability algorithm failed in all case studies
- Our algorithm exploits problem structure in several ways

- **Periodic controller** $\rightsquigarrow$ **only check conditions occasionally**
- $\mathcal{T}$ **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**
- Often reach **leaves with same action** $\rightsquigarrow$ **no precision loss**
- **Axis-aligned predicates** $\rightsquigarrow$ **cheap interval abstractions**

Dealing with set splits

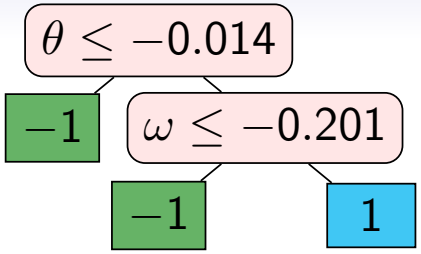
- Only set $\mathcal{R}_\tau$ after control period is relevant

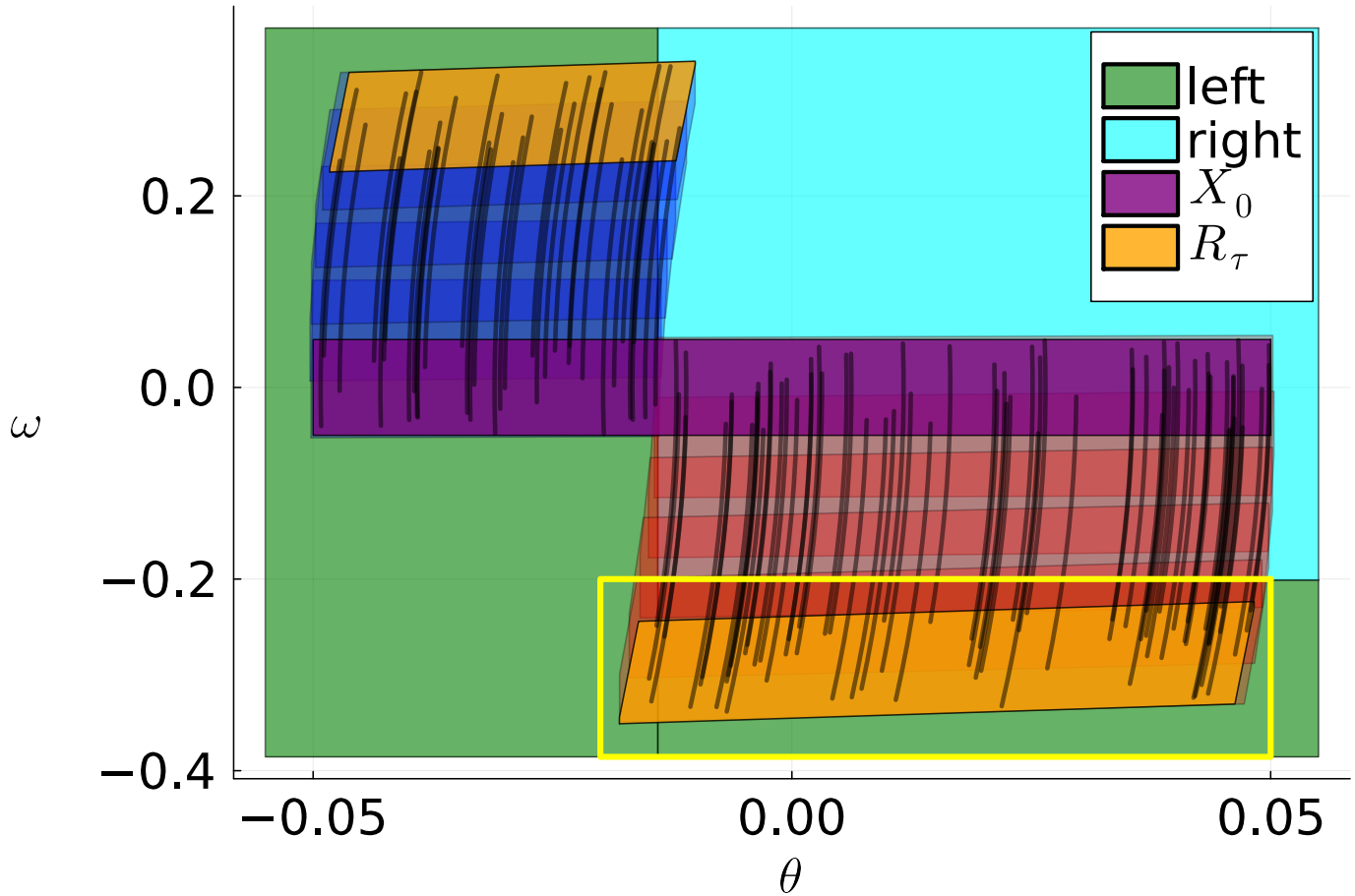




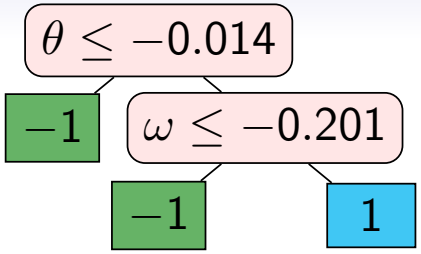
Dealing with set splits

- Bottom set covered by green sets ✓

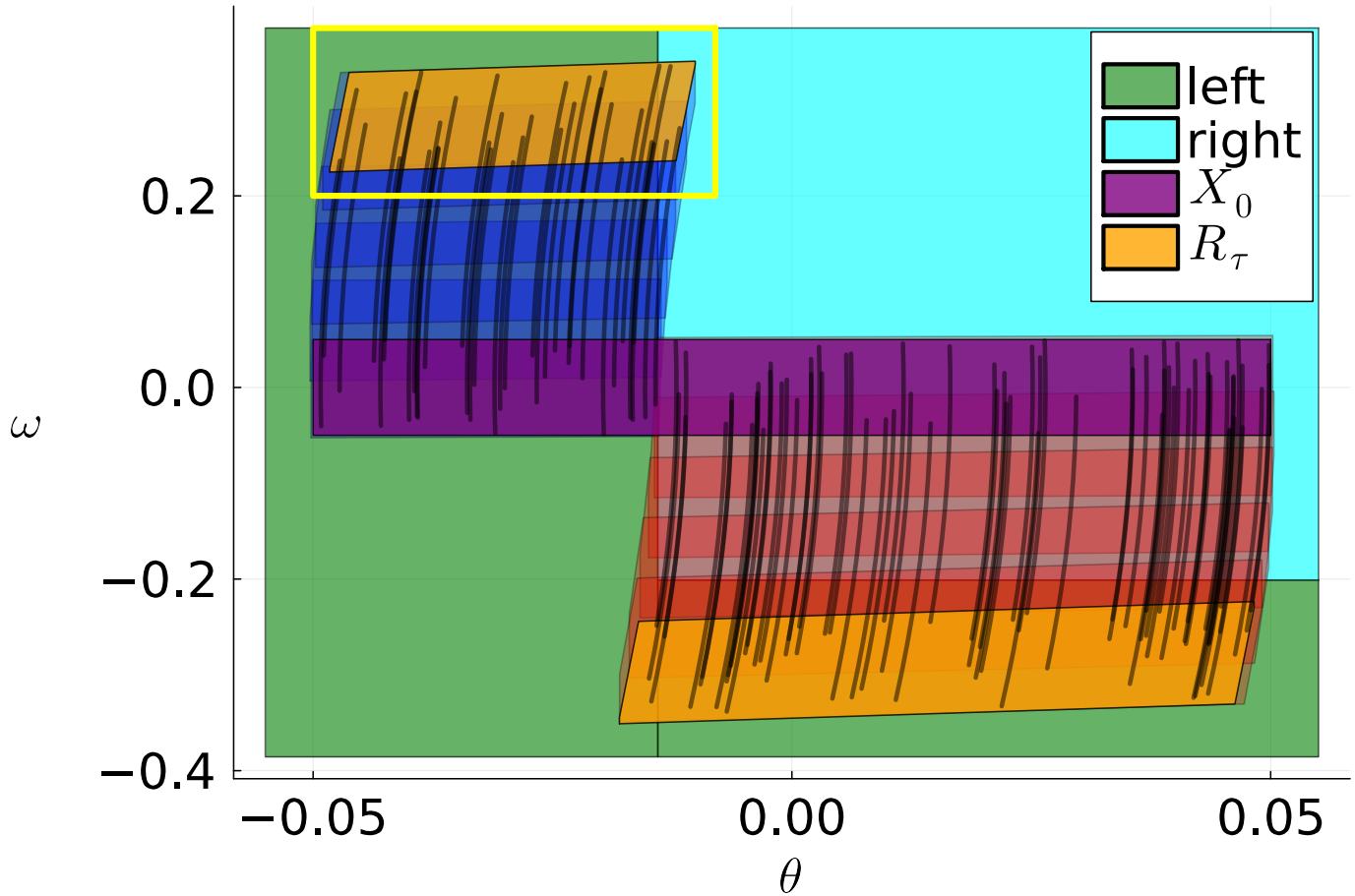




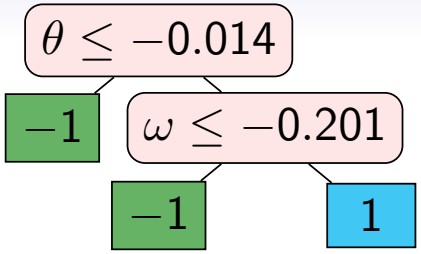
Dealing with set splits

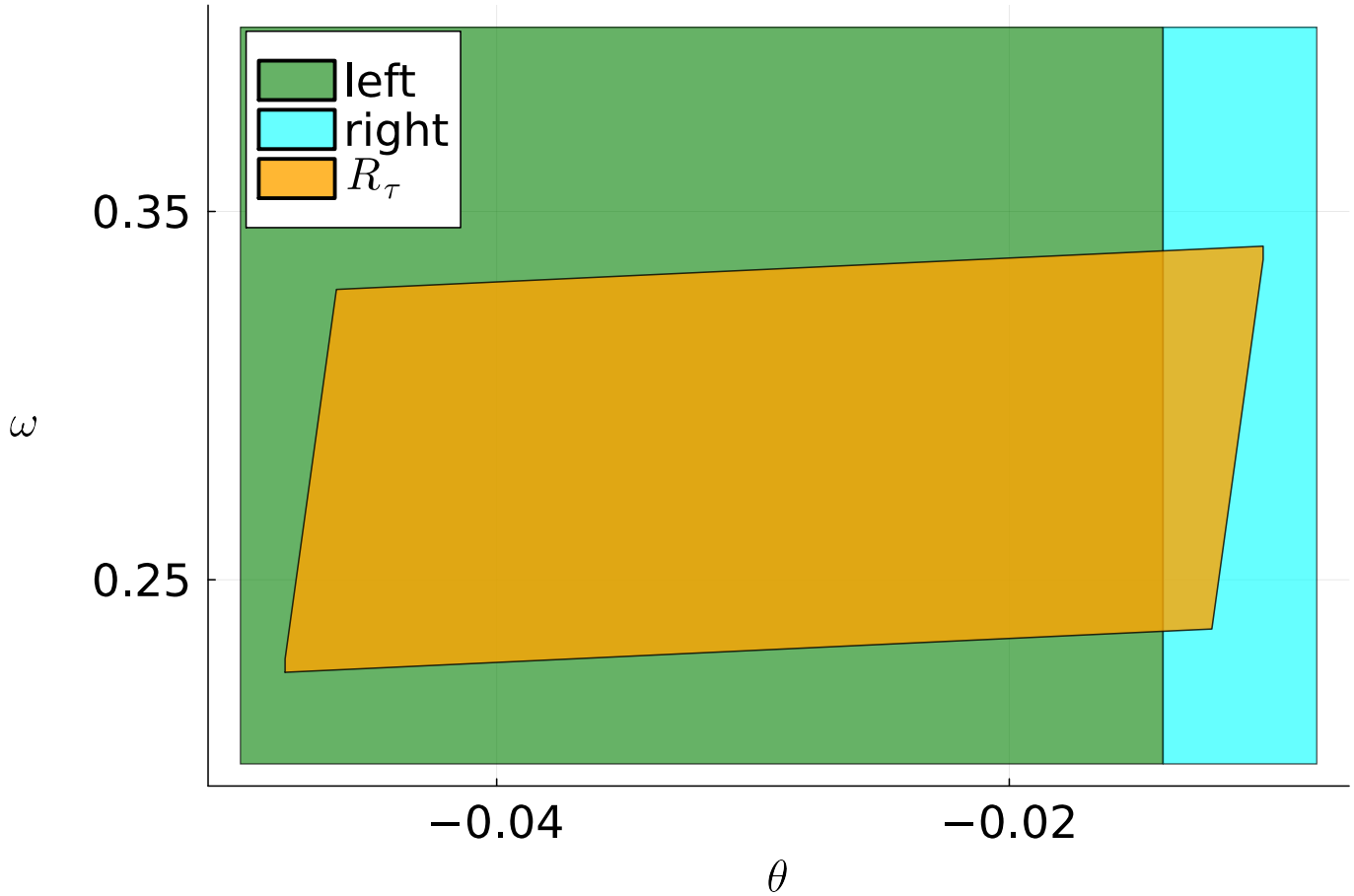


- Split case - let's zoom in



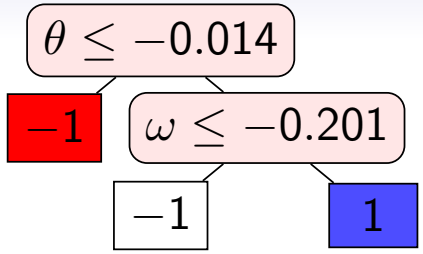
Dealing with set splits

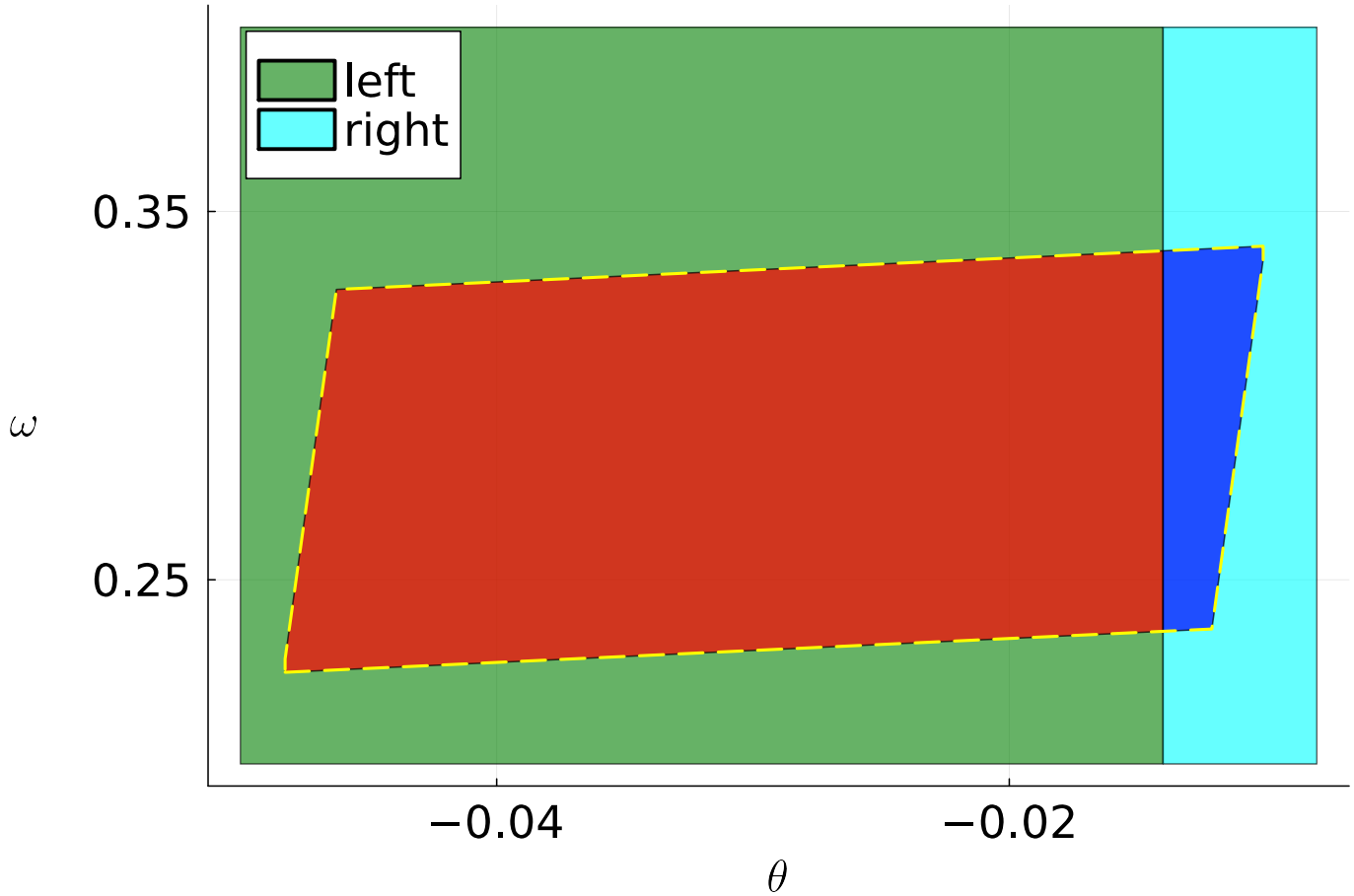




Dealing with set splits

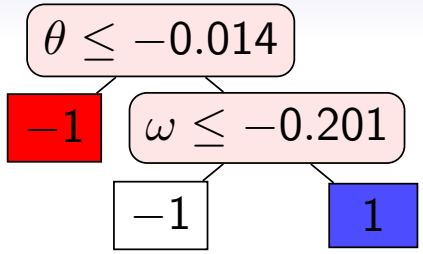
- **Complexity of set representation grows**

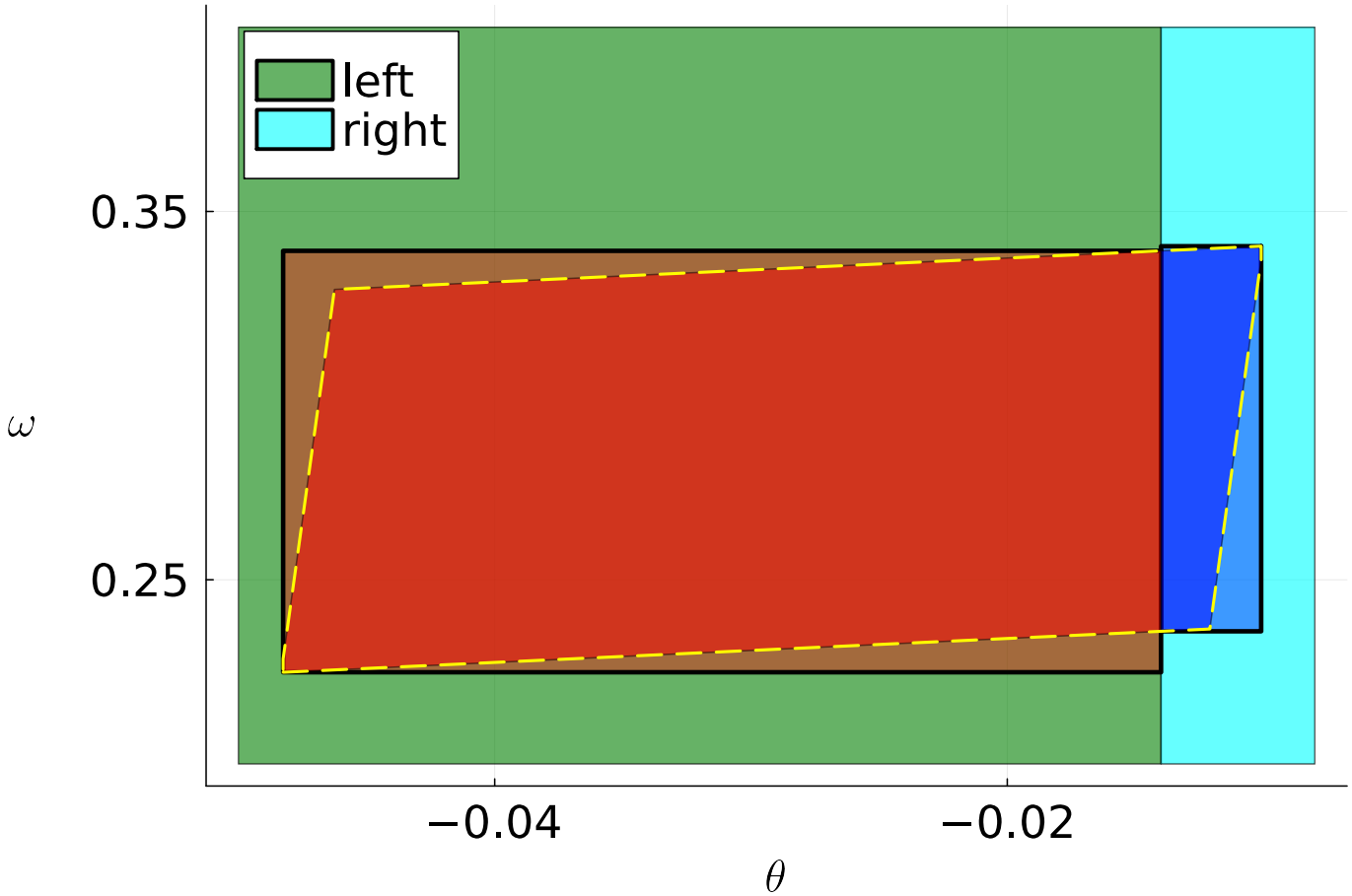




Dealing with set splits

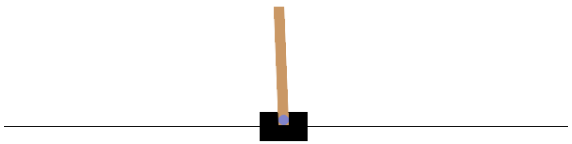
- **Complexity of set representation grows**
- **Interval approximations** to tame complexity



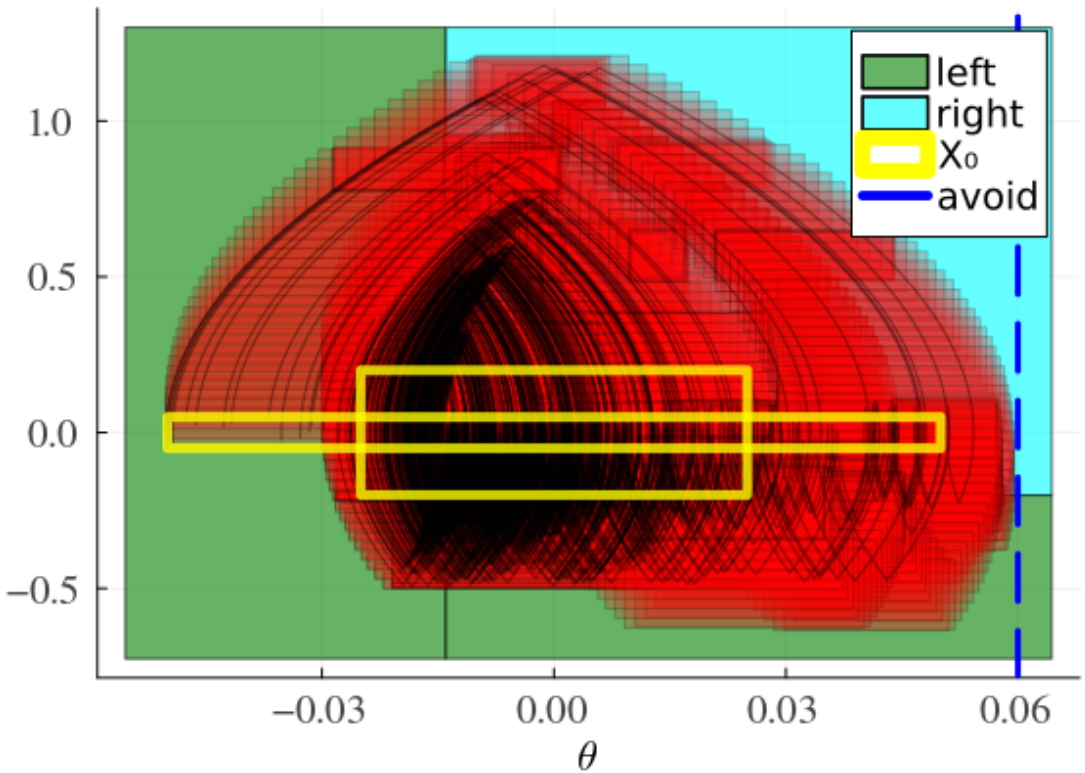


Overview

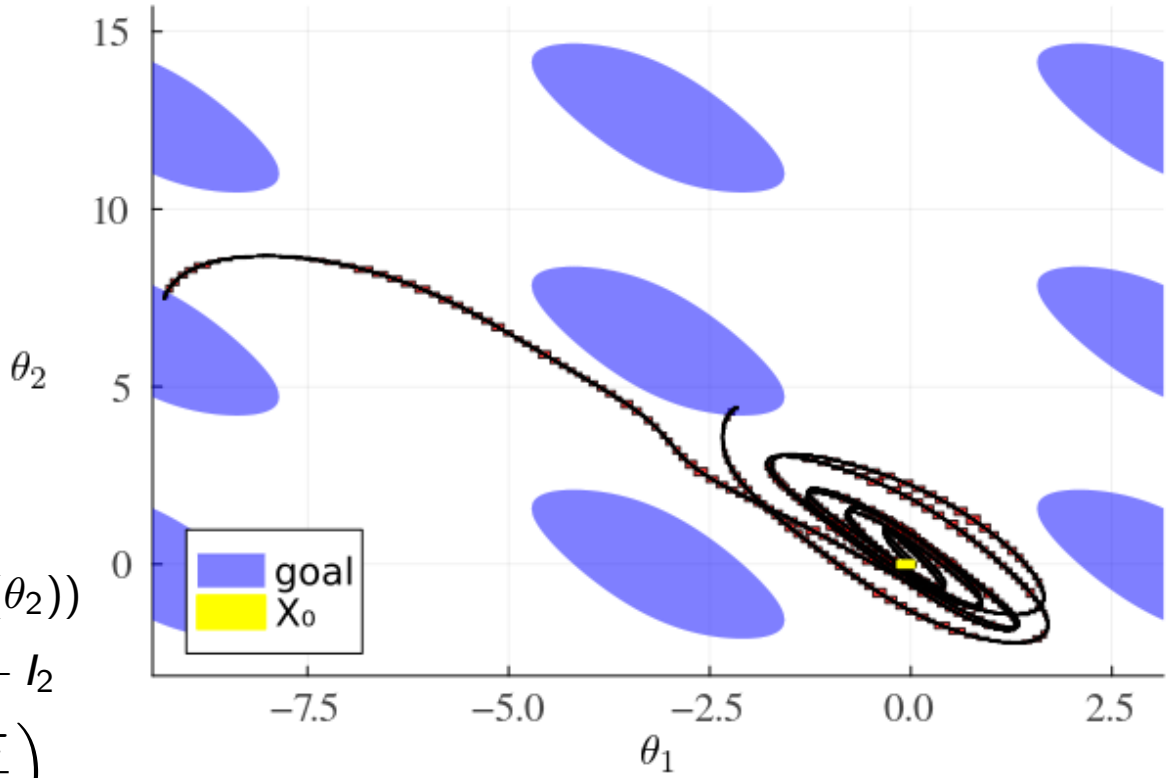
Stabilization policy of a pole on a cart



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



Acrobot policy swinging up to goal height



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

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

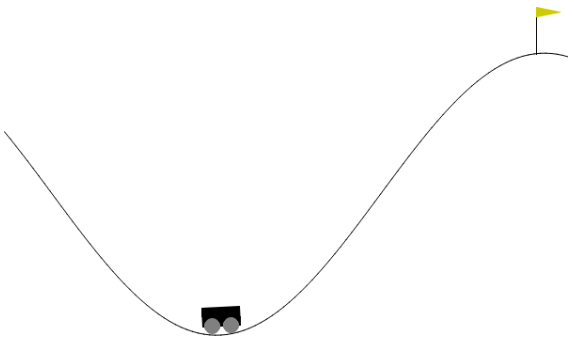
$$d_2 = m_2(lc_2^2 + I_1 lc_2 \cos(\theta_2)) + I_2$$

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

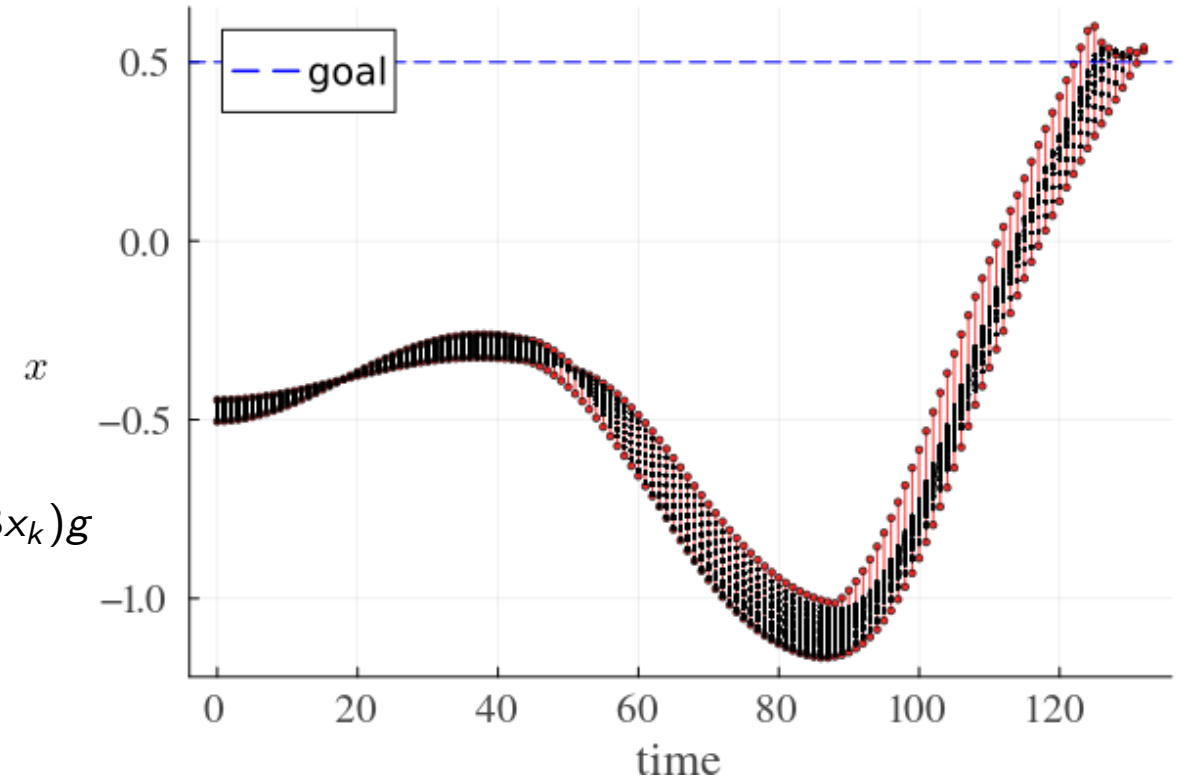
$$\phi_1 = -m_2 I_1 lc_2 \dot{\theta}_2^2 \sin(\theta_2) - 2m_2 I_1 lc_2 \dot{\theta}_2 \dot{\theta}_1 \sin(\theta_2) + (m_1 lc_1 + m_2 I_1) g \cos\left(\theta_1 - \frac{\pi}{2}\right) + \phi_2$$

$$\psi = \left(u + \frac{d_2}{d_1}\phi_1 - m_2 I_1 lc_2 \dot{\theta}_1^2 \sin(\theta_2) - \phi_2\right)\left(m_2 lc_2^2 + I_2 - \frac{d_2^2}{d_1}\right)^{-1}$$

Car policy reaching the top of a mountain



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



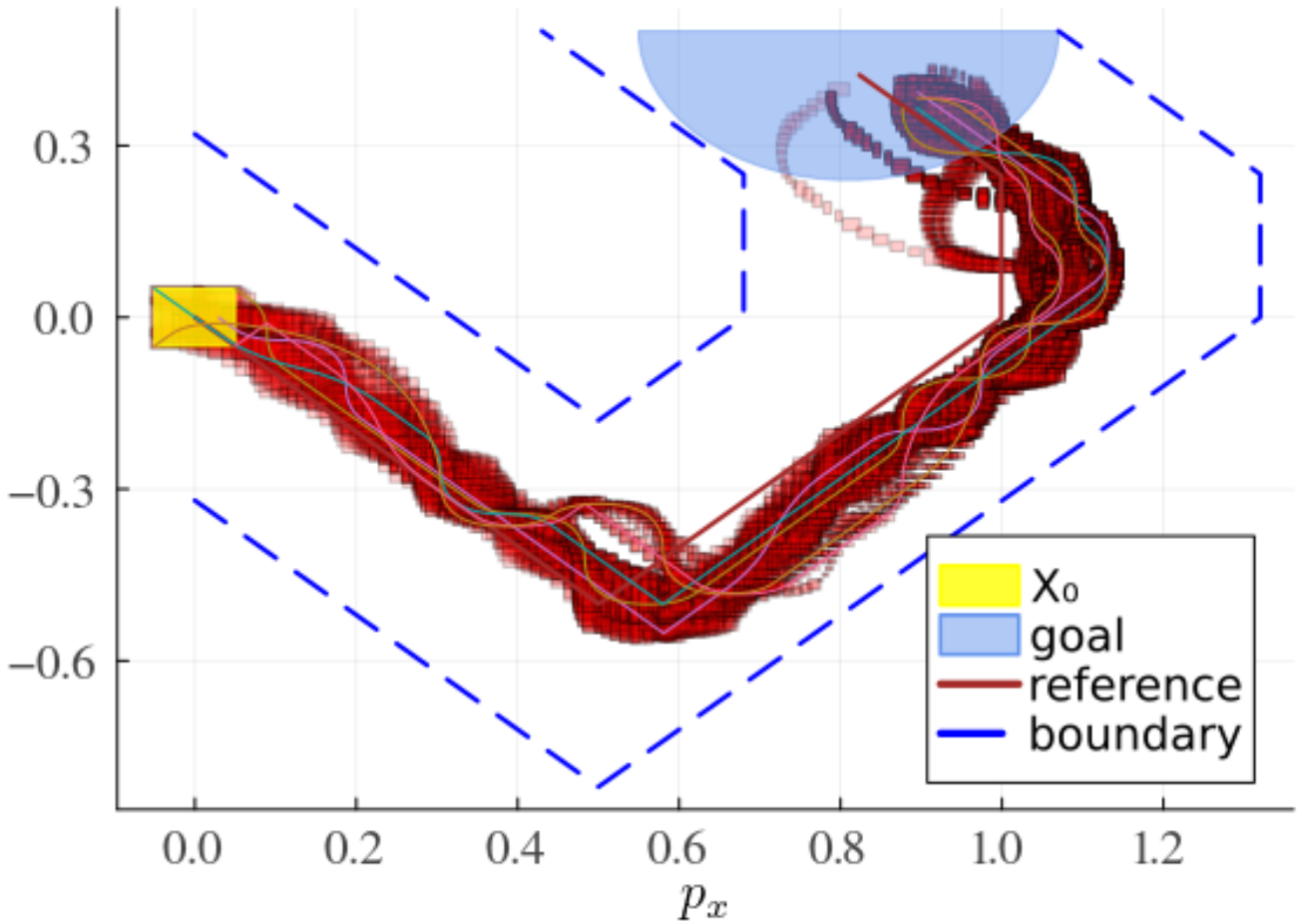
Discrete-time setting

Quadrotor policy following reference trajectory to goal

$\mathcal{T}$: 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 $\mathcal{T}$			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

Conclusion

- **Parametric reachability algorithm** with sufficient conditions for soundness and relative completeness
- **Instantiated algorithm** based on Taylor models and axis-aligned decisions ("$x \leq c$"), exploiting problem structure
- **Undecidable** (unbounded time) resp. **PSPACE-complete** (bounded time)
- **Public implementation** and experimental evaluation
- Orthogonal to **neural-network control systems**
  - Finitely many control actions, but discontinuous
- Future work:
  - Constrained polynomial zonotopes
  - Learning safe decision-tree policies