hybr2d Theory of Hybrid Systems Informatik 2 | RWTH AACHEN UNIVERSITY

Verifying AI-controlled Hybrid Systems

Master's Thesis Presentation

Gabriela Jiang

March 27, 2023

Contents

Neural Networks in Hybrid Systems

- Hybrid systems: systems combining continuous and discrete dynamics
- If too complex, may be regulated by neural network controllers
- Often in engineering and control systems

Neural Networks in Hybrid Systems

- Hybrid systems: systems combining continuous and discrete dynamics
- If too complex, may be regulated by neural network controllers
- Often in engineering and control systems

Examples

Automotive systems, robotics, power systems

Neural Networks in Hybrid Systems

- Hybrid systems: systems combining continuous and discrete dynamics
- If too complex, may be regulated by neural network controllers
- Often in engineering and control systems

Examples

Automotive systems, robotics, power systems

Importance

- Ensure safety and reliability in critical systems
- Predict and prevent undesirable behaviors

Challenges and Current Methods

- Challenges
  - Complexity of modeling and analyzing hybrid systems
  - Black box behavior of neural networks
  - Dealing with over-approximation

Challenges and Current Methods

- Challenges
  - Complexity of modeling and analyzing hybrid systems
  - Black box behavior of neural networks
  - Dealing with over-approximation
- Current methods
  - Model checking
  - Theorem proving
  - Simulation-based approaches

Challenges and Current Methods

- Challenges
  - Complexity of modeling and analyzing hybrid systems
  - Black box behavior of neural networks
  - Dealing with over-approximation
- Current methods
  - Model checking
  - Theorem proving
  - Simulation-based approaches
- Limitations of current methods
  - Limited scalability
  - Assumptions about system behavior
  - Lack of guarantees for certain properties

Contents

Two Types of Hybrid Systems

- Autonomous system

$$\dot{x}(t) = Ax(t)$$

Two Types of Hybrid Systems

- Autonomous system

$$\dot{x}(t) = Ax(t)$$

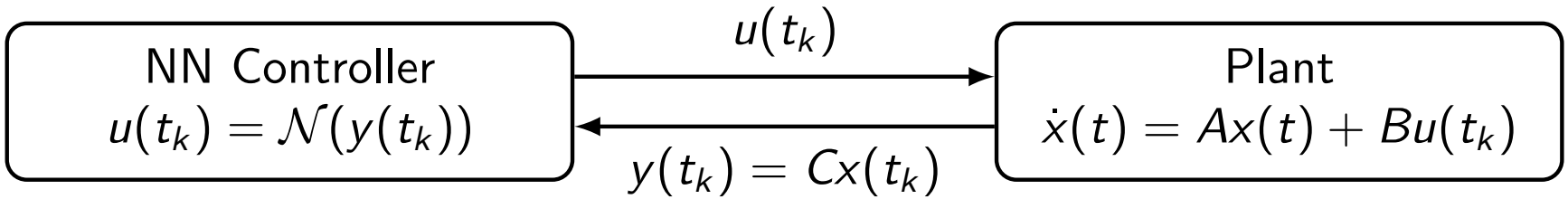
- Non-autonomous system

$$\dot{x}(t) = Ax(t) + Bu(t)$$

where x are the variables, A is the flow matrix at time t, u the external inputs.

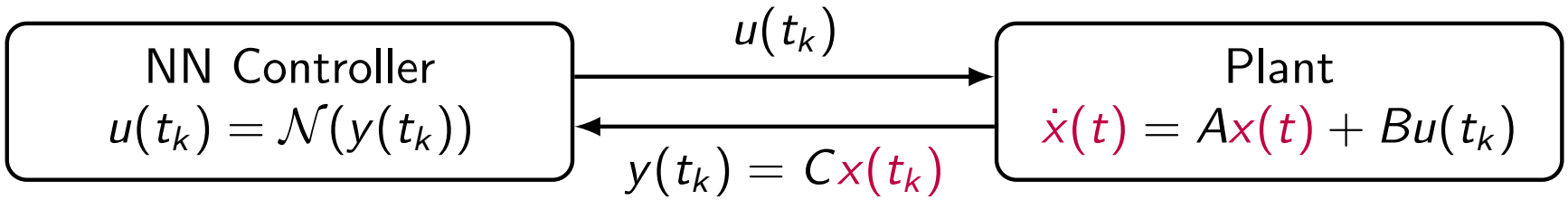
Neural Network Control System

$$(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$$



Neural Network Control System

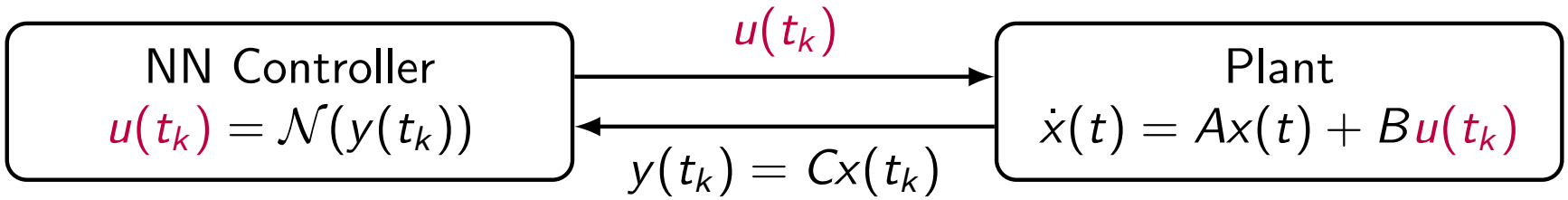
$$(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$$



$X = \{x_1, \ldots, x_n\}$ is the set of finitely many state variables

Neural Network Control System

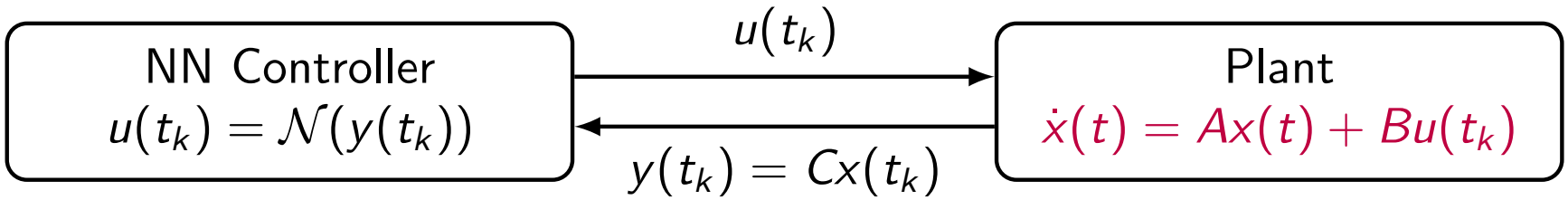
$$(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$$



$U = \{u_1, \ldots, u_m\}$ the set of finitely many control variables

Neural Network Control System

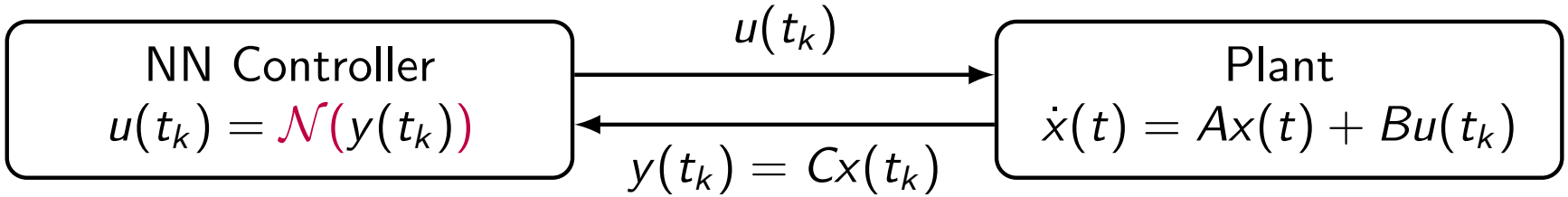
$$(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$$



$F : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ the ODE defining the continuous dynamics of the plant

Neural Network Control System

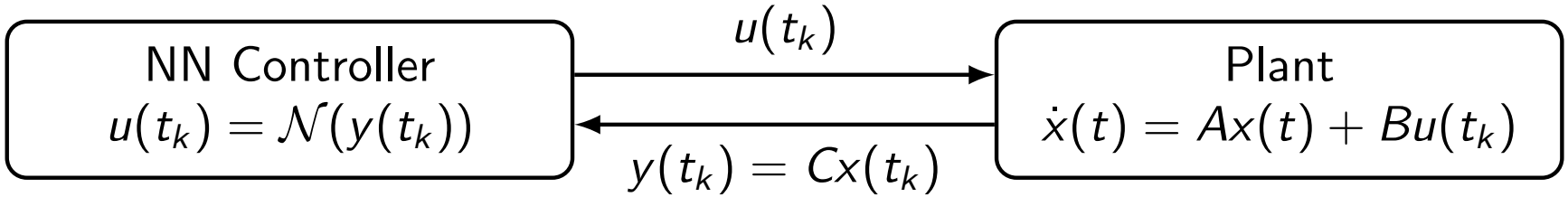
$$(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$$



$\mathcal{N} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ denotes the input-output mapping by the NN controller

Neural Network Control System

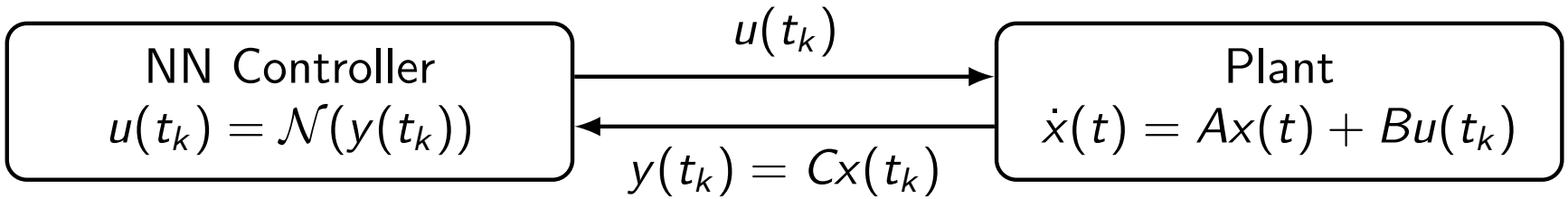
$$(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$$



$\delta \in \mathbb{R}_{>0}$ denotes the control step size

Neural Network Control System

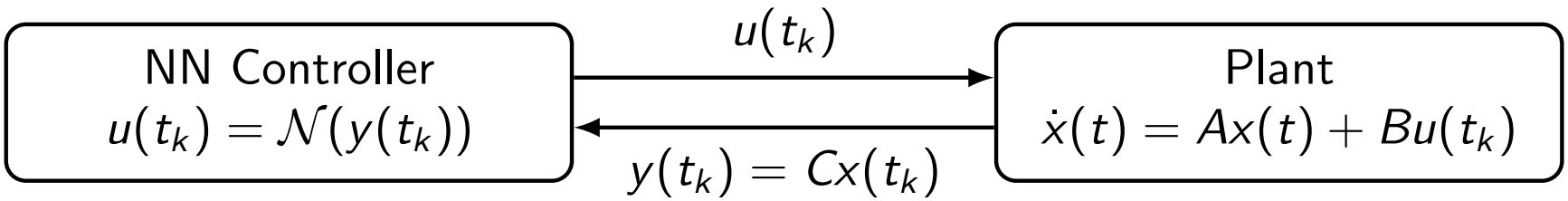
$$(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$$



$\mathcal{X}_0 \subset \mathbb{R}^n$ is the set of the initial values of the state variables

Neural Network Control System

Modeled as

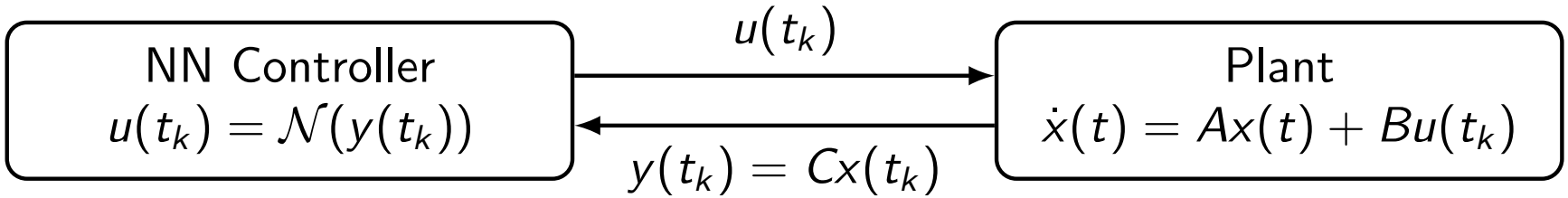


Neural network

Simplified
hybrid automaton

Neural Network Control System

Reachability analysis



Exact star-based
NN reachability

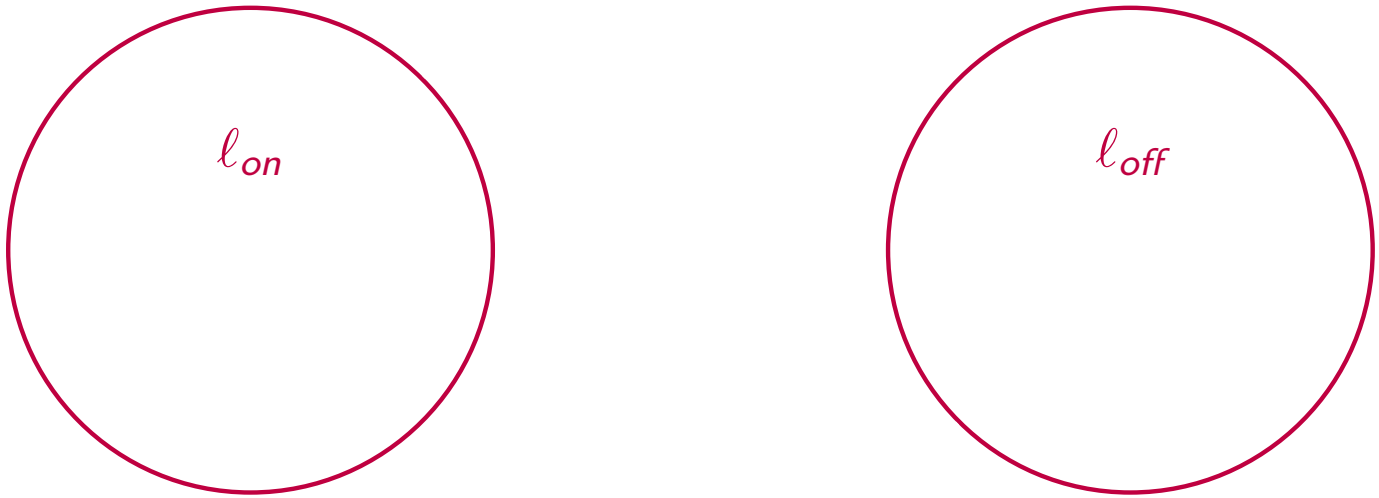
Flowpipe construction

Hybrid Automaton of the Thermostat [Alur et al. 1995]

$$(\mathit{Loc}, \mathit{Var}, \mathit{Con}, \mathit{Lab}, \mathit{Edge}, \mathit{Act}, \mathit{Inv}, \mathit{Init})$$

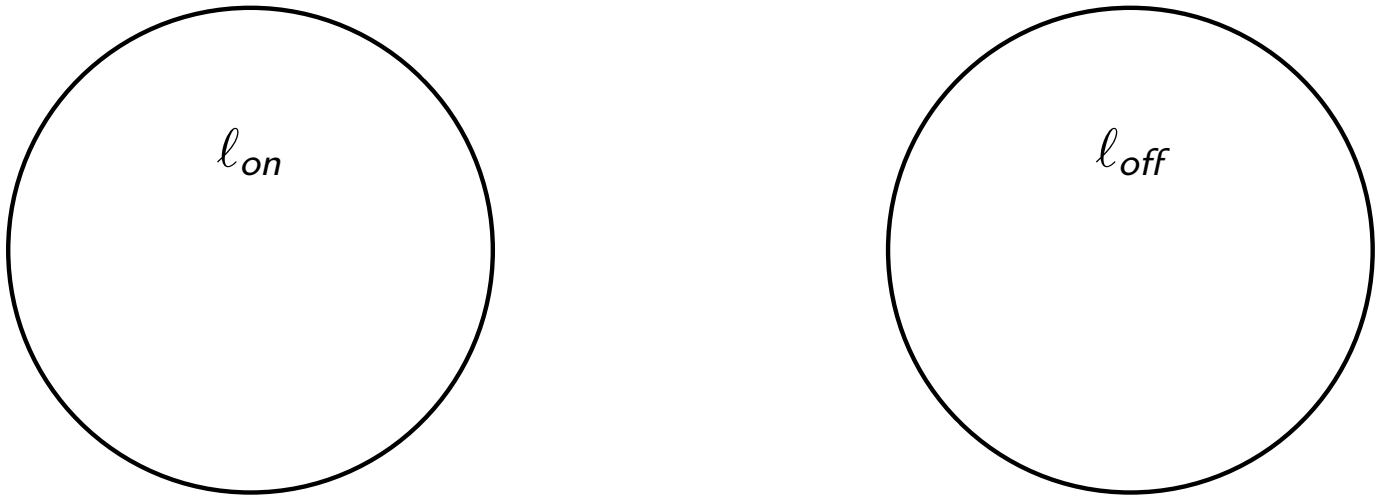
Hybrid Automaton of the Thermostat [Alur et al. 1995]

$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$



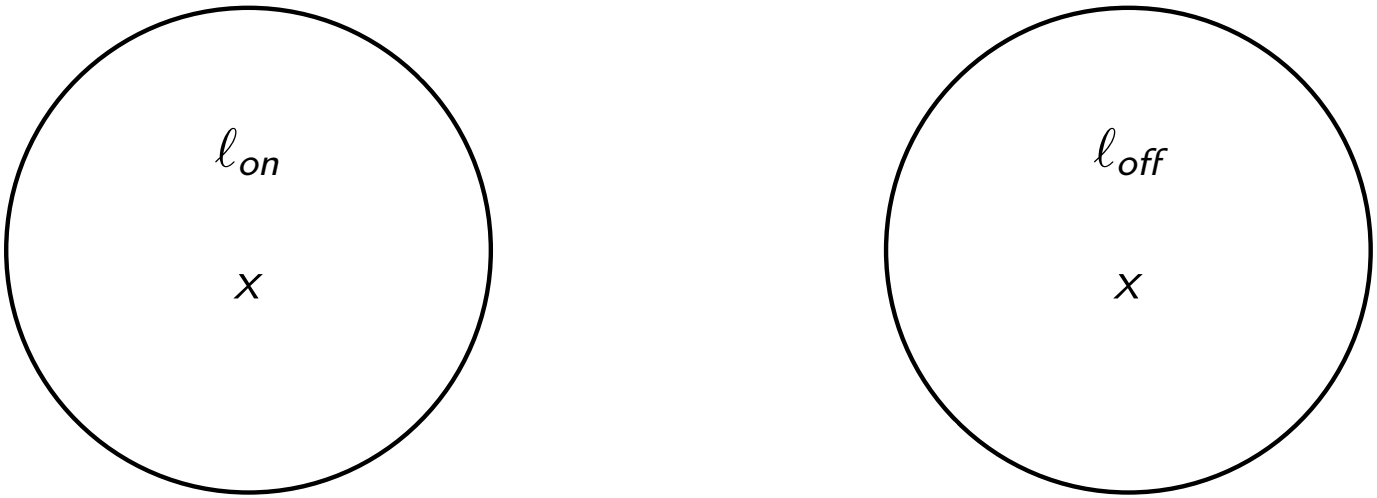
Hybrid Automaton of the Thermostat [Alur et al. 1995]

$$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$$



Hybrid Automaton of the Thermostat [Alur et al. 1995]

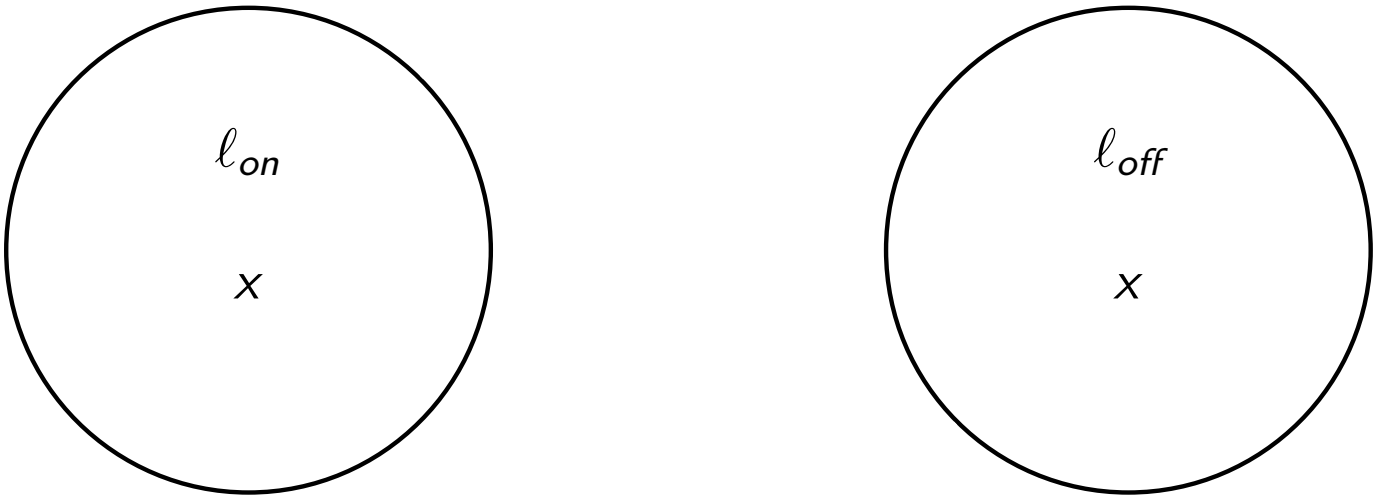
$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$



Hybrid Automaton of the Thermostat [Alur et al. 1995]

$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$

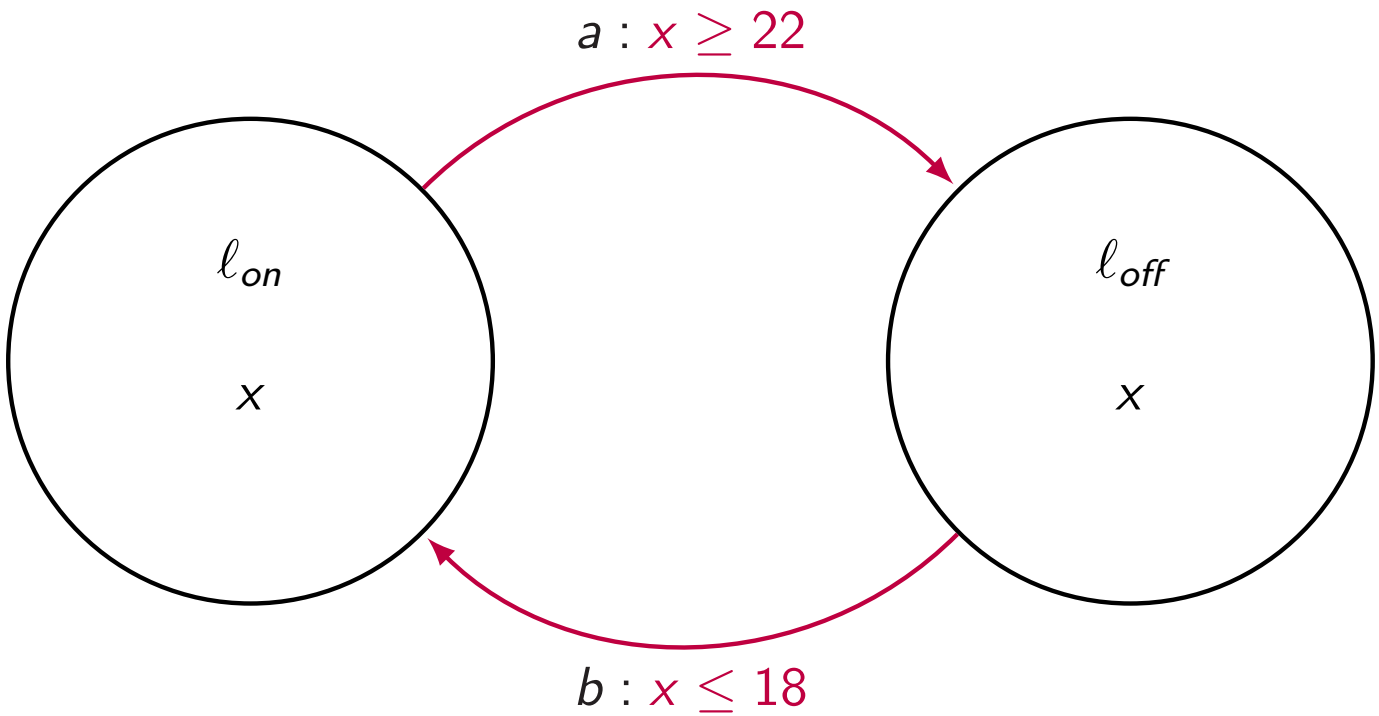
a :



b :

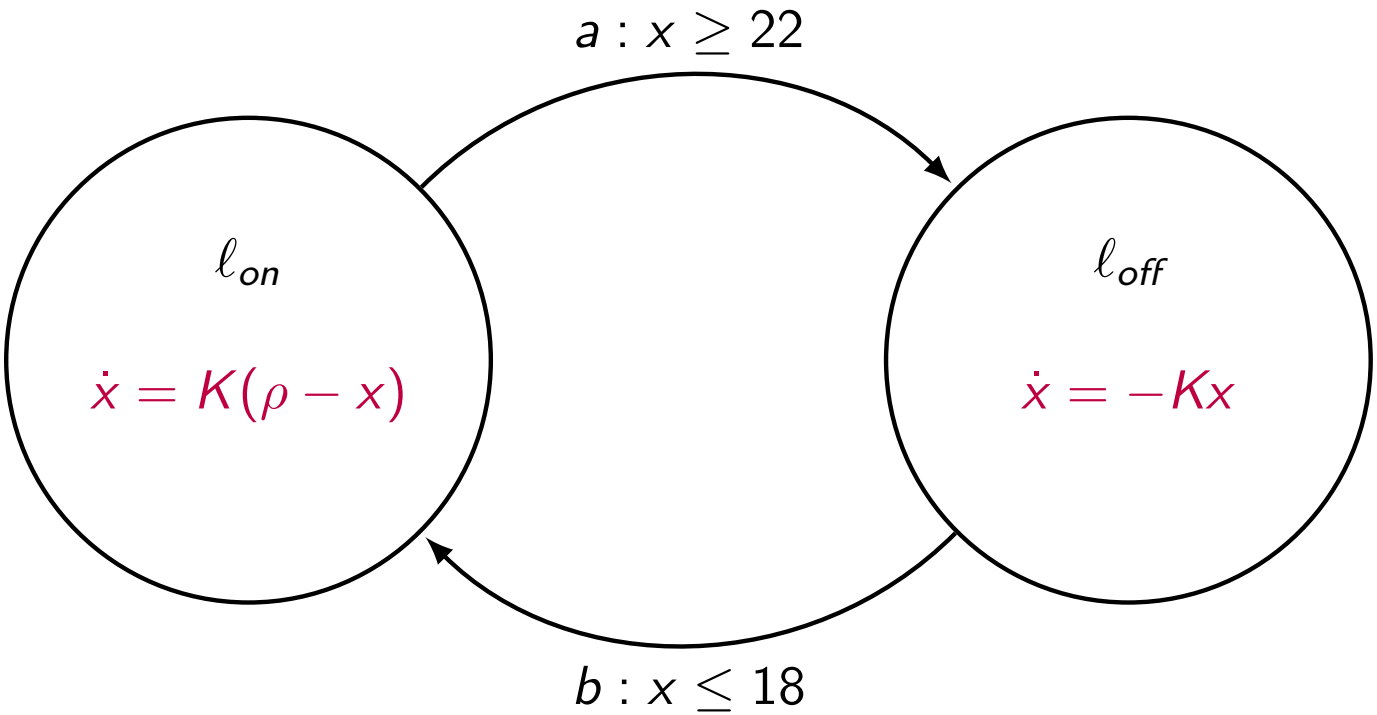
Hybrid Automaton of the Thermostat [Alur et al. 1995]

$$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$$



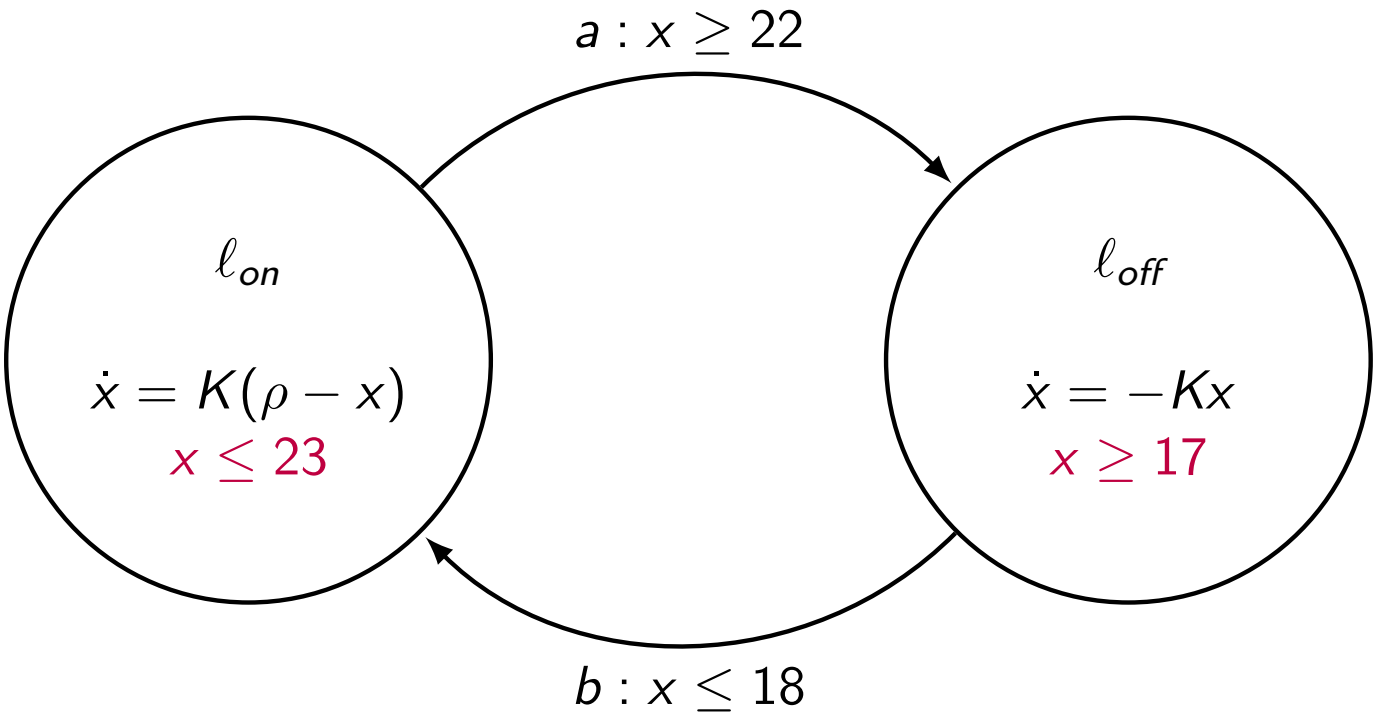
Hybrid Automaton of the Thermostat [Alur et al. 1995]

$$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$$



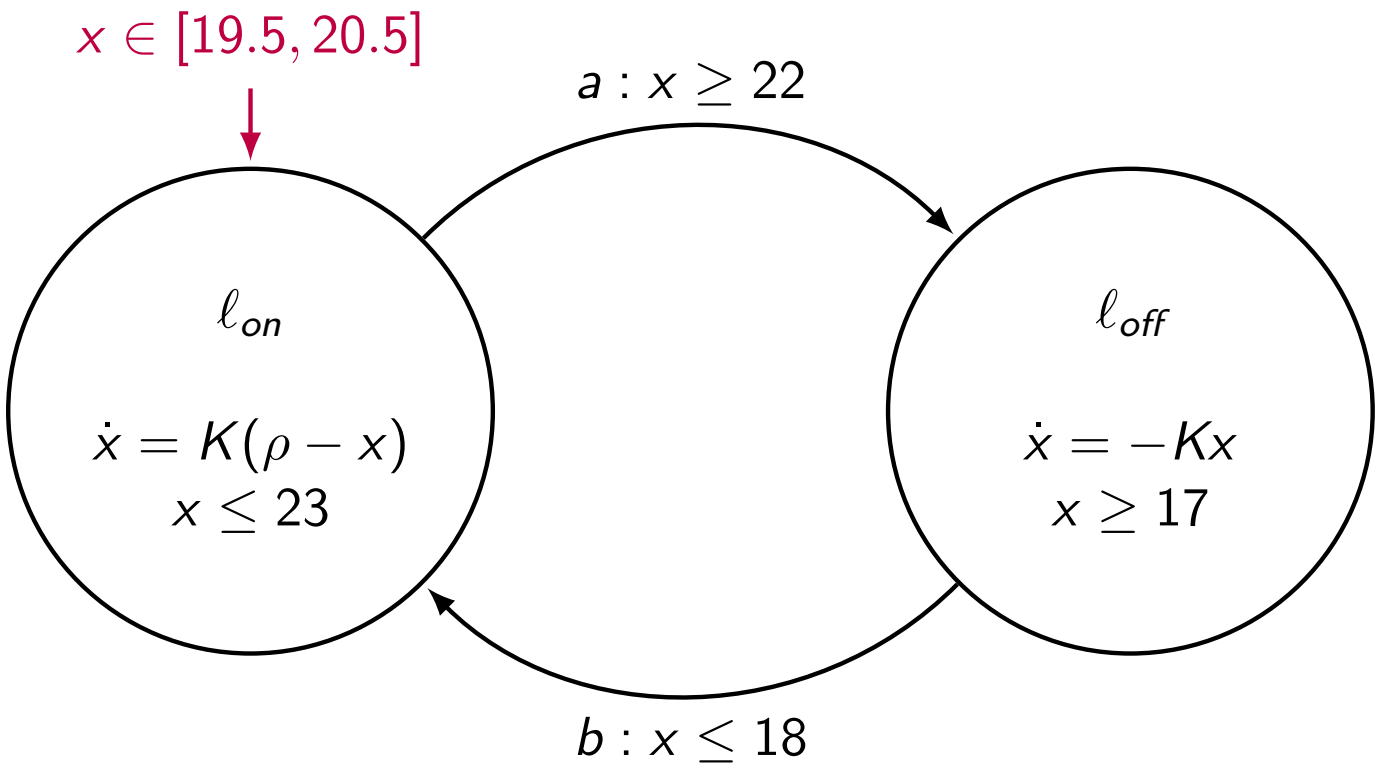
Hybrid Automaton of the Thermostat [Alur et al. 1995]

$$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$$



Hybrid Automaton of the Thermostat [Alur et al. 1995]

$$(Loc, Var, Con, Lab, Edge, Act, Inv, Init)$$



Reachability Analysis of Hybrid Automata

- Exact reachable set

$$\mathcal{R}(I) = \{\sigma \in \Sigma \mid \exists \sigma' \in I . \sigma' \rightarrow^* \sigma\}$$

not computable [Henzinger et al. 1998]

Reachability Analysis of Hybrid Automata

- Exact reachable set

$$\mathcal{R}(I) = \{\sigma \in \Sigma \mid \exists \sigma' \in I . \sigma' \rightarrow^* \sigma\}$$

  not computable [Henzinger et al. 1998]
- Hence over-approximate approaches where

$$\mathcal{R}_{[0,T]}(I) \subseteq \Omega_{[0,T]}(I)$$

Reachability Analysis of Hybrid Automata

- Exact reachable set

$$\mathcal{R}(I) = \{\sigma \in \Sigma \mid \exists \sigma' \in I . \sigma' \rightarrow^* \sigma\}$$

  not computable [Henzinger et al. 1998]
- Hence over-approximate approaches where

$$\mathcal{R}_{[0,T]}(I) \subseteq \Omega_{[0,T]}(I)$$

- Such as flowpipe construction

Reachability Analysis of Hybrid Automata

- Exact reachable set

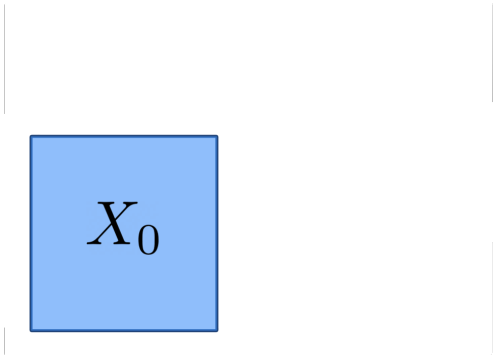
$$\mathcal{R}(I) = \{\sigma \in \Sigma \mid \exists \sigma' \in I . \sigma' \rightarrow^* \sigma\}$$

  not computable [Henzinger et al. 1998]
- Hence over-approximate approaches where

$$\mathcal{R}_{[0,T]}(I) \subseteq \Omega_{[0,T]}(I)$$

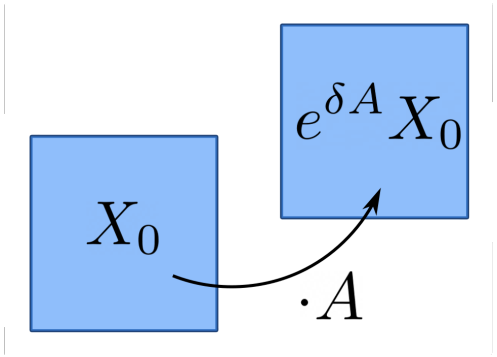
- Such as flowpipe construction
- In the following, $T = K\delta$

Flowpipe Construction



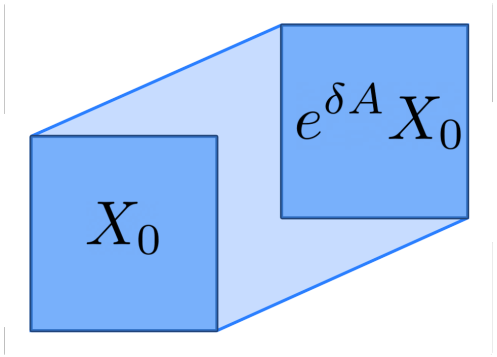
X_0

Flowpipe Construction



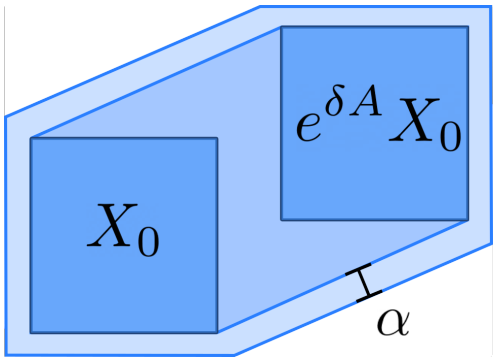
$$X_0 \cup e^{\delta A} X_0$$

Flowpipe Construction



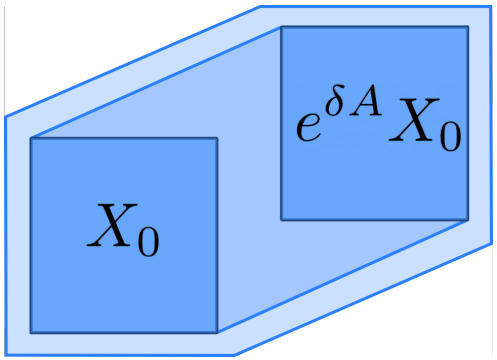
$$conv(X_0 \cup e^{\delta A} X_0)$$

Flowpipe Construction



$$\Omega_0 = conv(X_0 \cup e^{\delta A} X_0) \oplus \mathcal{B}_\alpha$$

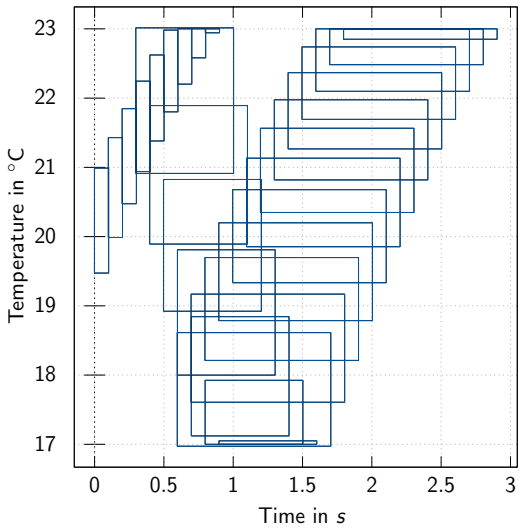
Flowpipe Construction



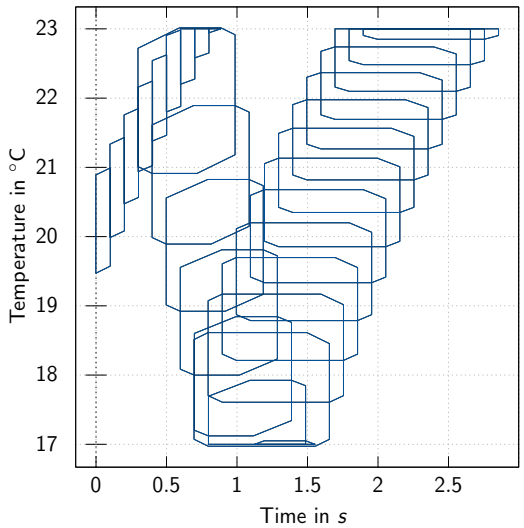
$$\Omega_0 = conv(X_0 \cup e^{\delta A} X_0) \oplus \mathcal{B}_\alpha$$

$$\Omega_{i+1} = e^{\delta A} \Omega_i$$

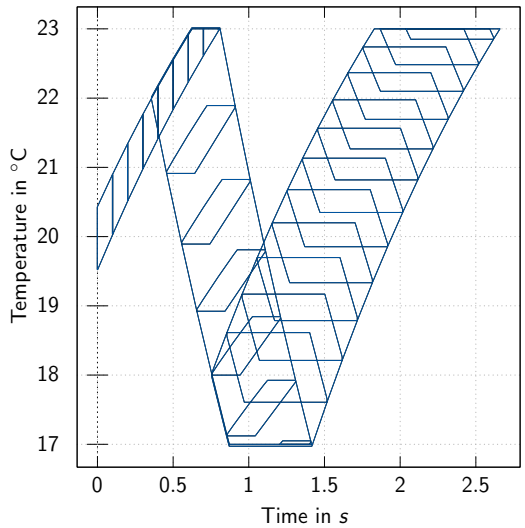
State Set Representations



(a) Box



(b) $\mathcal{H}$-polytope



(c) $\mathcal{V}$-polytope

Reachability Analysis of Neural Networks

Definition (Reachable set of an FNN)

- Given
  - Input set $I = \left\{x \in \mathbb{R}^d \mid Ax \leq b\right\}$ with $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$
  - k-layer FNN $\mathcal{N} = \{L_1, \cdots, L_k\}$

Reachability Analysis of Neural Networks

Definition (Reachable set of an FNN)

- Given
  - Input set $I = \left\{x \in \mathbb{R}^d \mid Ax \leq b\right\}$ with $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$
  - k-layer FNN $\mathcal{N} = \{L_1, \cdots, L_k\}$
- Reachable set $\mathcal{N}(I) = \mathcal{R}_{L_k}$ is defined by

$$\mathcal{R}_{L_0} := I$$
$$\mathcal{R}_{L_i} := \left\{y_i \mid y_i = \phi_i\left(W_i y_{i-1} + b_i\right), y_{i-1} \in \mathcal{R}_{L_{i-1}}\right\}, \quad i = 1, \ldots, k$$

Reachability Analysis of Neural Networks

Definition (Reachable set of an FNN)

- Given
  - Input set $I = \left\{x \in \mathbb{R}^d \mid Ax \leq b\right\}$ with $A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n$
  - k-layer FNN $\mathcal{N} = \{L_1, \cdots, L_k\}$
- Reachable set $\mathcal{N}(I) = \mathcal{R}_{L_k}$ is defined by

$$\mathcal{R}_{L_0} := I$$
$$\mathcal{R}_{L_i} := \left\{y_i \mid y_i = \phi_i\left(W_i y_{i-1} + b_i\right), y_{i-1} \in \mathcal{R}_{L_{i-1}}\right\}, \quad i = 1, \ldots, k$$

Approach [Tran 2020]

Exact analysis for FNNs with piece-wise linear activations.

Star Set Representation

Definition (Generalized star set)

A generalized star set is a tuple $\Theta = \langle c, V, P \rangle$ where

Star Set Representation

Definition (Generalized star set)

A generalized star set is a tuple $\Theta = \langle c, V, P \rangle$ where

- $c \in \mathbb{R}^d$ is the center,

Star Set Representation

Definition (Generalized star set)

A generalized star set is a tuple $\Theta = \langle c, V, P \rangle$ where

- $c \in \mathbb{R}^d$ is the center,
- $V = \{v_1, \ldots, v_m\} \subseteq \mathbb{R}^d$ a set of basis vectors, and

Star Set Representation

Definition (Generalized star set)

A generalized star set is a tuple $\Theta = \langle c, V, P \rangle$ where

- $c \in \mathbb{R}^d$ is the center,
- $V = \{v_1, \ldots, v_m\} \subseteq \mathbb{R}^d$ a set of basis vectors, and
- $P : \mathbb{R}^m \to \{\top, \bot\}$ a predicate.

Star Set Representation

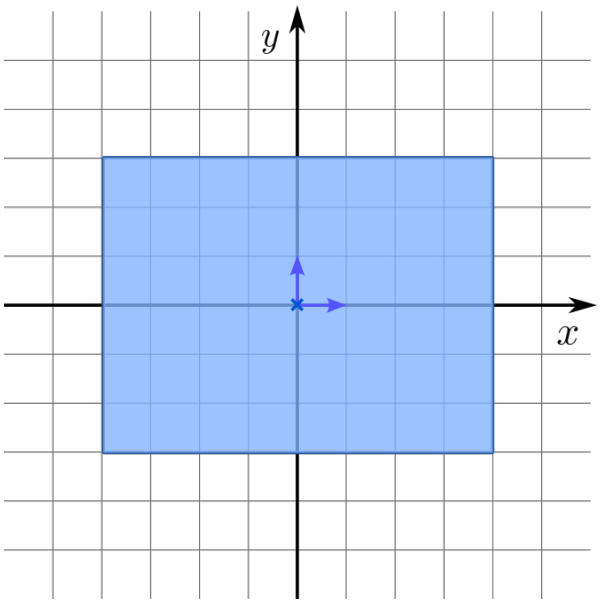
Definition (Generalized star set)

A generalized star set is a tuple $\Theta = \langle c, V, P \rangle$ where

- $c \in \mathbb{R}^d$ is the center,
- $V = \{v_1, \ldots, v_m\} \subseteq \mathbb{R}^d$ a set of basis vectors, and
- $P : \mathbb{R}^m \to \{\top, \bot\}$ a predicate.

Efficient w.r.t. computing affine mappings and halfspace intersections.

Example Starset



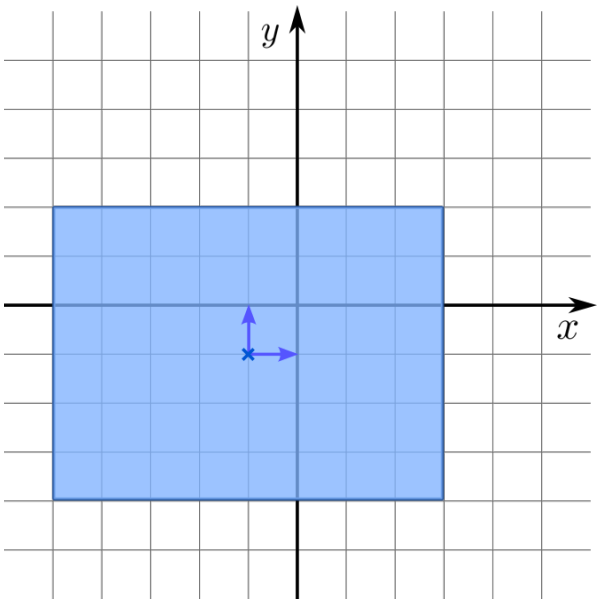
$\Theta = \langle c, V, P \rangle$

$$c = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

$$V = \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\},$$

$$P = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} -4 \\ 4 \\ -3 \\ 3 \end{bmatrix}$$

Example Starset

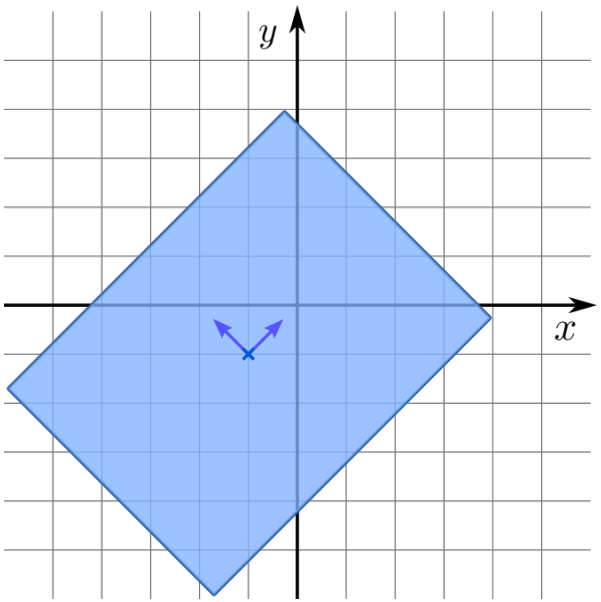


$\Theta + t$

Translate by

$$t = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Example Starset

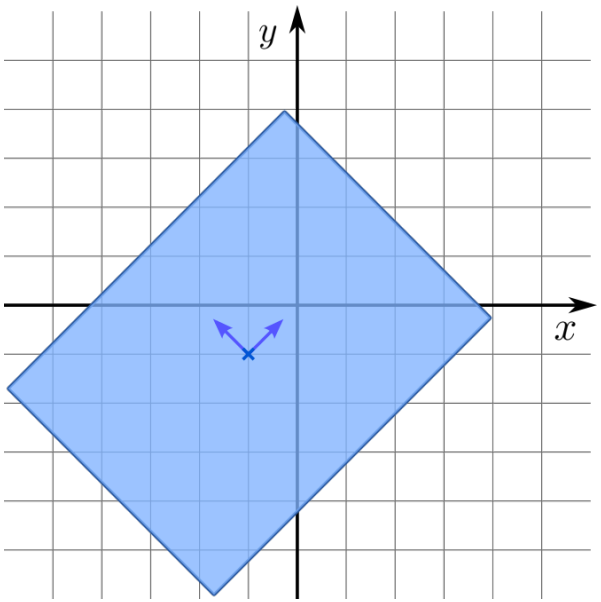


$$\Theta' = M(\Theta + t)$$

Rotate by

$$M = \begin{bmatrix} \cos 45^\circ & -\sin 45^\circ \\ \sin 45^\circ & \cos 45^\circ \end{bmatrix}$$

Example Starset



$\Theta' = \langle c', V', P \rangle$

$$c' = \begin{bmatrix} -1 \\ -1 \end{bmatrix},$$

$$V' = \left\{ \begin{bmatrix} \cos 45^\circ \\ \sin 45^\circ \end{bmatrix}, \begin{bmatrix} -\sin 45^\circ \\ \cos 45^\circ \end{bmatrix} \right\},$$

$$P = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} -4 \\ 4 \\ -3 \\ 3 \end{bmatrix}$$

Contents

Goal

- Develop techniques for verifying the safety of NNCS

Goal

- Develop techniques for verifying the safety of NNCS
- Focus on non-autonomous hybrid systems with discrete interactions

Goal

- Develop techniques for verifying the safety of NNCS
- Focus on non-autonomous hybrid systems with discrete interactions
- Keep over-approximation errors small

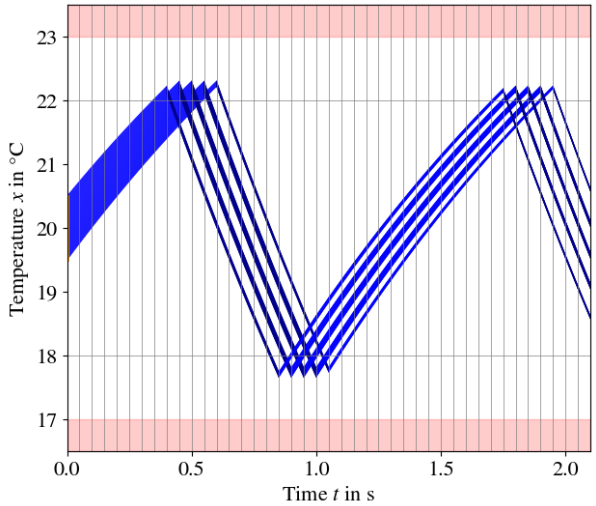
Interaction

1. Continuous
   - Independent of the time step δ

Interaction

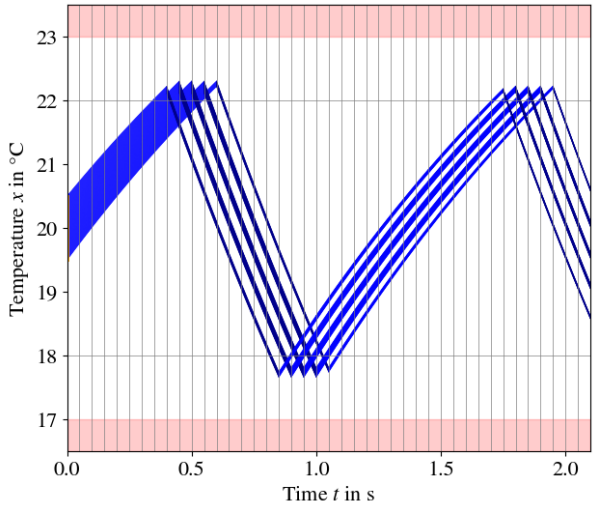
1. Continuous
   - Independent of the time step δ
2. Discrete
   - Irregular: time steps $\delta_1, \delta_2, \ldots$
   - Regular: time step δ

The Step Size δ is Crucial

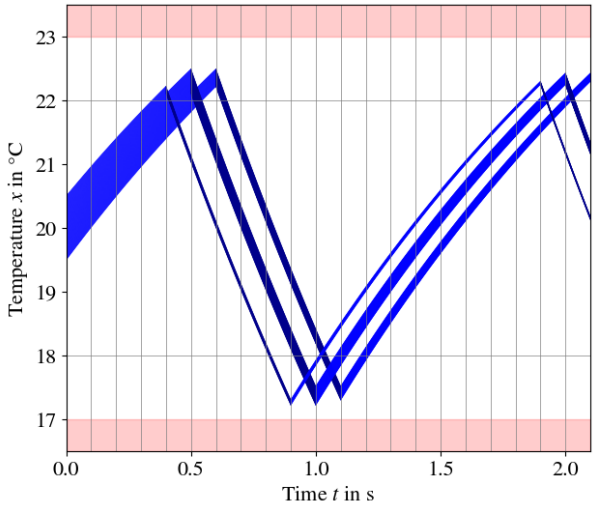


(a) $\delta = 0.01$

The Step Size δ is Crucial

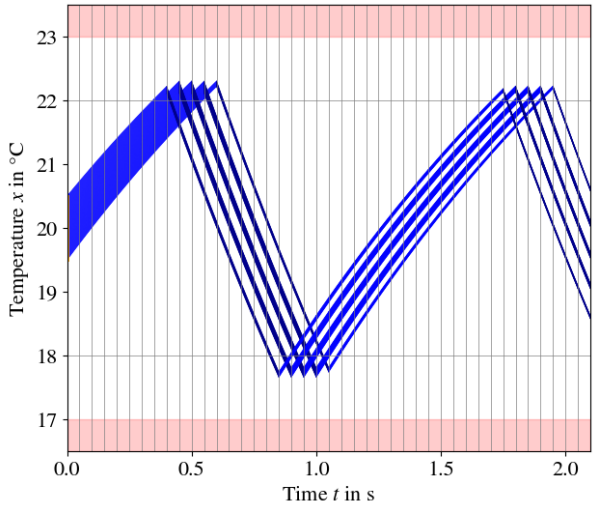


(a) $\delta = 0.01$

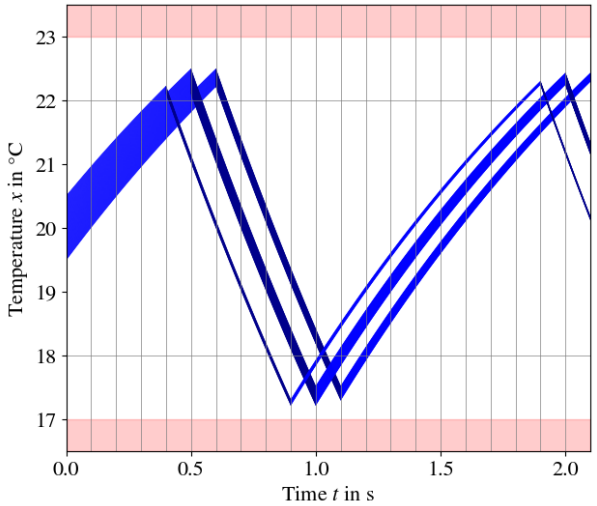


(b) $\delta = 0.1$

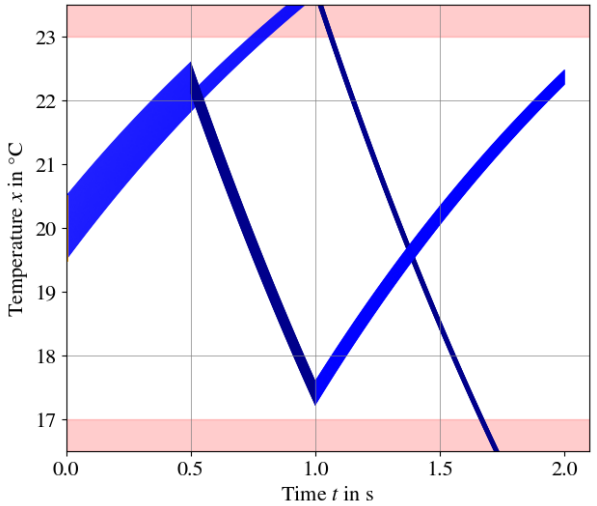
The Step Size δ is Crucial



(a) $\delta = 0.01$

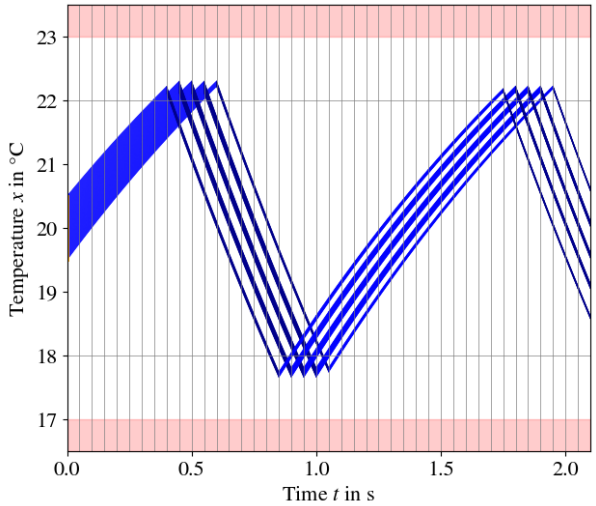


(b) $\delta = 0.1$

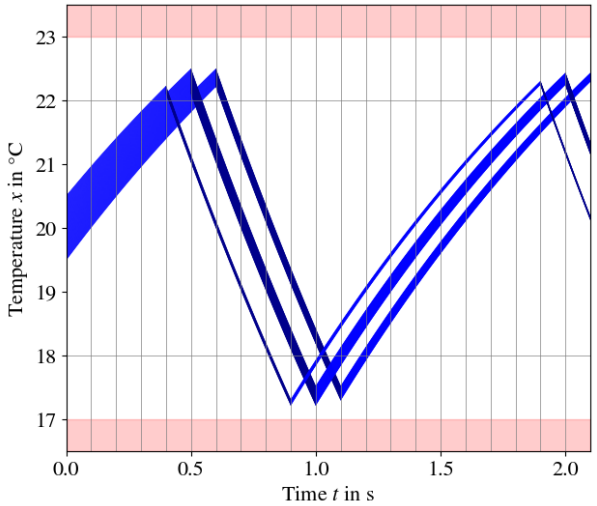


(c) $\delta = 0.5$

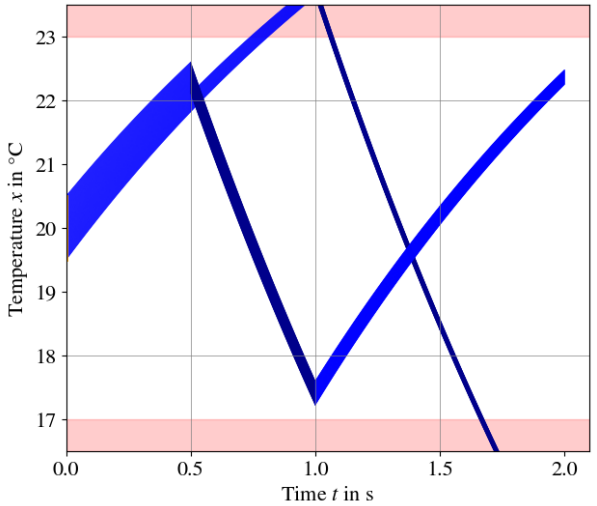
The Step Size δ is Crucial



(a) $\delta = 0.01$



(b) $\delta = 0.1$



(c) $\delta = 0.5$

- Trade-off between safety and computational effort

Conceptualization

- Challenges

Conceptualization

- Challenges
  - Devise a suitable controller to ensure the system's safety

Conceptualization

- Challenges
  - Devise a suitable controller to ensure the system's safety
  - Need benchmark for validating method

Conceptualization

- Challenges
  - Devise a suitable controller to ensure the system's safety
  - Need benchmark for validating method
- Rationale

Conceptualization

- Challenges
  - Devise a suitable controller to ensure the system's safety
  - Need benchmark for validating method
- Rationale
  - No suitable benchmarks

Conceptualization

- Challenges
  - Devise a suitable controller to ensure the system's safety
  - Need benchmark for validating method
- Rationale
  - No suitable benchmarks
  - Create own benchmarks

Conceptualization

- Challenges
  - Devise a suitable controller to ensure the system's safety
  - Need benchmark for validating method
- Rationale
  - No suitable benchmarks
  - Create own benchmarks
  - Train controller to learn behavior of hybrid automata

Conceptualization

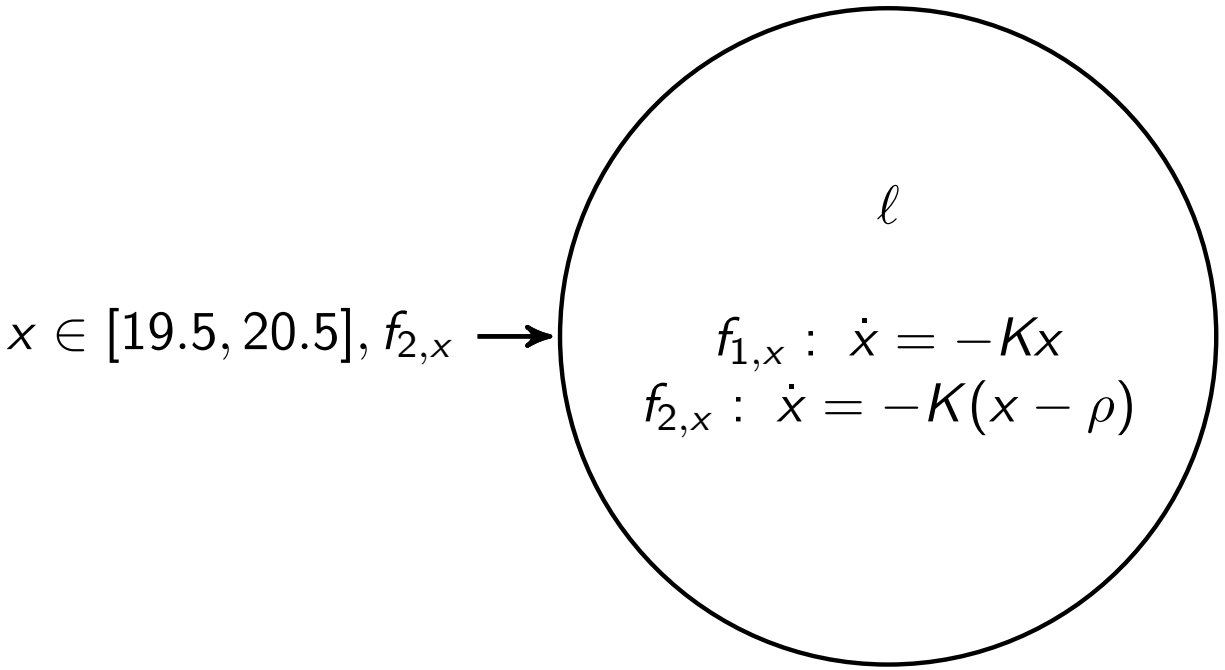
- Challenges
  - Devise a suitable controller to ensure the system's safety
  - Need benchmark for validating method
- Rationale
  - No suitable benchmarks
  - Create own benchmarks
  - Train controller to learn behavior of hybrid automata

Conceptualization

- Challenges
  - Devise a suitable controller to ensure the system's safety
  - Need benchmark for validating method
- Rationale
  - No suitable benchmarks
  - Create own benchmarks
  - Train controller to learn behavior of hybrid automata
  - Reduce hybrid automaton to a simplified automaton

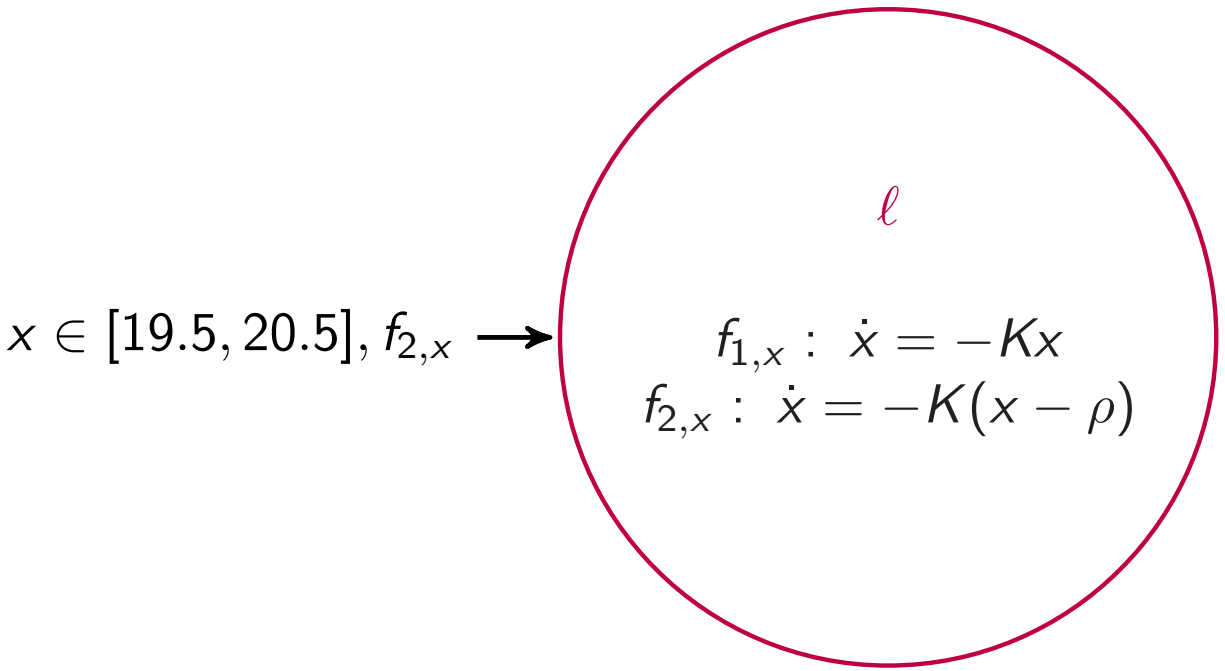
Simplified Hybrid Automaton

$$\mathcal{H}' = (Loc', Var', Lab', Act', Init')$$



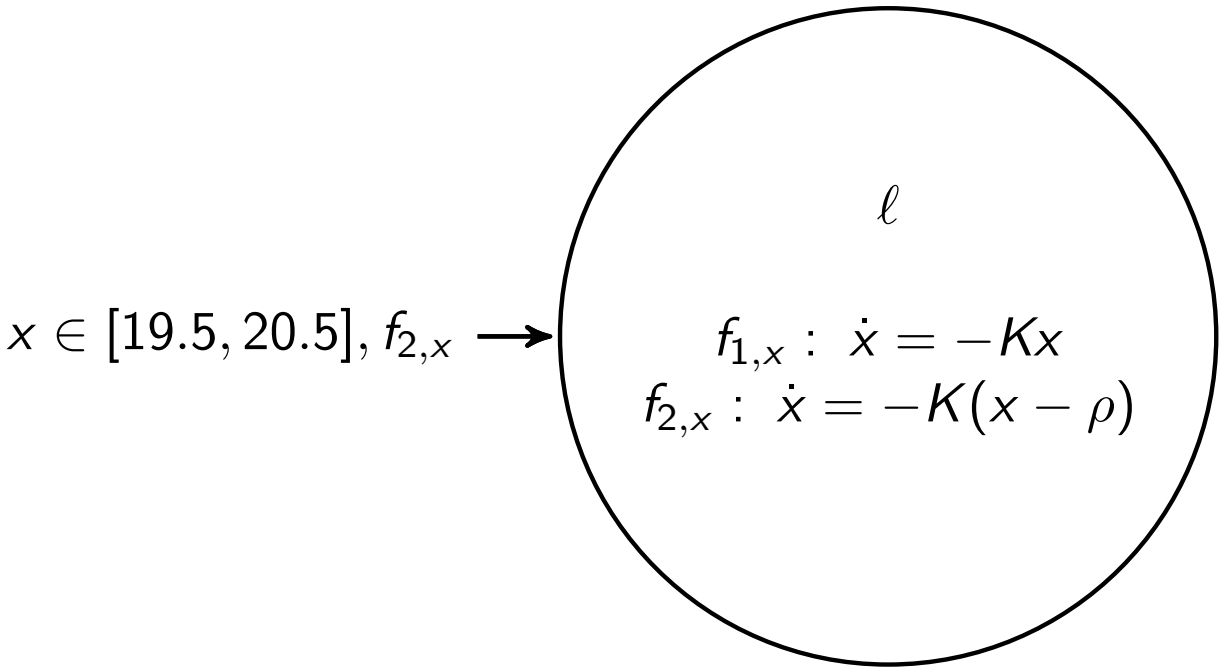
Simplified Hybrid Automaton

$$\mathcal{H}' = (Loc', Var', Lab', Act', Init')$$



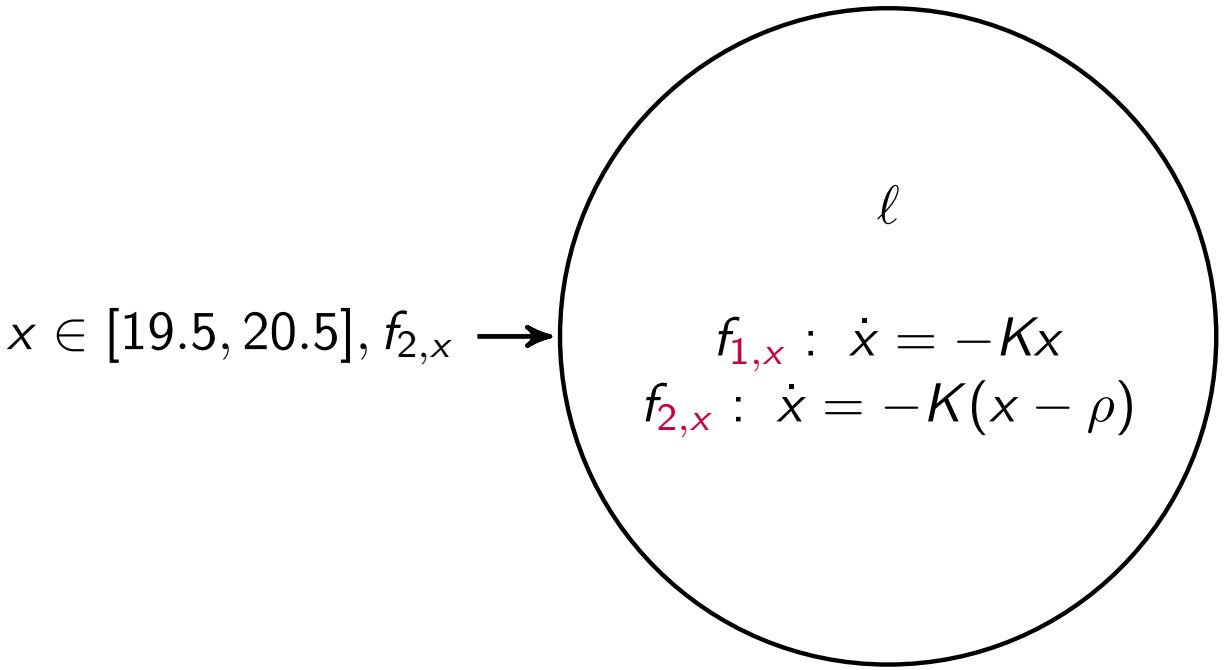
Simplified Hybrid Automaton

$$\mathcal{H}' = (Loc', Var', Lab', Act', Init')$$



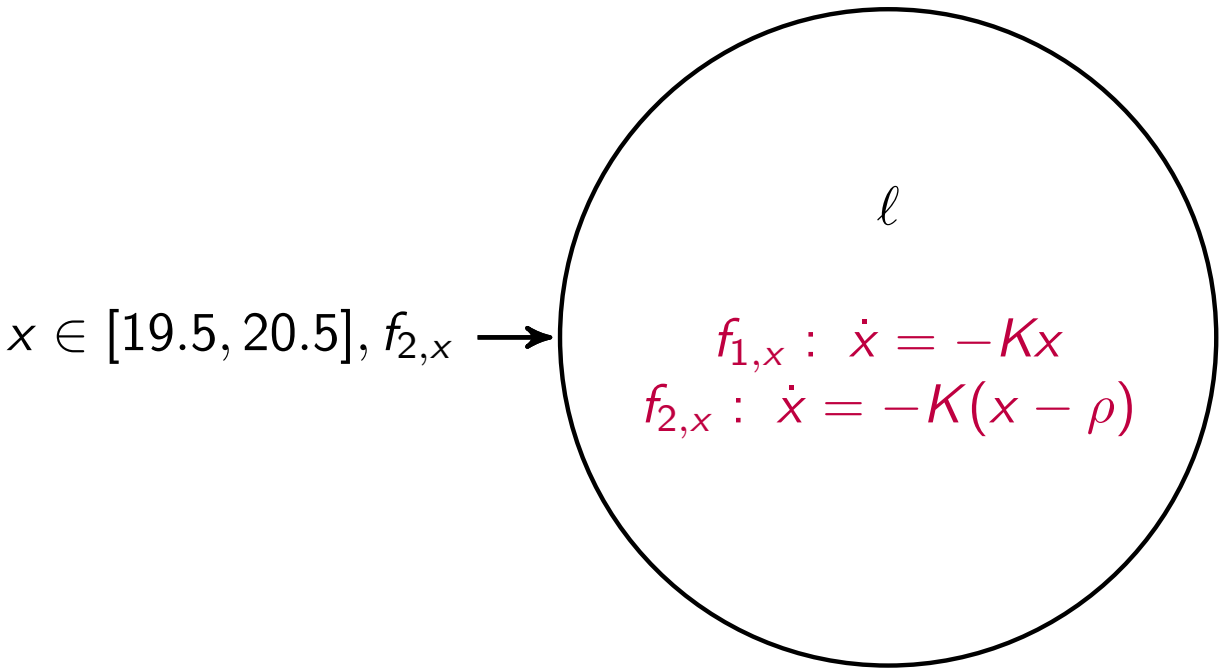
Simplified Hybrid Automaton

$$\mathcal{H}' = (Loc', Var', Lab', Act', Init')$$



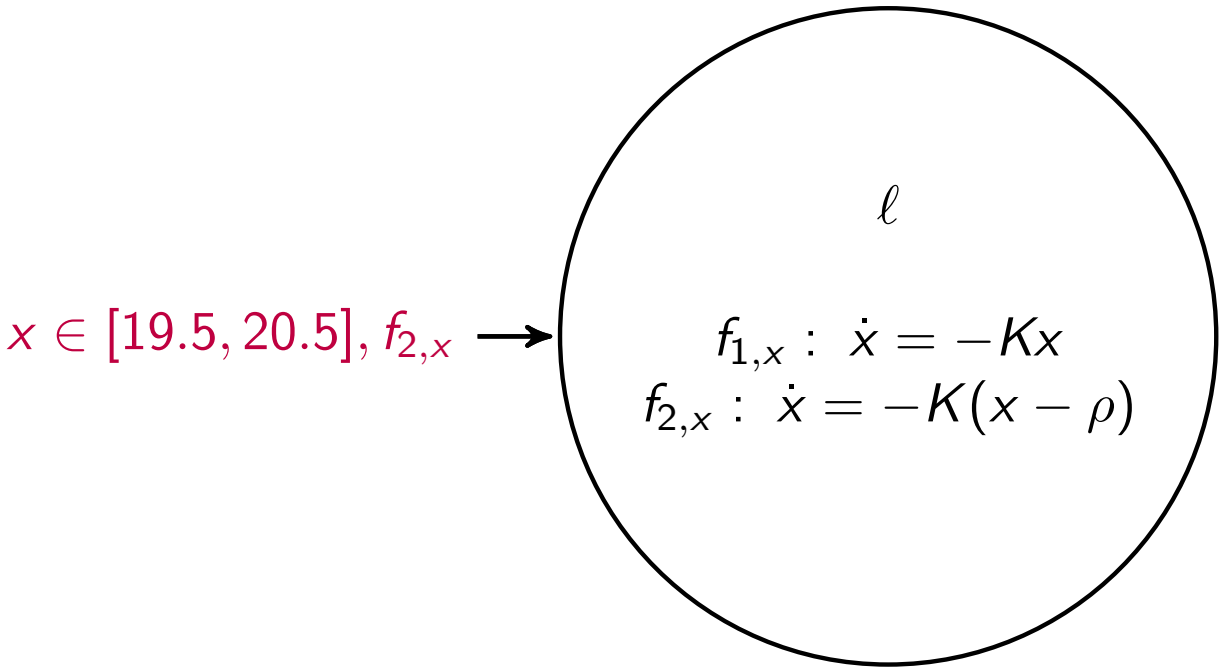
Simplified Hybrid Automaton

$$\mathcal{H}' = (Loc', Var', Lab', Act', Init')$$



Simplified Hybrid Automaton

$$\mathcal{H}' = (Loc', Var', Lab', Act', Init')$$



Modeling the Neural Network Controller

Two broad types of output:

1. Continuous value from an uncountable but usually bounded set

Modeling the Neural Network Controller

Two broad types of output:

1. Continuous value from an uncountable but usually bounded set
   - PID controller

Modeling the Neural Network Controller

Two broad types of output:

1. Continuous value from an uncountable but usually bounded set
   - PID controller
2. Discrete value from a finite and countable set

Modeling the Neural Network Controller

Two broad types of output:

1. Continuous value from an uncountable but usually bounded set
   - PID controller
2. Discrete value from a finite and countable set
   - Multi-class classifier

Modeling the Neural Network Controller

Two broad types of output:

1. Continuous value from an uncountable but usually bounded set
   - PID controller
2. Discrete value from a finite and countable set
   - Multi-class classifier
   - Variable in the hybrid system's flow function

Neural Network Output

- Flow of non-autonomous system:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

Neural Network Output

- Flow of non-autonomous system:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

- Flows of the thermostat:

$$\dot{x} = -Kx \qquad (\ell_{\text{off}})$$
$$\dot{x} = K(\rho - x) = -Kx + K\rho \qquad (\ell_{\text{on}})$$

where $K\rho$ can be seen as the external input $Bu(t)$ from the controller

Neural Network Output

- Flow of non-autonomous system:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

- Flows of the thermostat:

$$\dot{x} = -Kx \qquad (\ell_{\text{off}})$$
$$\dot{x} = K(\rho - x) = -Kx + K\rho \qquad (\ell_{\text{on}})$$

  where $K\rho$ can be seen as the external input $Bu(t)$ from the controller
- Hence, let the neural network predict $K\rho$

Neural Network Output

- Flow of non-autonomous system:

$$\dot{x}(t) = Ax(t) + Bu(t)$$

- Flows of the thermostat:

$$\dot{x} = -Kx \qquad (\ell_{\text{off}})$$
$$\dot{x} = K(\rho - x) = -Kx + K\rho \qquad (\ell_{\text{on}})$$

  where $K\rho$ can be seen as the external input $Bu(t)$ from the controller
- Hence, let the neural network predict $K\rho$
- Or more generally, h_i

Output Classification

- Use step function

$$f_{\text{step}}(x) = \sum_{i=1}^{n} \alpha_i \chi_{A_i}(x)$$

where $\alpha_i \in \mathbb{R}$, $n > 0$, and

$$\chi_{A_i}(x) = \begin{cases} 1, & \text{if } x \in A_i, \\ 0, & \text{otherwise} \end{cases}$$

for each interval A_i, $i = 1, \ldots, n$.

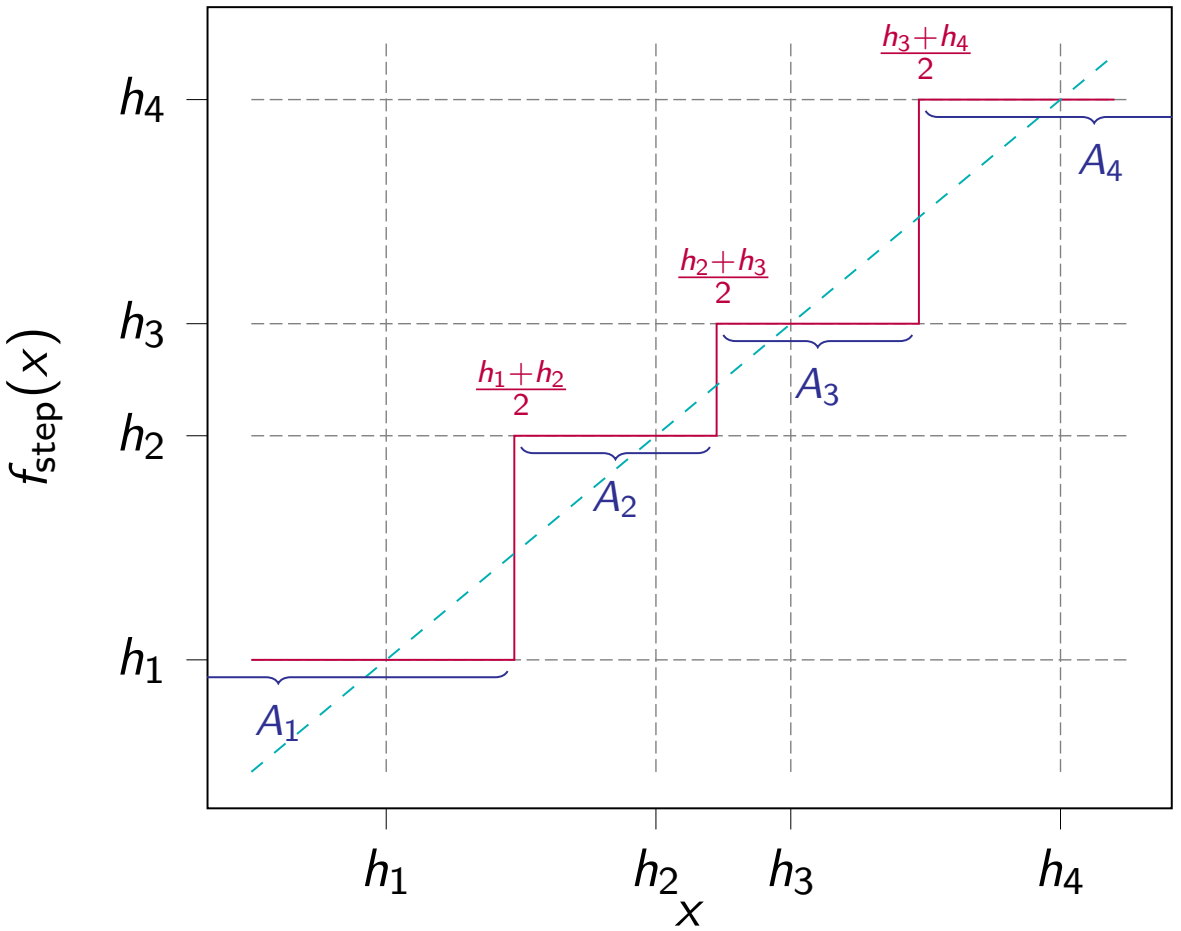
Output Classification (cont'd)

- Given: ordered target outputs $h_i \in \mathbb{R}$ with $i = 1, \ldots, n$ and $h_{i-1} < h_i$
- Apply step function with intervals

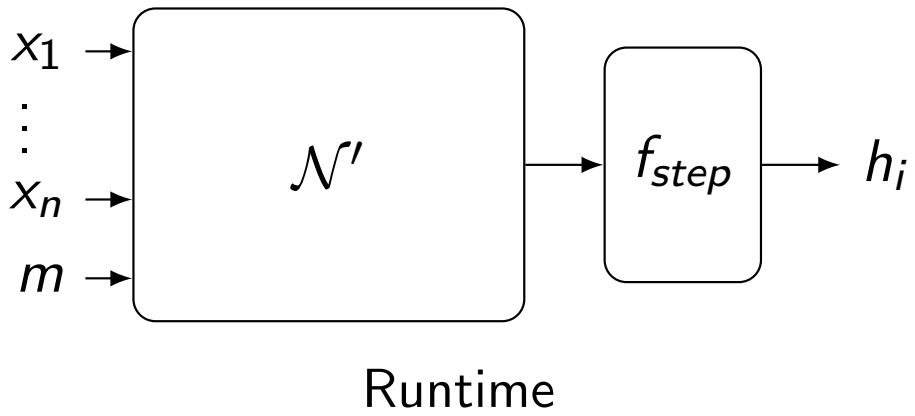
$$A_i = \begin{cases} (-\infty, \frac{h_i + h_{i+1}}{2}), & \text{if } i = 1, \\ [\frac{h_i + h_{i-1}}{2}, \infty), & \text{if } i = n, \\ [\frac{h_{i-1} + h_i}{2}, \frac{h_i + h_{i+1}}{2}), & \text{otherwise} \end{cases}$$

and $\alpha_i = h_i$.

Output Classification: Visualized

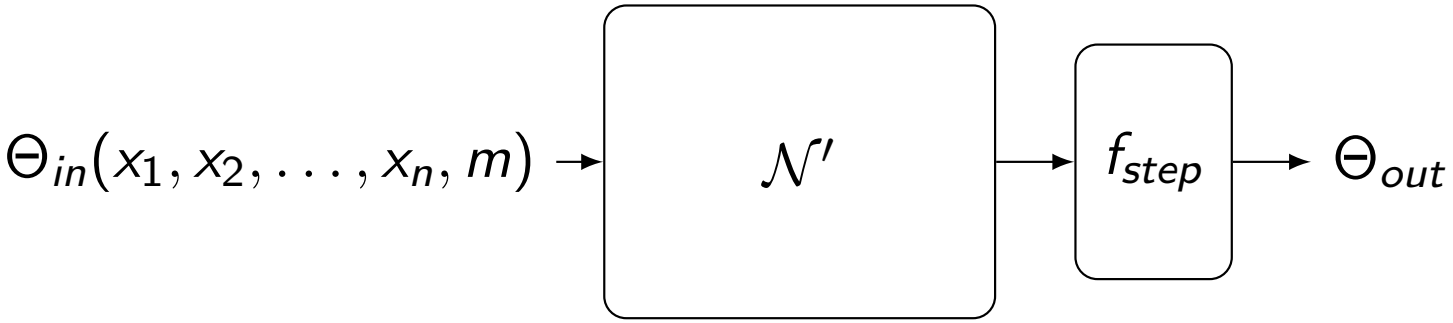


Neural Network Architecture



Runtime

Neural Network Architecture



NN reachability analysis

where $x_j \subset \mathbb{R}$
with Θ_{out} covering trivial sets $\{h_i\}$

Input Star Definition

- Goal: Capture the plant's current state in the input to the controller

Input Star Definition

- Goal: Capture the plant's current state in the input to the controller

Example (Thermostat)

At time point t_k, let

Input Star Definition

- Goal: Capture the plant's current state in the input to the controller

Example (Thermostat)

At time point t_k, let

- temperature $x \in [x_{k_l}, x_{k_u}]$,

Input Star Definition

- Goal: Capture the plant's current state in the input to the controller

Example (Thermostat)

At time point t_k, let

- temperature $x \in [x_{k_l}, x_{k_u}]$,
- mode $m_k \in \{0, 1\}$ (off / on).

Input Star Definition

- Goal: Capture the plant's current state in the input to the controller

Example (Thermostat)

At time point t_k, let

- temperature $x \in [x_{k_l}, x_{k_u}]$,
- mode $m_k \in \{0, 1\}$ (off / on).

Define $\Theta_{I_k} = \langle c, V, P \rangle$ with $P(\alpha) \triangleq C\alpha \leq s$ s.t.

$$c = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad V = \{e_1, e_2\}, \quad C = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}, \quad s = \begin{bmatrix} -x_{k_l} \\ x_{k_u} \\ -m_k \\ m_k \end{bmatrix}.$$

Flowpipe Construction Using Control Input

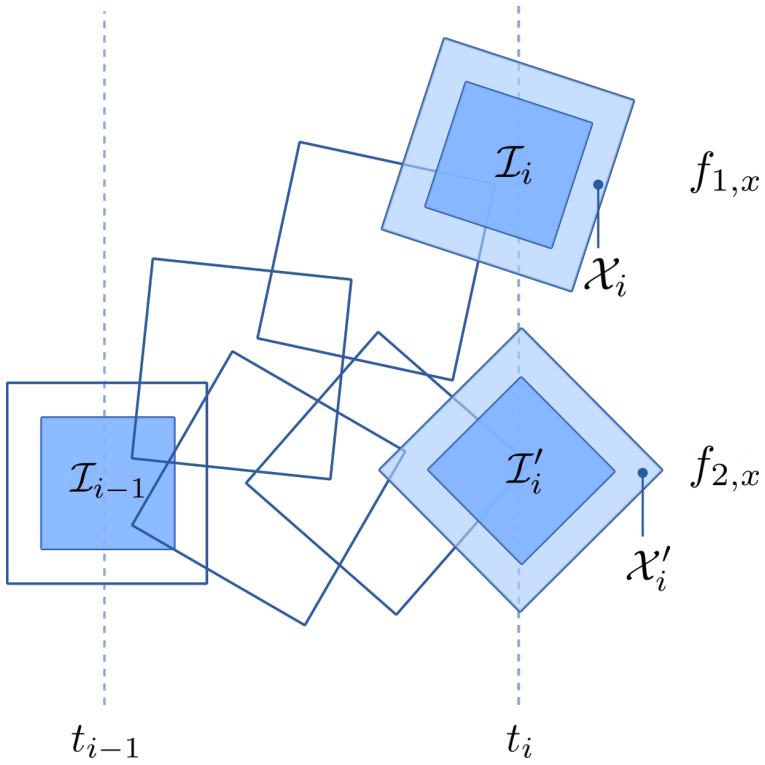
Definition (Initial set of NNCS analysis)

- Let $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$ be an NNCS and $T = K\delta$ the time horizon
- Initial set $\mathcal{I}_k$ at time point $t_k = k\delta$ is defined by

$$\mathcal{I}_0 := \mathcal{X}_0$$
$$\mathcal{I}_k := e^{\delta A_{k-1}} \mathcal{I}_{k-1}, \quad k = 1, \ldots, K$$

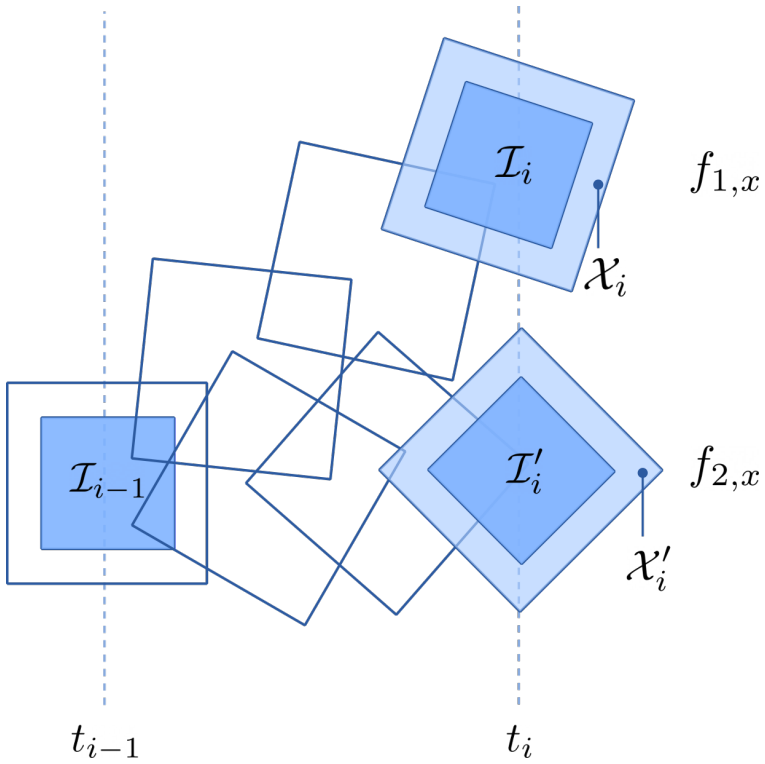
where A_{k-1} is the flow matrix over the time interval $[t_{k-1}, t_k]$.

Handling Potential Branching



$$\mathcal{S}_{i-1} = \{(\mathcal{X}_i, F, \mathcal{I}) \mid \mathcal{X}_i \subseteq V', F \in \mathit{Act}'(\ell), \mathcal{I} \in \mathit{Init}'\}$$

Handling Potential Branching



$$\mathcal{S}_{i-1} = \{(\mathcal{X}_i, f_{1,x}, \mathcal{I}_{i-1}), (\mathcal{X}'_i, f_{2,x}, \mathcal{I}_{i-1})\}$$

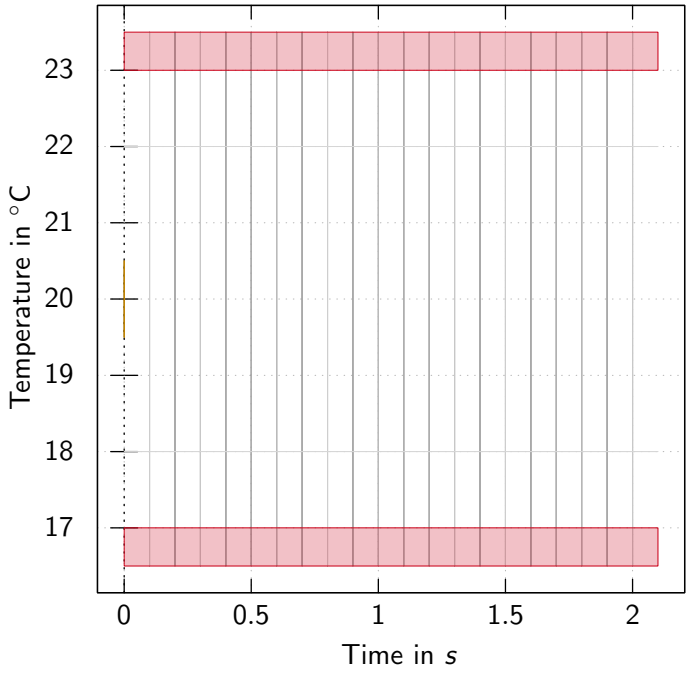
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\quad\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\quad\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: $\quad$**for** $i = 1$ to K **do**
6: $\quad\quad\mathcal{S}_i \leftarrow \emptyset$;
7: $\quad\quad$**for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $\quad\quad\quad(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\quad\quad\quad\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: $\quad\quad$**end for**
11: $\quad\quad$**for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\quad\quad\quad\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: $\quad\quad$**end for**
15: $\quad\quad\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\quad\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: $\quad$**end for**
18: $\quad$**return** $\mathcal{R}$
19: **end procedure**

$t_0 = 0$



$$\mathcal{I}_0 = \{(f_{2,x}, \nu) \in \Sigma' \mid \nu(x) \in [19.5, 20.5]\}$$

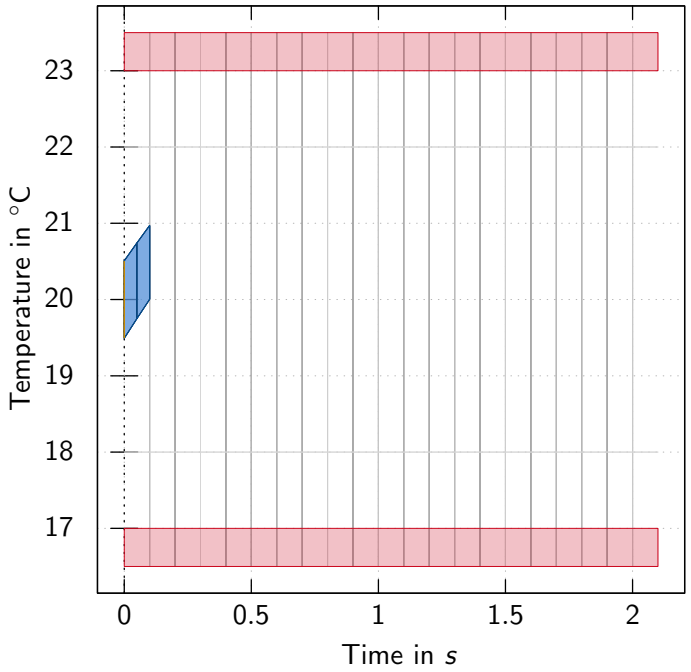
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: **for** $i = 1$ to K **do**
6: $\mathcal{S}_i \leftarrow \emptyset$;
7: **for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: **end for**
11: **for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: **end for**
15: $\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: **end for**
18: **return** $\mathcal{R}$
19: **end procedure**

$t_0 = 0$



Flowpipe construction for δ time

$\mathcal{R} = \Omega_{[0,\delta]}(\mathcal{X}_0, \mathcal{U})$

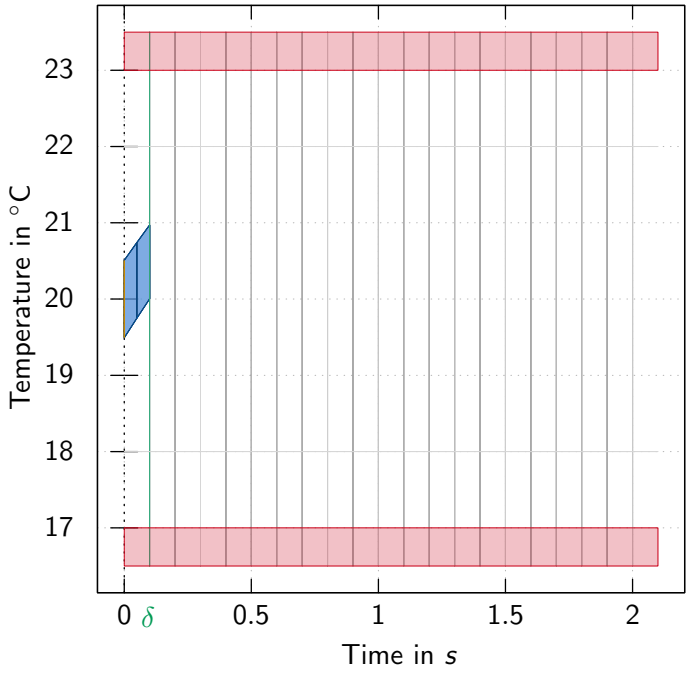
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\quad\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\quad\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: $\quad$**for** $i = 1$ to K **do**
6: $\quad\quad\mathcal{S}_i \leftarrow \emptyset$;
7: $\quad\quad$**for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $\quad\quad\quad(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\quad\quad\quad\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: $\quad\quad$**end for**
11: $\quad\quad$**for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\quad\quad\quad\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: $\quad\quad$**end for**
15: $\quad\quad\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\quad\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: $\quad$**end for**
18: $\quad$**return** $\mathcal{R}$
19: **end procedure**

$t_0 = 0$



$$\mathcal{X}_i = \mathcal{R} \cap \{\nu \mid \nu(t) = \delta\}$$
$$= \{\nu \mid \nu(t) \in \delta, \nu(x) \in [20, 21]\}$$

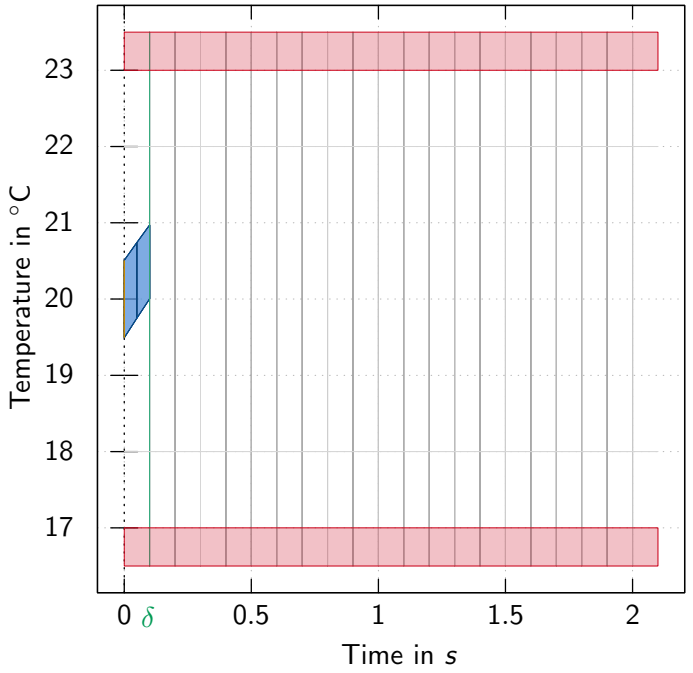
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\quad\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\quad\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$;
5: $\quad$**for** $i = 1$ to K **do**
6: $\quad\quad\mathcal{S}_i \leftarrow \emptyset$;
7: $\quad\quad$**for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $\quad\quad\quad(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\quad\quad\quad\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: $\quad\quad$**end for**
11: $\quad\quad$**for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\quad\quad\quad\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: $\quad\quad$**end for**
15: $\quad\quad\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\quad\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: $\quad$**end for**
18: $\quad$**return** $\mathcal{R}$
19: **end procedure**

$t_1 = \delta$



$$\mathcal{X}_i = \mathcal{R} \cap \{\nu \mid \nu(t) = \delta\}$$
$$= \{\nu \mid \nu(t) \in \delta, \nu(x) \in [20, 21]\}$$
$$\mathcal{S}_{i-1} = \{(\mathcal{X}_i, f_{2,x}, \mathcal{I}_0)\}$$

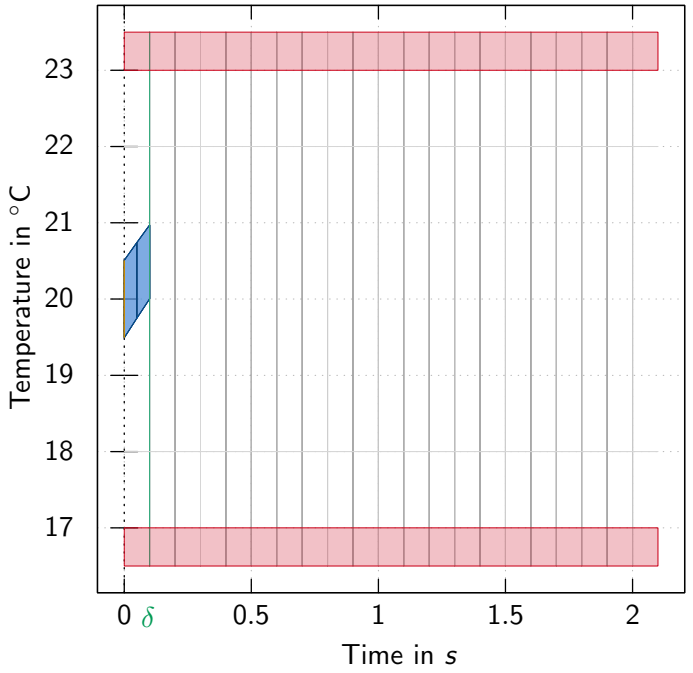
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: **for** $i = 1$ to K **do**
6: $\mathcal{S}_i \leftarrow \emptyset$;
7: **for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: **end for**
11: **for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: **end for**
15: $\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: **end for**
18: **return** $\mathcal{R}$
19: **end procedure**

$t_1 = \delta$



$$\mathcal{X}_i = \mathcal{R} \cap \{\nu \mid \nu(t) = \delta\}$$
$$= \{\nu \mid \nu(t) \in \delta, \nu(x) \in [20, 21]\}$$
$$\mathcal{S}_{i-1} = \{(\mathcal{X}_i, f_{2,x}, \mathcal{I}_0)\}$$

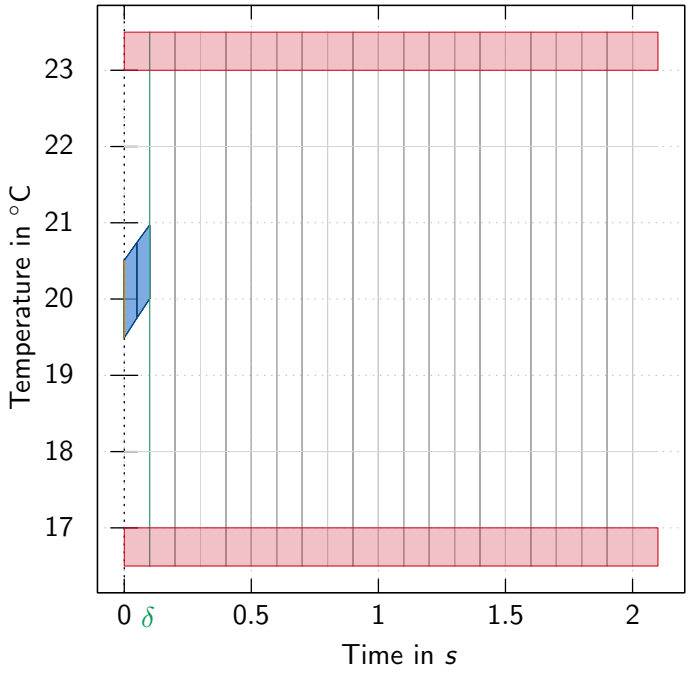
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\quad\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\quad\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: $\quad$**for** $i = 1$ to K **do**
6: $\quad\quad\mathcal{S}_i \leftarrow \emptyset$;
7: $\quad\quad$**for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $\quad\quad\quad(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\quad\quad\quad\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: $\quad\quad$**end for**
11: $\quad\quad$**for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\quad\quad\quad\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: $\quad\quad$**end for**
15: $\quad\quad\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\quad\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: $\quad$**end for**
18: $\quad$**return** $\mathcal{R}$
19: **end procedure**

$t_1 = \delta$



$$\mathcal{N}(\Theta(\mathcal{X}_j)) = \Theta_y \text{ where } \Theta_y.range = \{K\rho\}$$
$$\implies F' = f_{2,x}$$
$$\implies \mathcal{I}' = \text{AFFINETRANSFORMATION}(\mathcal{I}_0, f_{2,x}, \delta)$$
$$\implies \mathcal{S}_i = \{(\mathcal{X}_i, f_{2,x}, \mathcal{I}')\}$$

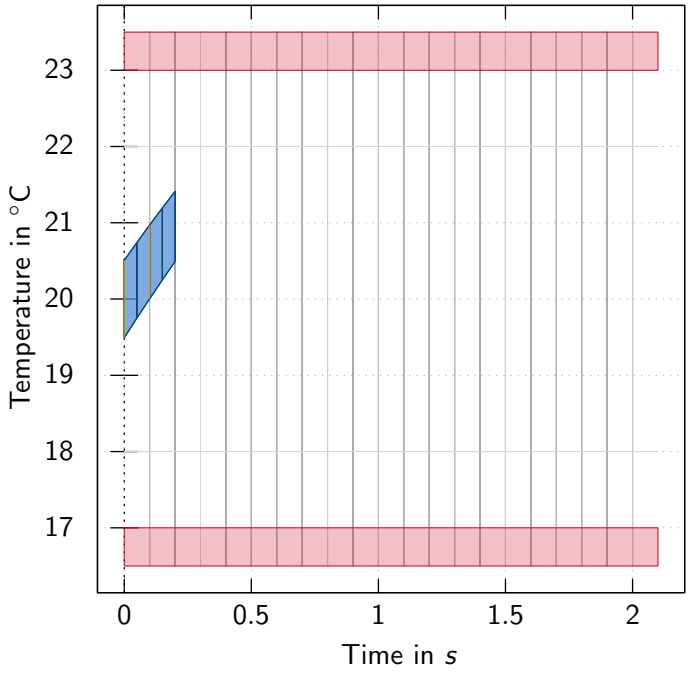
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** nncsReach($\mathcal{I}_0, K$)
2: $\mathcal{R} \leftarrow$ constructFlowpipe($\mathcal{I}_0, \delta$);
3: $\mathcal{X}_i \leftarrow$ getRelevantSets($\mathcal{R}, \delta$);
4: $\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: **for** $i = 1$ to K **do**
6: $\mathcal{S}_i \leftarrow \emptyset$;
7: **for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $(\mathcal{X}', F', \mathcal{I}') \leftarrow$ computeSetFlowMaps($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: **end for**
11: **for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\mathcal{R}_j \leftarrow$ constructFlowpipe($\mathcal{I}_j, \delta$);
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: **end for**
15: $\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\mathcal{X}_i \leftarrow$ getRelevantSets($\mathcal{R}, (i+1)\delta$);
17: **end for**
18: **return** $\mathcal{R}$
19: **end procedure**

$t_1 = \delta$



$\mathcal{R}_j = \Omega_{[\delta,2\delta]}(\mathcal{X}_j, \mathcal{U})$

$\mathcal{R} = \Omega_{[0,2\delta]}(\mathcal{X}_0, \mathcal{U})$

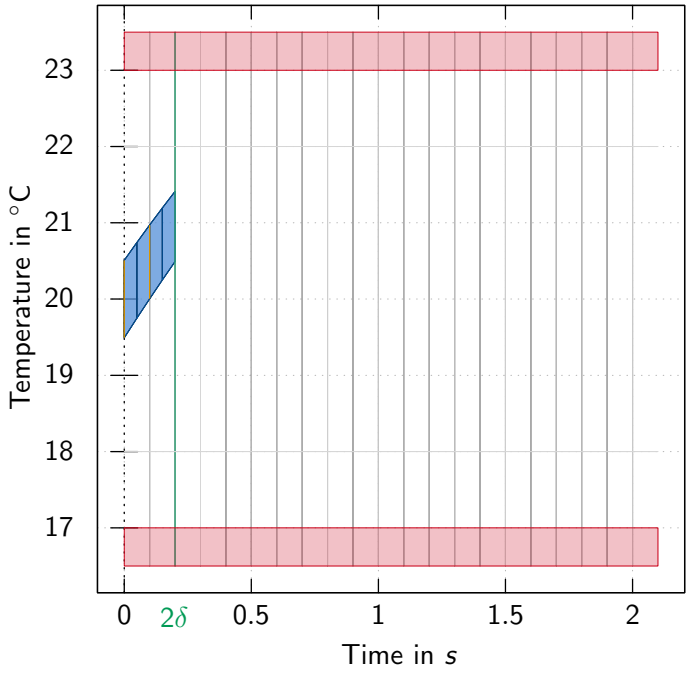
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\quad\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\quad\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: $\quad$**for** $i = 1$ to K **do**
6: $\quad\quad\mathcal{S}_i \leftarrow \emptyset$;
7: $\quad\quad$**for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $\quad\quad\quad(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\quad\quad\quad\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: $\quad\quad$**end for**
11: $\quad\quad$**for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\quad\quad\quad\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: $\quad\quad$**end for**
15: $\quad\quad\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\quad\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: $\quad$**end for**
18: $\quad$**return** $\mathcal{R}$
19: **end procedure**

$t_1 = \delta$



$$\mathcal{X}_i = \mathcal{R} \cap \{\nu \mid \nu(t) = 2\delta\}$$
$$= \{\nu \mid \nu(t) \in \delta, \nu(x) \in [20.5, 21.5]\}$$

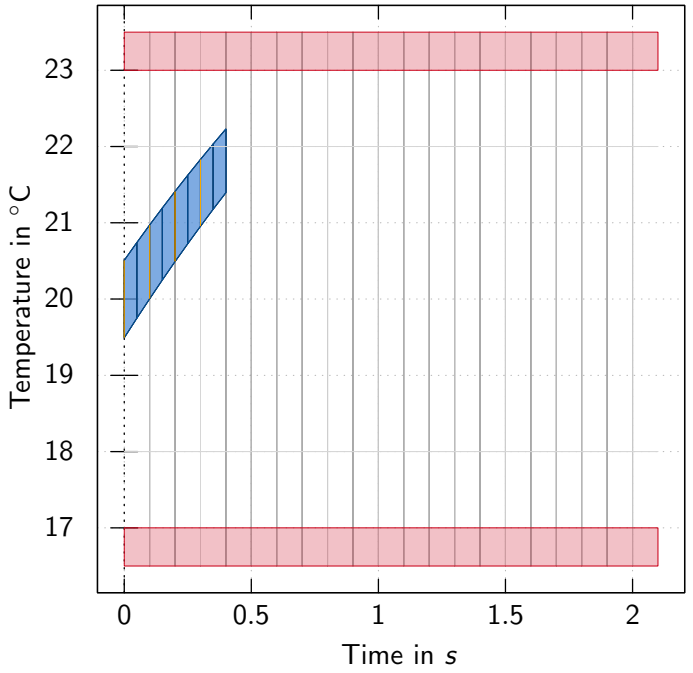
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: **for** $i = 1$ to K **do**
6: $\mathcal{S}_i \leftarrow \emptyset$;
7: **for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: **end for**
11: **for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: **end for**
15: $\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: **end for**
18: **return** $\mathcal{R}$
19: **end procedure**

$t_4 = 4\delta$



$$\mathcal{N}(\Theta(\mathcal{X}_j)).range = \{0, K\rho\}$$
$$\implies \mathcal{S}_i = \{(\mathcal{X}_i, f_{1,x}, \mathcal{I}'), (\mathcal{X}_i, f_{2,x}, \mathcal{I}'')\}$$

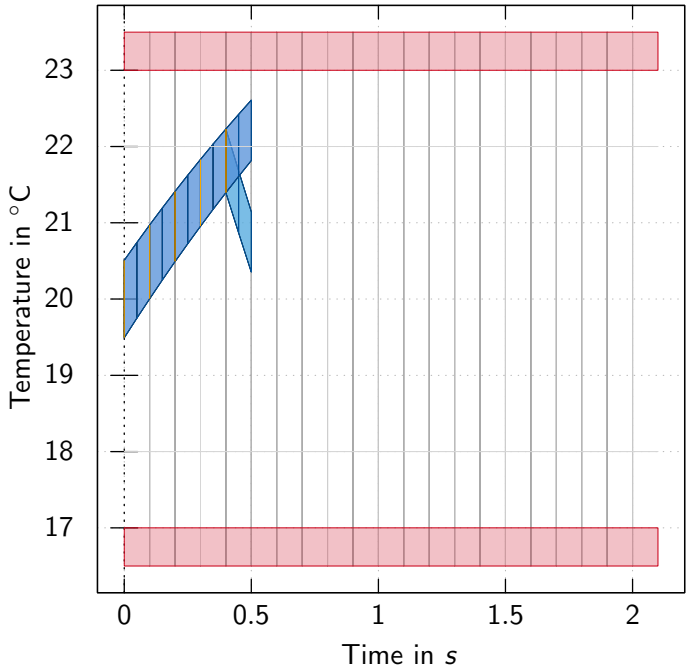
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: **for** $i = 1$ to K **do**
6: $\mathcal{S}_i \leftarrow \emptyset$;
7: **for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: **end for**
11: **for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: **end for**
15: $\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: **end for**
18: **return** $\mathcal{R}$
19: **end procedure**

$t_4 = 4\delta$



Branching into two flowpipes with different flows

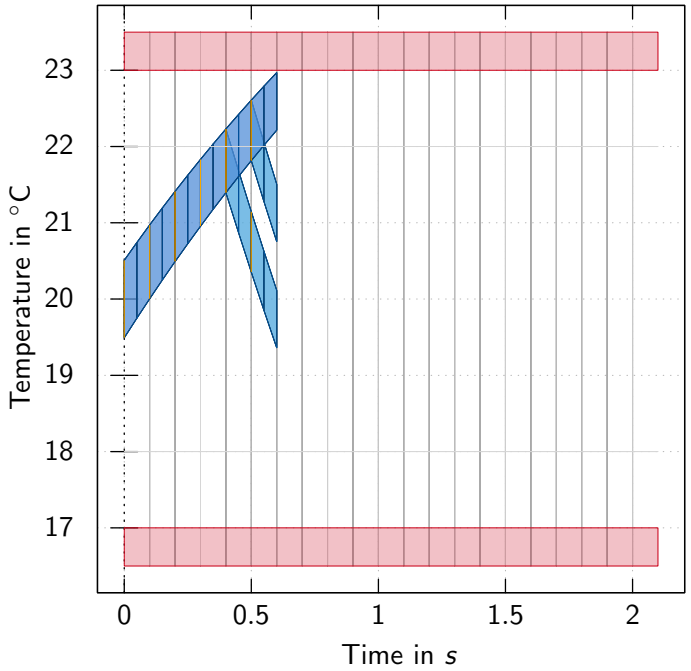
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: **for** $i = 1$ to K **do**
6: $\mathcal{S}_i \leftarrow \emptyset$;
7: **for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: **end for**
11: **for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: **end for**
15: $\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: **end for**
18: **return** $\mathcal{R}$
19: **end procedure**

$t_5 = 5\delta$



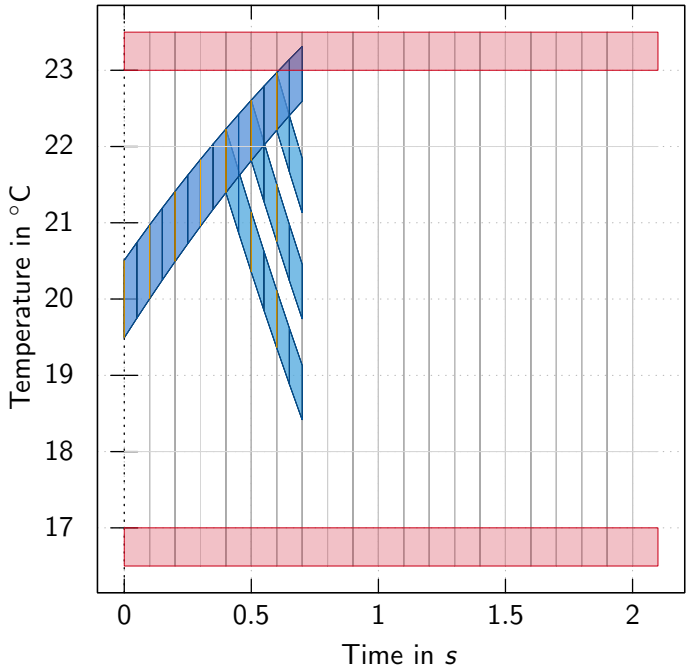
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: **for** $i = 1$ to K **do**
6: $\mathcal{S}_i \leftarrow \emptyset$;
7: **for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: **end for**
11: **for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: **end for**
15: $\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: **end for**
18: **return** $\mathcal{R}$
19: **end procedure**

$t_6 = 6\delta$



Unsafe region entered

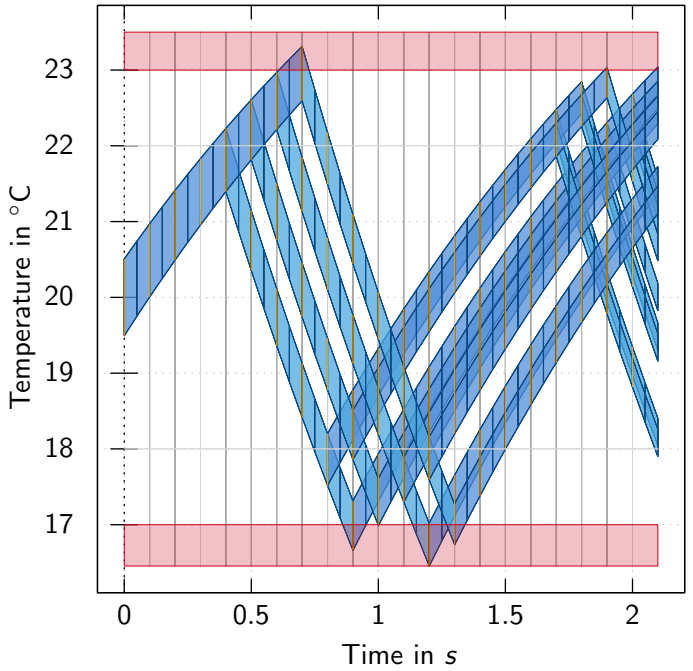
NNCS Analysis: Pseudo-code

Algorithm NNCS analysis (discrete).

Input: NNCS $(X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$, HA $\mathcal{H}'$, initial set $\mathcal{I}_0$, $T = K\delta$
Output: Reachable states $\mathcal{R}$ over $[0, T]$

1: **procedure** NNCSREACH($\mathcal{I}_0, K$)
2: $\quad\mathcal{R} \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_0, \delta$);
3: $\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, \delta$);
4: $\quad\mathcal{S}_{i-1} \leftarrow \{(\mathcal{X}_i, F_0, \mathcal{I}_0)\}$
5: $\quad$**for** $i = 1$ to K **do**
6: $\quad\quad\mathcal{S}_i \leftarrow \emptyset$;
7: $\quad\quad$**for** $(\mathcal{X}_j, F_j, \mathcal{I}_j) \in \mathcal{S}_{i-1}$ **do**
8: $\quad\quad\quad(\mathcal{X}', F', \mathcal{I}') \leftarrow$ COMPUTESETFLOWMAPS($\mathcal{X}_j, F_j, \mathcal{I}_j$);
9: $\quad\quad\quad\mathcal{S}_i \leftarrow \mathcal{S}_i \cup \{(\mathcal{X}', F', \mathcal{I}')\}$;
10: $\quad\quad$**end for**
11: $\quad\quad$**for** $j = 1$ to $|\mathcal{S}_i|$ **do**
12: $\quad\quad\quad\mathcal{R}_j \leftarrow$ CONSTRUCTFLOWPIPE($\mathcal{I}_j, \delta$);
13: $\quad\quad\quad\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_j$;
14: $\quad\quad$**end for**
15: $\quad\quad\mathcal{S}_{i-1} \leftarrow \mathcal{S}_i$;
16: $\quad\quad\mathcal{X}_i \leftarrow$ GETRELEVANTSETS($\mathcal{R}, (i+1)\delta$);
17: $\quad$**end for**
18: $\quad$**return** $\mathcal{R}$
19: **end procedure**

$t_{21} = 21\delta$



Full time horizon reached

Soundness of NNCS Analysis

Theorem (Over-approximation of First Segment)

Let $\mathfrak{N} = (X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$ *be an NNCS. Then*

$$\mathcal{R}_{[0,\delta]}(\mathcal{X}_0) \subseteq \Omega_{[0,\delta]}(\mathcal{X}_0, \mathcal{U}).$$

Soundness of NNCS Analysis

Theorem (Over-approximation of First Segment)

Let $\mathfrak{N} = (X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$ be an NNCS. Then

$$\mathcal{R}_{[0,\delta]}(\mathcal{X}_0) \subseteq \Omega_{[0,\delta]}(\mathcal{X}_0, \mathcal{U}).$$

$$\implies \mathcal{R}_{[0,T]}(\mathcal{X}_0) \subseteq \Omega_{[0,T]}(\mathcal{X}_0, \mathcal{U}) = \bigcup_{i=1}^{K} \Omega_{[(i-1)\delta, i\delta]}(\mathcal{I}_{i-1}, \mathcal{U}).$$

Soundness of NNCS Analysis

Theorem (Over-approximation of First Segment)

Let $\mathfrak{N} = (X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$ *be an NNCS. Then*

$$\mathcal{R}_{[0,\delta]}(\mathcal{X}_0) \subseteq \Omega_{[0,\delta]}(\mathcal{X}_0, \mathcal{U}).$$

$$\implies \mathcal{R}_{[0,T]}(\mathcal{X}_0) \subseteq \Omega_{[0,T]}(\mathcal{X}_0, \mathcal{U}) = \bigcup_{i=1}^{K} \Omega_{[(i-1)\delta, i\delta]}(\mathcal{I}_{i-1}, \mathcal{U}).$$

Theorem

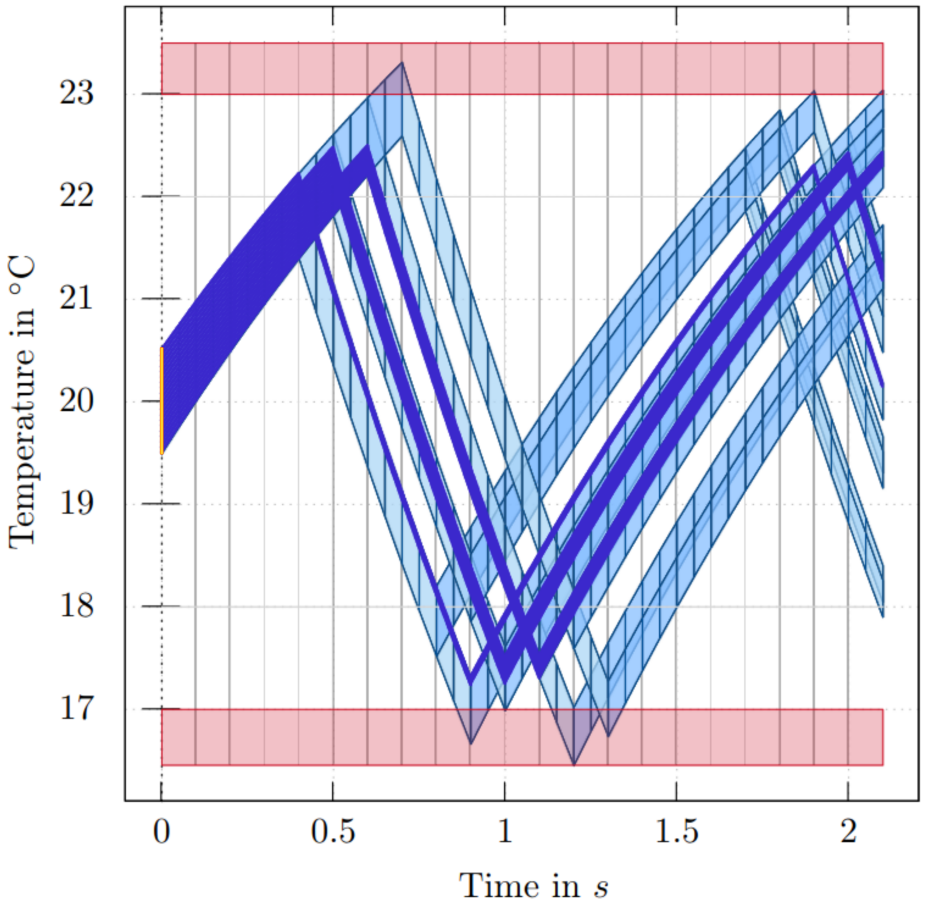
Soundness Let $\mathfrak{N} = (X, U, F, \mathcal{N}, \delta, \mathcal{X}_0)$ *be an NNCS,* $I \subseteq X$ *the initial state set,* $\bar{S} \subseteq X$ *the set of bad states. Then* $\mathfrak{N}$ *is safe iff* $\Omega_{[0,T]}(I) \cap \bar{S} = \emptyset$.

Contents

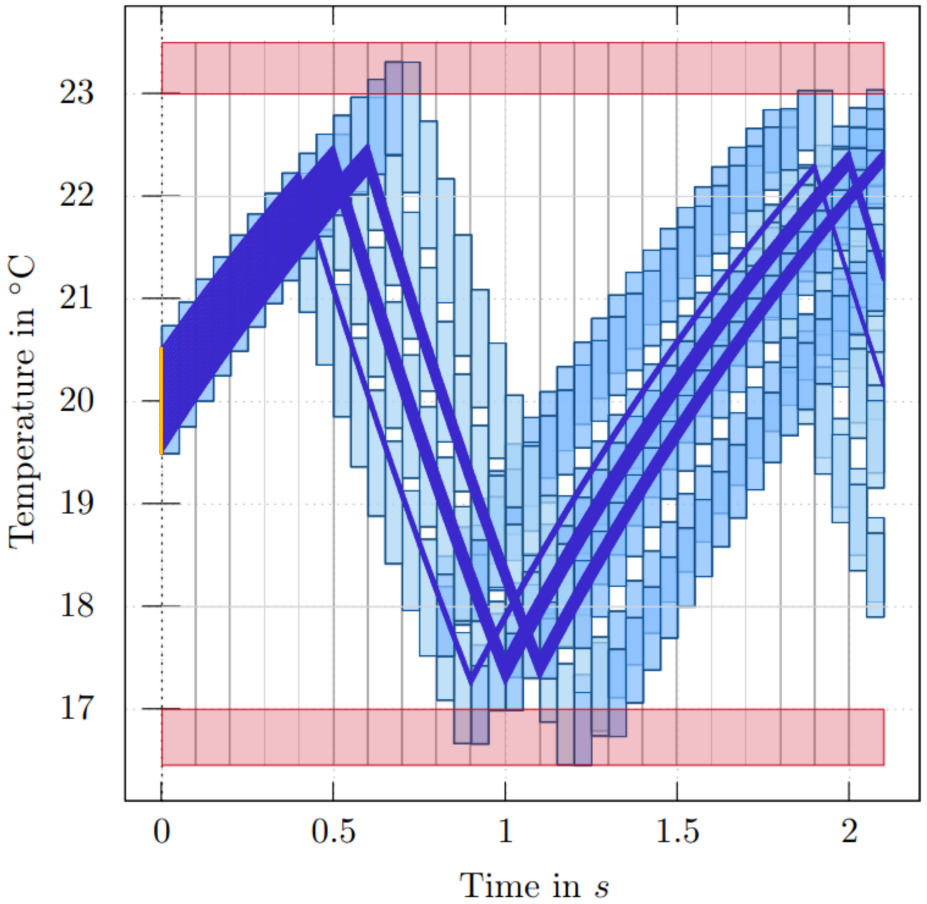
Benchmarks

- Thermostat
  - Global time horizon $T = 2.1$
  - Step size $\delta = 0.1$
  - Segment size 0.05
  - Initial set $x \in [19.5, 20.5]$
- Rod reactor
  - Global time horizon $T = 17$
  - Step size $\delta = 1$
  - Segment size 0.2
  - Initial set $x \in [510, 520], c_1 = c_2 = 20$

Result on Thermostat

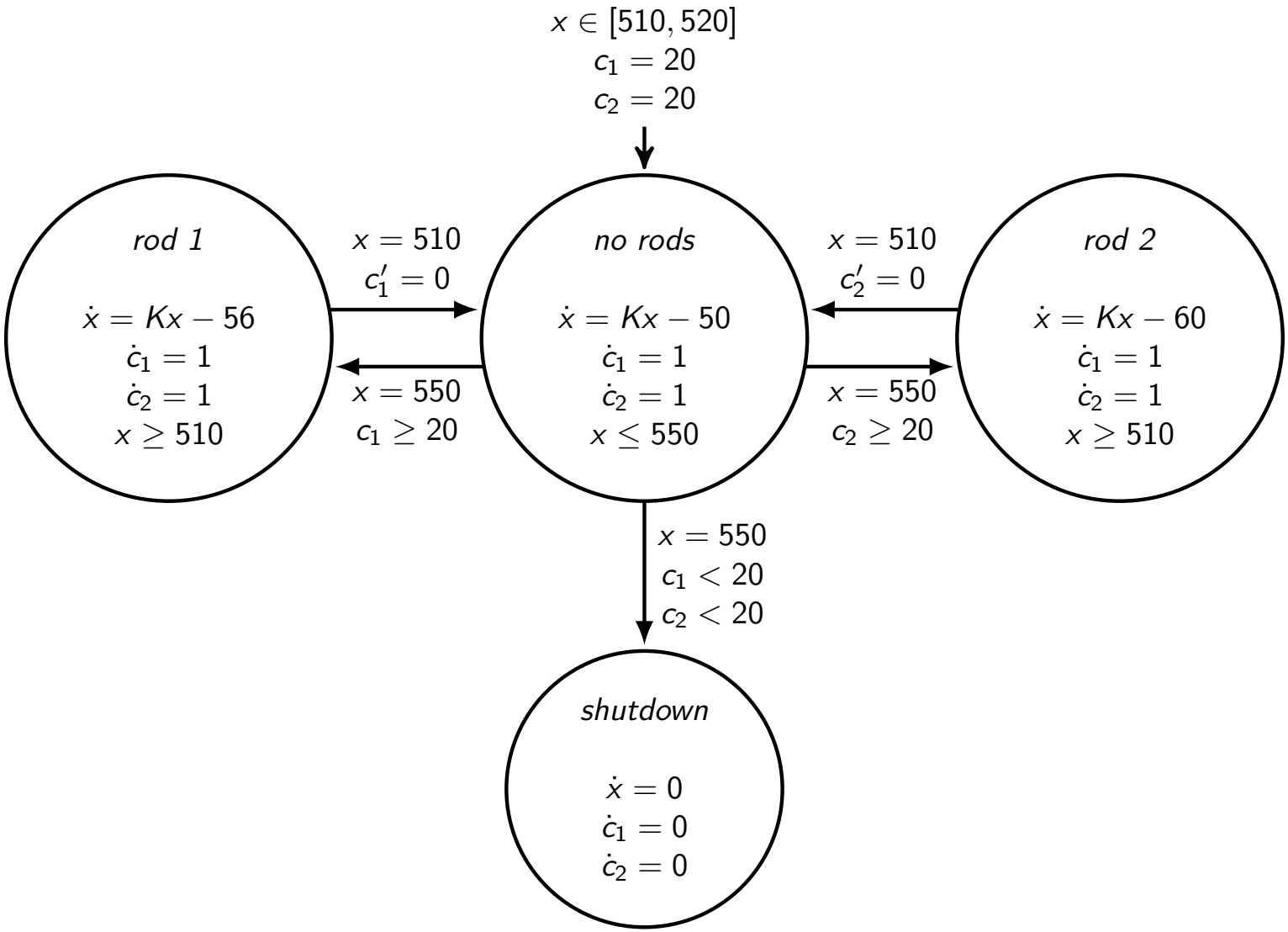


(a) $\mathcal{V}$-polytope representation

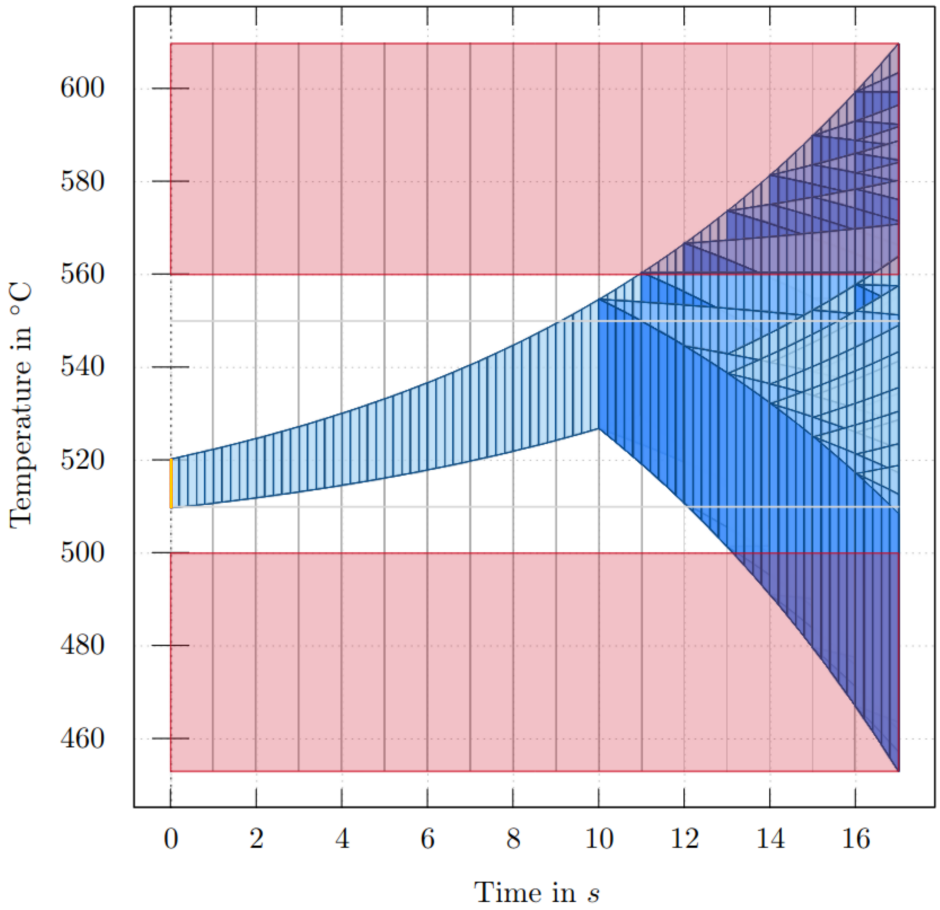


(b) Box representation

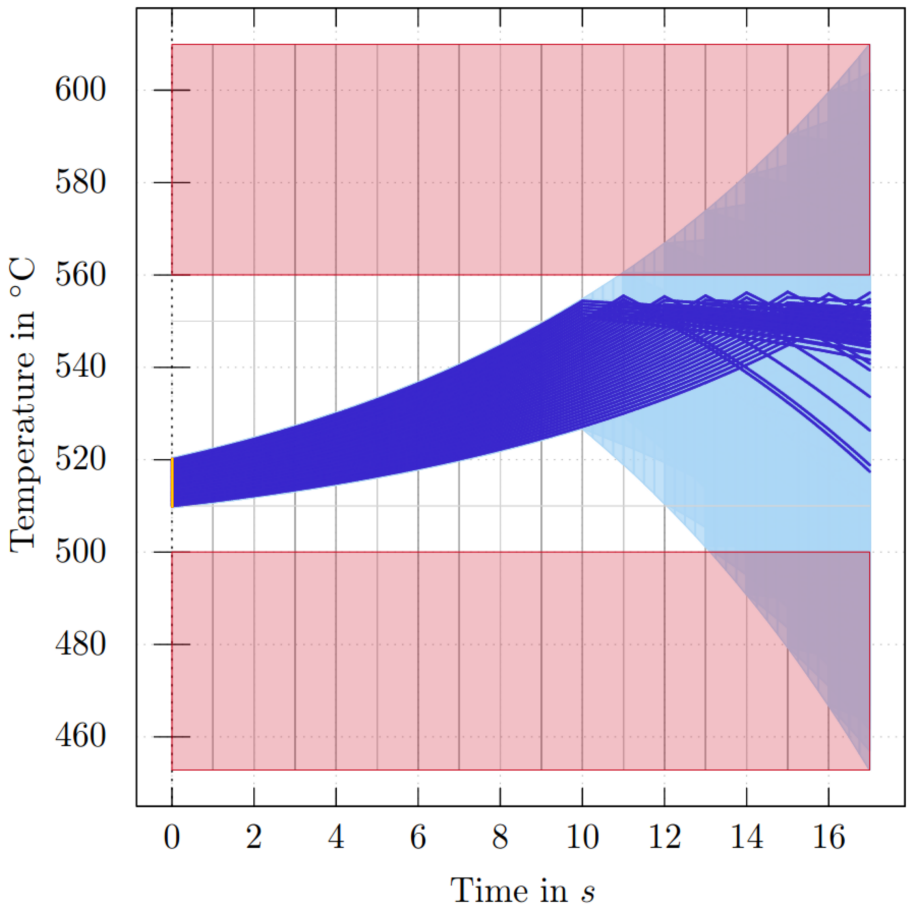
Rod Reactor Hybrid Automaton



Result on Rod Reactor



(a) Reachable states colored by flow



(b) With trajectories (dark blue) overlay

Quantifying Metrics

- Run-time
  - Run 100 times

Quantifying Metrics

- Run-time
  - Run 100 times
- Accuracy
  - Area of projected flowpipes

Quantifying Metrics

- Run-time
  - Run 100 times
- Accuracy
  - Area of projected flowpipes
- Complexity
  - Branching and number of flowpipes

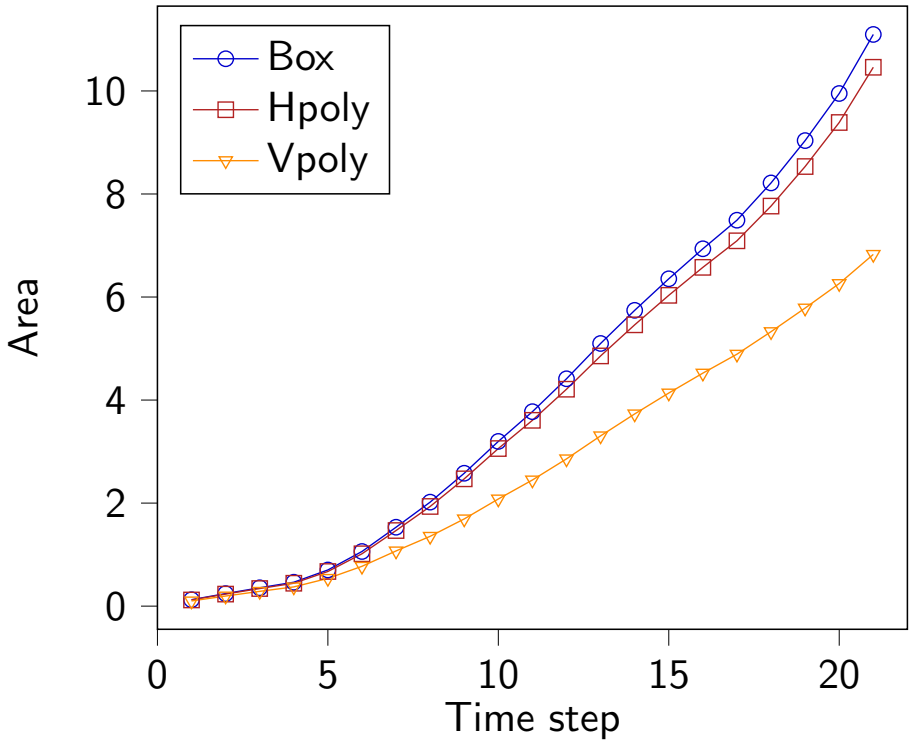
Run-time in ms

Rep	Mean	Median	Min	Max	Std
Thermostat					
Box	880.73	864.33	657.49	3433.20	279.67
$\mathcal{H}$-polytope	4426.76	4418.52	3759.03	4980.86	186.12
$\mathcal{V}$-polytope	2544.66	2496.54	1650.08	5090.44	355.09
Rod reactor					
Box	6969.49	6953.31	4877.65	9638.46	474.57
$\mathcal{V}$-polytope	93445.35	91314.90	87657.10	116347.00	4646.71

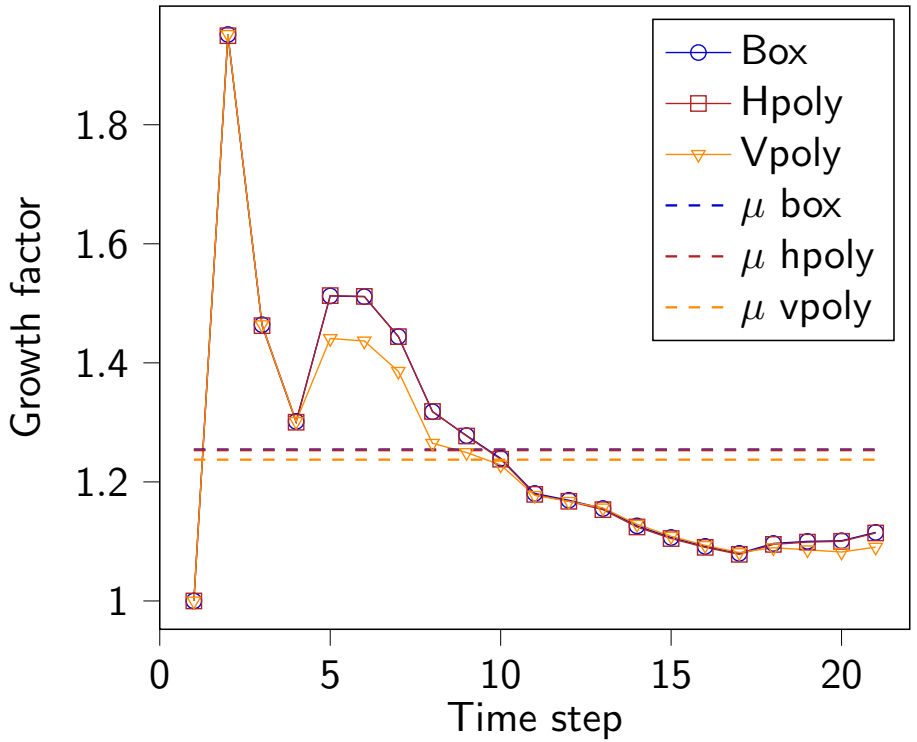
Total area over projections

Projection	Box	$\mathcal{H}$-polytope	$\mathcal{V}$-polytope
Thermostat			
t, x	11.0967	10.4597	6.8272
Rod Reactor			
t, x	8226.88	-	8087.65
t, c_1	49.02	-	5.82292

Accuracy Thermostat

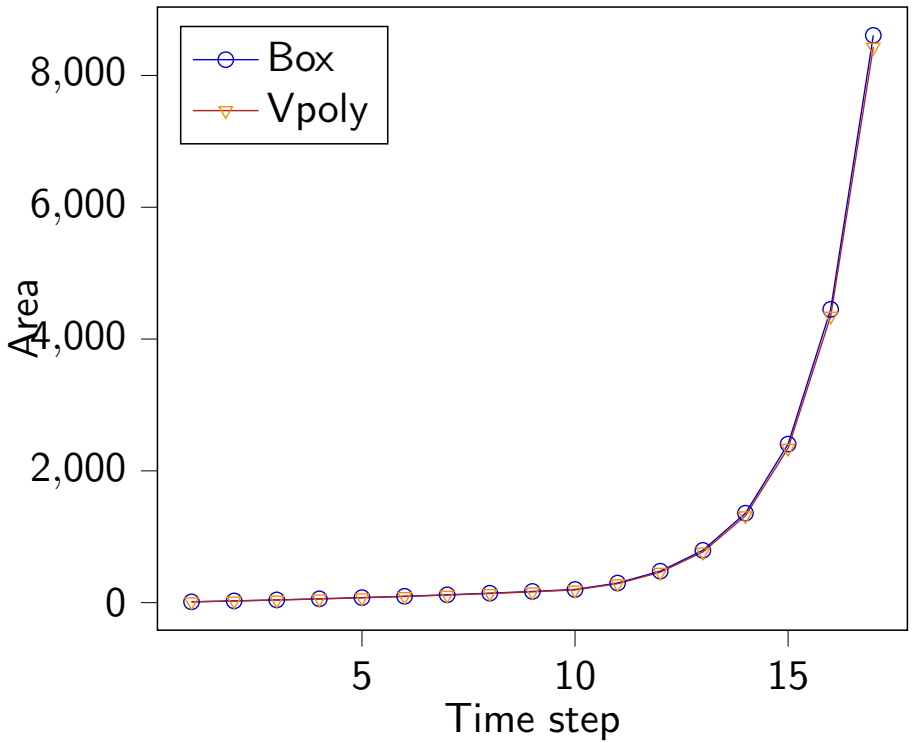


(a) Cumulative area

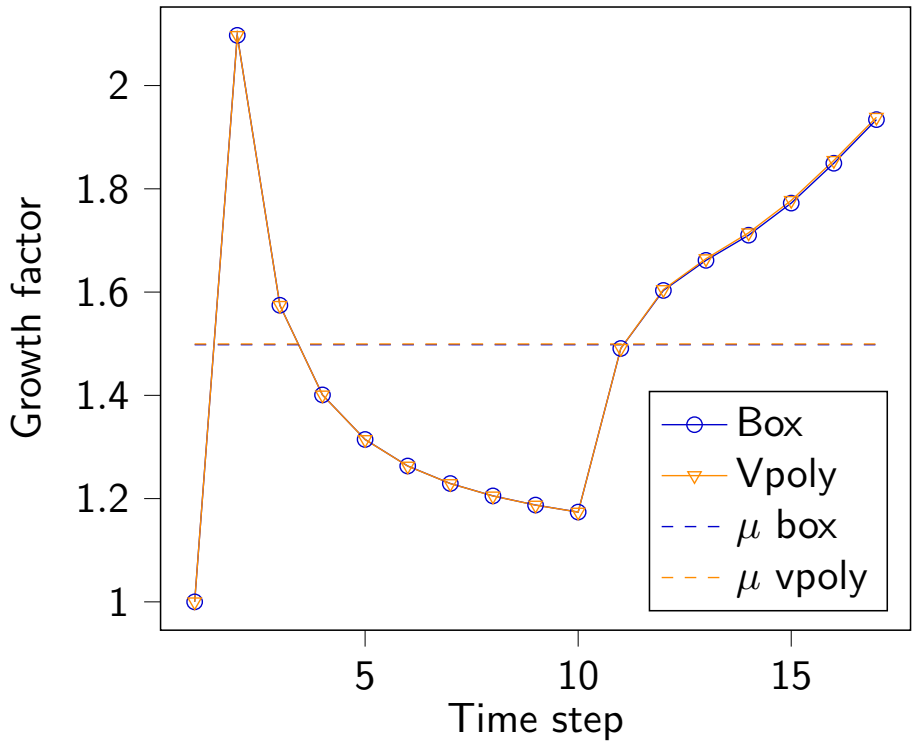


(b) Area growth factor

Accuracy Rod Reactor

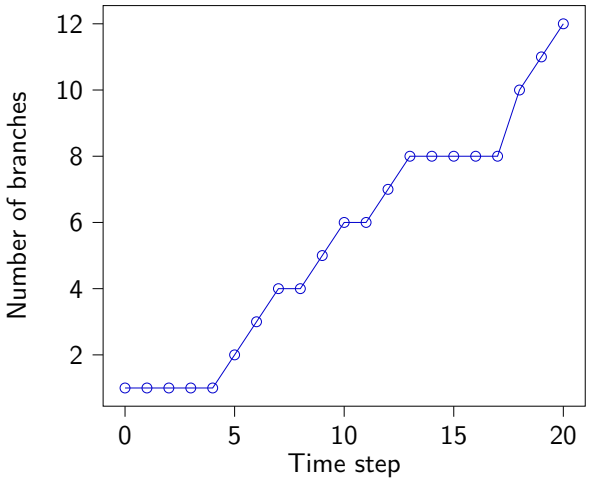


(a) Cumulative area

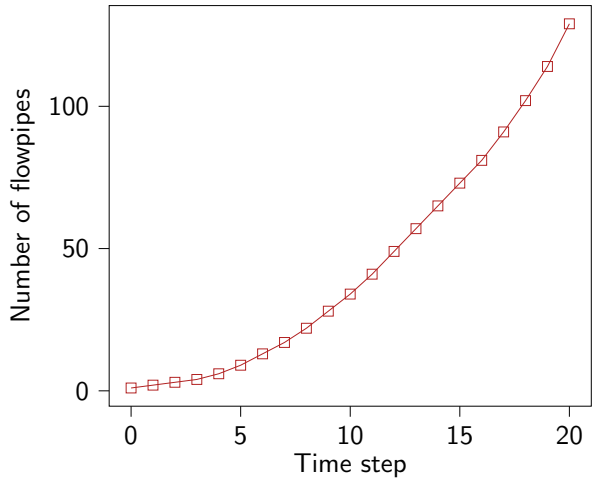


(b) Area growth factor

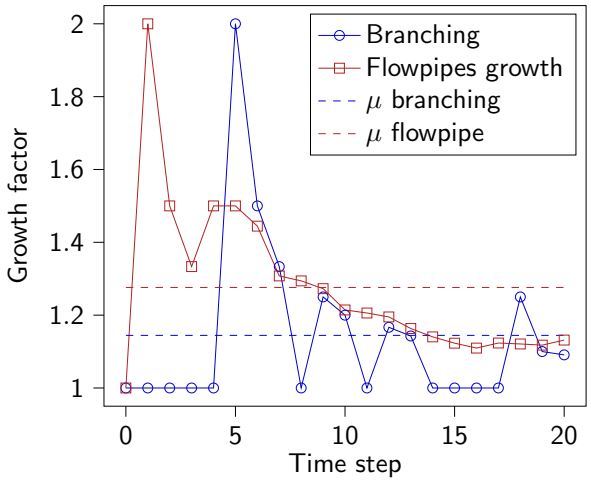
Complexity Thermostat



(a) Branching

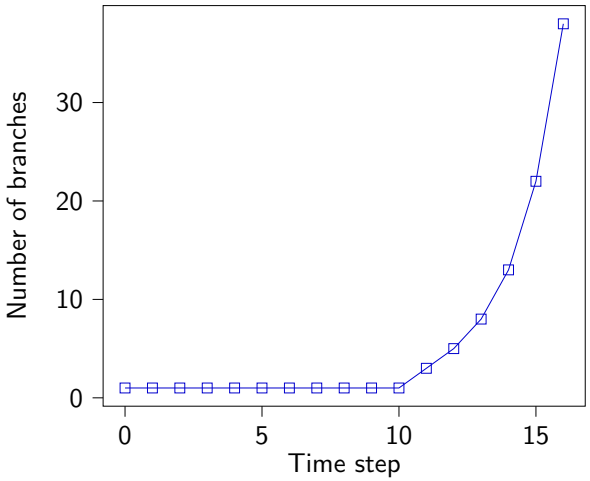


(b) Flowpipes

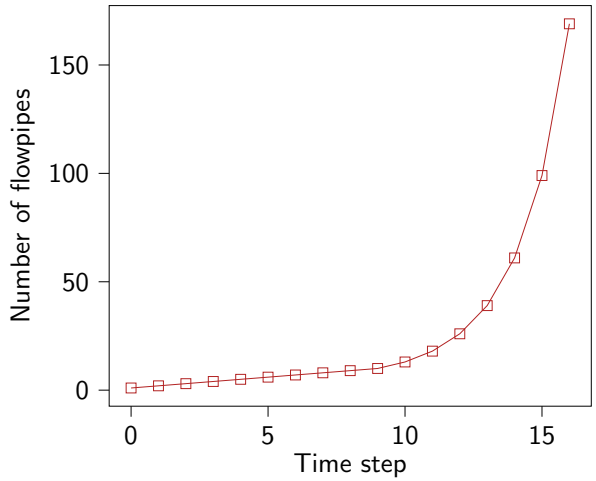


(c) Growth factors

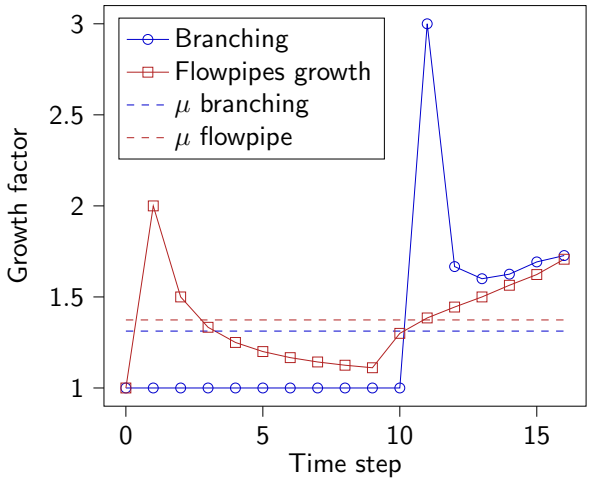
Complexity Rod Reactor



(a) Branching



(b) Flowpipes



(c) Growth factors

Contents

Discussion

- Algorithm for NNCS safety verification of hybrid systems

Discussion

- Algorithm for NNCS safety verification of hybrid systems
  - Flowpipe construction
  - Exact star-based neural network reachability analysis

Discussion

- Algorithm for NNCS safety verification of hybrid systems
  - Flowpipe construction
  - Exact star-based neural network reachability analysis
- Mitigates accumulated wrapping effects

Discussion

- Algorithm for NNCS safety verification of hybrid systems
  - Flowpipe construction
  - Exact star-based neural network reachability analysis
- Mitigates accumulated wrapping effects
- Assessed effectiveness on two benchmarks

Discussion

- Algorithm for NNCS safety verification of hybrid systems
  - Flowpipe construction
  - Exact star-based neural network reachability analysis
- Mitigates accumulated wrapping effects
- Assessed effectiveness on two benchmarks
- Quantified run-time, accuracy, and complexity

Discussion

- Algorithm for NNCS safety verification of hybrid systems
  - Flowpipe construction
  - Exact star-based neural network reachability analysis
- Mitigates accumulated wrapping effects
- Assessed effectiveness on two benchmarks
- Quantified run-time, accuracy, and complexity
- Implemented with HyPro [Schupp et al. 2017]

Future Work

- Refine controller output to reduce over-approximation errors

Future Work

- Refine controller output to reduce over-approximation errors
- Handle nonlinear dynamics for comparability with existing techniques

Future Work

- Refine controller output to reduce over-approximation errors
- Handle nonlinear dynamics for comparability with existing techniques
- Extend to PID controller output

Future Work

- Refine controller output to reduce over-approximation errors
- Handle nonlinear dynamics for comparability with existing techniques
- Extend to PID controller output
- Use reinforcement learning to find optimal controls

Future Work

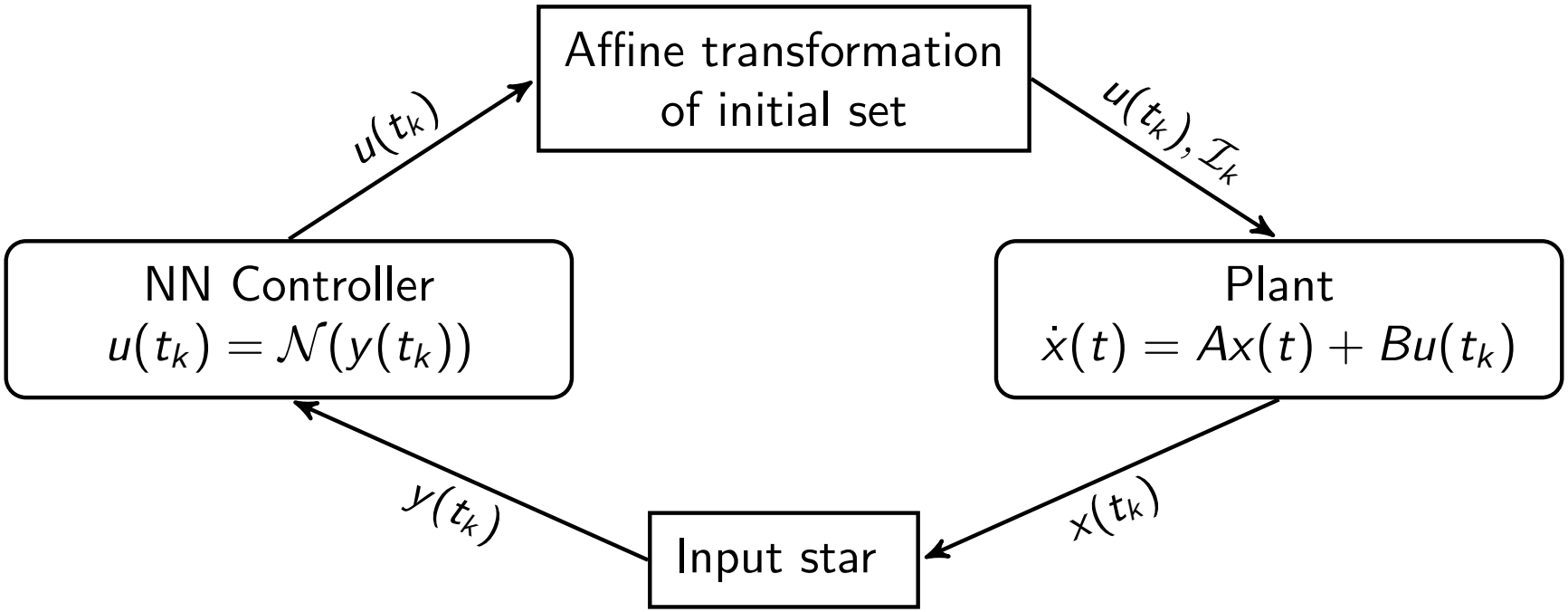
- Refine controller output to reduce over-approximation errors
- Handle nonlinear dynamics for comparability with existing techniques
- Extend to PID controller output
- Use reinforcement learning to find optimal controls
- Analyze theoretical time and space complexity

Future Work

- Refine controller output to reduce over-approximation errors
- Handle nonlinear dynamics for comparability with existing techniques
- Extend to PID controller output
- Use reinforcement learning to find optimal controls
- Analyze theoretical time and space complexity
- Overcome numerical challenges

Summary

Thank you for your attention!



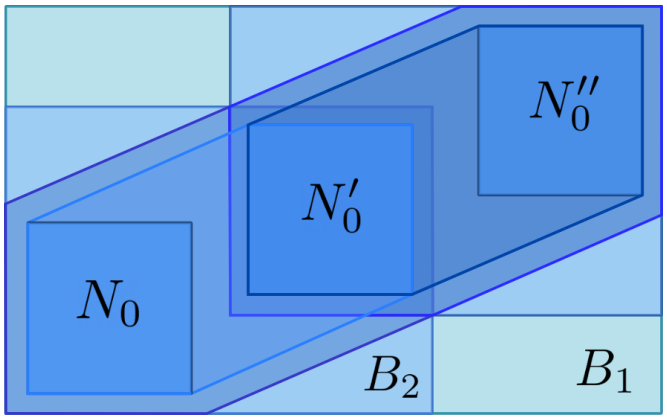
$$t_k = \delta, 2\delta, \ldots, K\delta$$

Contents

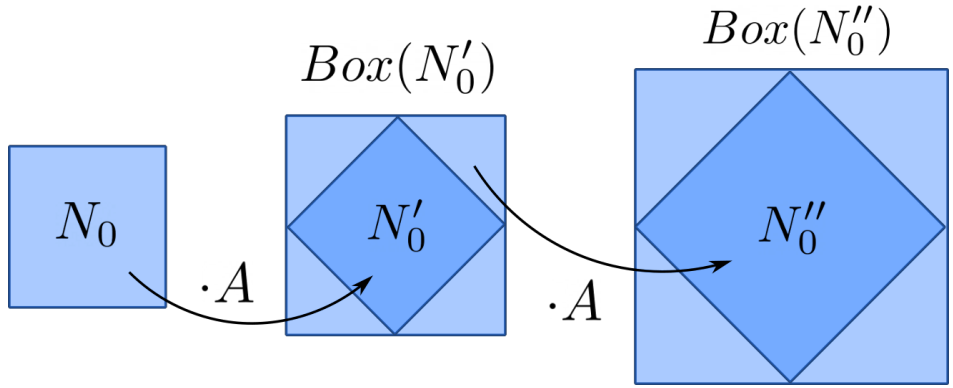
References I

- Alur, Rajeev et al. (1995). "The algorithmic analysis of hybrid systems". In: *Theoretical computer science* 138.1, pp. 3–34.
- Henzinger, Thomas A. et al. (1998). "What's Decidable about Hybrid Automata?" In: *Journal of Computer and System Sciences* 57.1, pp. 94–124. ISSN: 0022-0000. DOI: https://doi.org/10.1006/jcss.1998.1581. URL: https://www.sciencedirect.com/science/article/pii/S0022000098915811.
- Schupp, Stefan et al. (2017). "HyPro: A C++ library of state set representations for hybrid systems reachability analysis". In: *NASA Formal Methods Symposium*. Springer, pp. 288–294.
- Tran, Dung Hoang (2020). "Verification of Learning-Enabled Cyber-Physical Systems". PhD thesis. Vanderbilt University.

Wrapping Effects of Box Representation



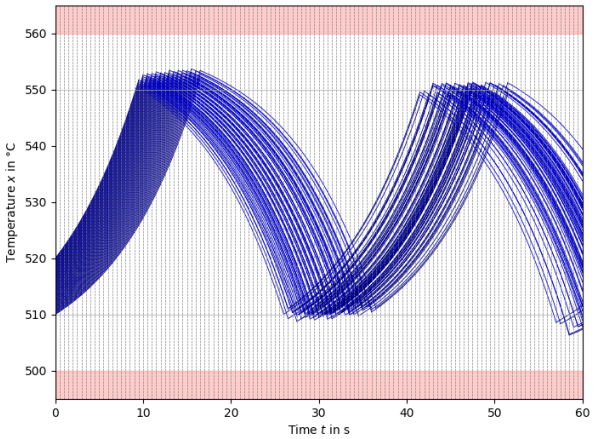
(a) Step-size dependent



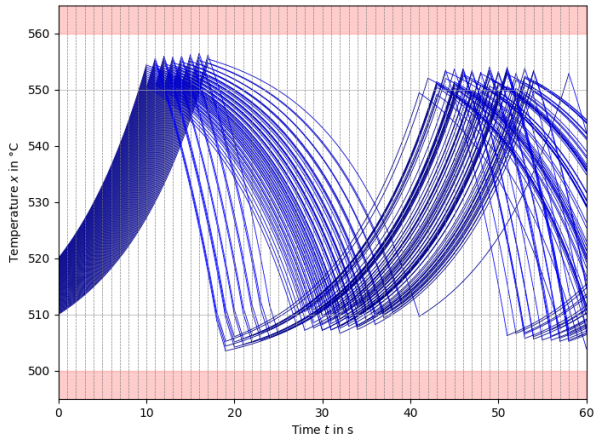
(b) Inherent over-approximation errors

Wrapping effects in the box representation.

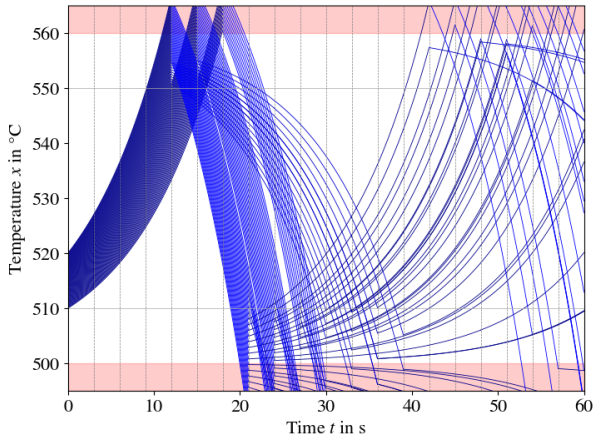
Step Size Rod Reactor



(a) $\delta = 0.5$



(b) $\delta = 1.0$



(c) $\delta = 3.0$

Reachability Analysis using Staircase Function

Algorithm Star-based exact reachability analysis for one layer with step function.

Input Input star set $I = [\Theta_1 \ \cdots \ \Theta_N]$, ordered classes $[h_1 \cdots h_m]$ with intervals $[A_1 \cdots A_m]$
Output Exact reachable set $\mathcal{R}$

procedure LAYERREACH(I, W, b)
  $\mathcal{R} \leftarrow \emptyset$;
  for $j = 1 : N$ **do**
    $I_1 \leftarrow W * \Theta_j + b = \langle Wc_j + b, WV_j, P_j \rangle$;
    $\mathcal{R}_1 \leftarrow I_1$;
    for $i = 1 : n$ **do**
      $[l_i, u_i] \leftarrow I_1.range(i)$; ▷ $l_i \leq x_i \leq u_i, x_i \in I_1[i]$
      $\mathcal{R}_1 \leftarrow$ STEPSTAIRCASE($\mathcal{R}_1, i, l_i, u_i$);
    end for
    $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_1$;
  end for
  return $\mathcal{R}$;
end procedure
procedure STEPSTAIRCASE($\tilde{I}, i, l_i, u_i$)
  $\tilde{\mathcal{R}} \leftarrow \emptyset$;
  $\tilde{I} = [\tilde{\Theta}_1 \ \cdots \ \tilde{\Theta}_k]$;
  $M \leftarrow [e_1 \ e_2 \ \cdots e_{i-1} \ 0 \ e_{i+1} \cdots e_n]$; ▷ Intermediate representations
  for $j = 1 : k$ **do**
    $\mathcal{R}_1 \leftarrow \emptyset$;
    $\tilde{\Theta}_j = \langle \tilde{c}_j, \tilde{V}_j, \tilde{P}_j, \rangle$;
    $i_{min} \leftarrow \arg\min_{i'} \{h_{i'} \mid l_i \in A_{i'}, i' \leq m\}$; ▷ Index of smallest class
    $i_{max} \leftarrow \arg\max_{i'} \{h_{i'} \mid u_i \in A_{i'}, i' \leq m\}$; ▷ Index of greatest class
    for $j' = i_{min} : i_{max}$ **do**
      $v_1 \leftarrow h_{j'} \cdot e_i$;
      $\tilde{\Theta}'_{j'} \leftarrow \langle M\tilde{c}_j + v_1, M\tilde{V}_j, \tilde{P}_j, \rangle$;
      $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup \tilde{\Theta}'_{j'}$;
    end for
    $\tilde{\mathcal{R}} \leftarrow \tilde{\mathcal{R}} \cup \mathcal{R}_1$;
  end for
  return $\tilde{\mathcal{R}}$;
end procedure

Proof of Over-approximation

Proof.

- Flowpipe construction with $\bar{\mathcal{R}}_{[0,\delta]}$ is over-approximation
- 1. controller invocation at time δ: reachable set $\mathcal{X}_i \subseteq \bar{\mathcal{R}}_{[0,\delta]}$ with $\mathcal{X}_i \cap \mathcal{R}_{[\delta,\delta]}$ is used to obtain the current state variable valuations for creating the input star Θ_I, which by design encompasses the reachable set at time δ.
- Exactness of star-based reachability algorithm
- Initial set for the first time step is $\mathcal{I}_k = e^{\delta A_0}\mathcal{X}_0$, for A_0 over $[0,\delta]$.
- Hence $\mathcal{I}_k \subseteq \bar{\mathcal{R}}_{[0,\delta]}$.
- Prediction yields a valid flow such that in the next iteration, flowpipe construction can be applied to the transformed initial set.

□