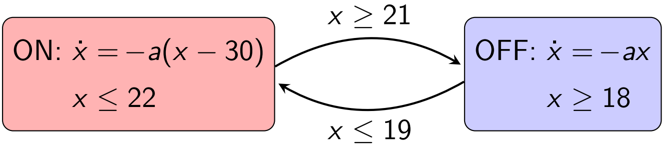
Reachability analysis of linear hybrid systems via block decomposition

Sergiy Bogomolov, Marcelo Forets, Goran Frehse, Kostiantyn Potomkin,
and **Christian Schilling**

EMSOFT 2020

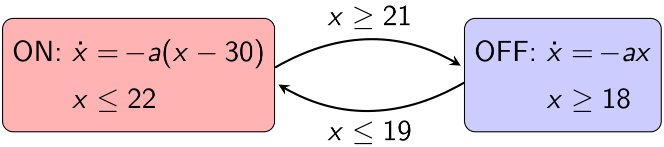
Overview

Linear hybrid systems



- Also called **hybrid automata with affine dynamics**

Linear hybrid systems



ON: $\dot{x} = -a(x - 30)$

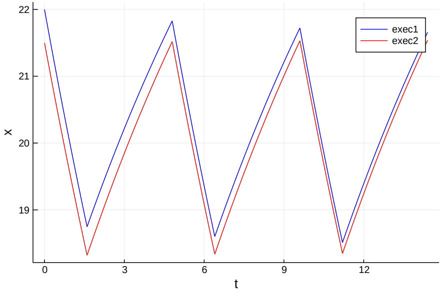
$x \leq 22$

$x \geq 21$

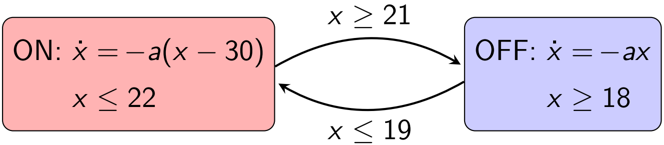
OFF: $\dot{x} = -ax$

$x \geq 18$

$x \leq 19$



Linear hybrid systems



ON: $\dot{x} = -a(x - 30)$

$x \leq 22$

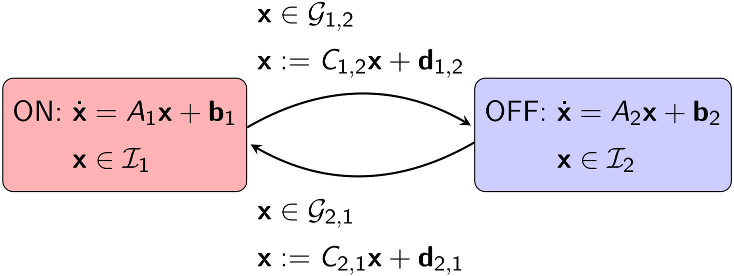
$x \geq 21$

OFF: $\dot{x} = -ax$

$x \geq 18$

$x \leq 19$

- General form:



$\mathbf{x} \in \mathcal{G}_{1,2}$

$\mathbf{x} := C_{1,2}\mathbf{x} + \mathbf{d}_{1,2}$

ON: $\dot{\mathbf{x}} = A_1\mathbf{x} + \mathbf{b}_1$

$\mathbf{x} \in \mathcal{I}_1$

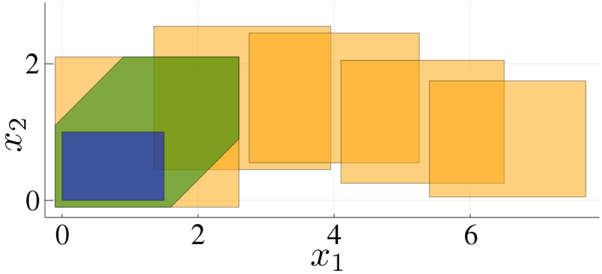
OFF: $\dot{\mathbf{x}} = A_2\mathbf{x} + \mathbf{b}_2$

$\mathbf{x} \in \mathcal{I}_2$

$\mathbf{x} \in \mathcal{G}_{2,1}$

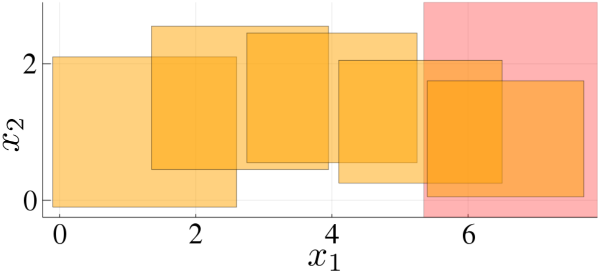
$\mathbf{x} := C_{2,1}\mathbf{x} + \mathbf{d}_{2,1}$

Reachability analysis for continuous systems



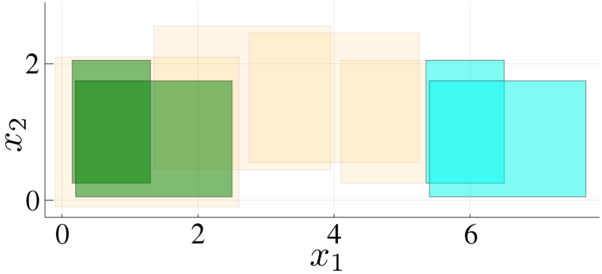
- Start from **set of initial states**
- Discretize time and **enclose all executions** in $t \in [0, \delta]$
- **Apply abstraction and propagate in discrete steps** until invariant is violated (not shown)
- **Flowpipe** (union of orange sets) encloses all executions

Discrete transitions: guard intersection



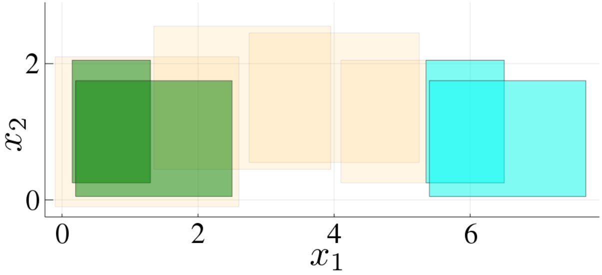
- Intersect with **transition guard** (polyhedron)

Discrete transitions: assignment



- Apply **assignment** (affine map) to **guard intersection**

Discrete transitions: fixpoint



- Check for **fixpoint** (inclusion in previous flowpipe)

Overview

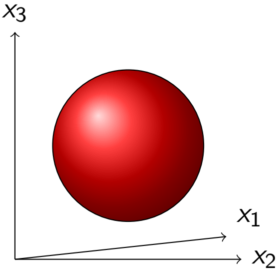
Preliminaries

Decomposed reachability analysis

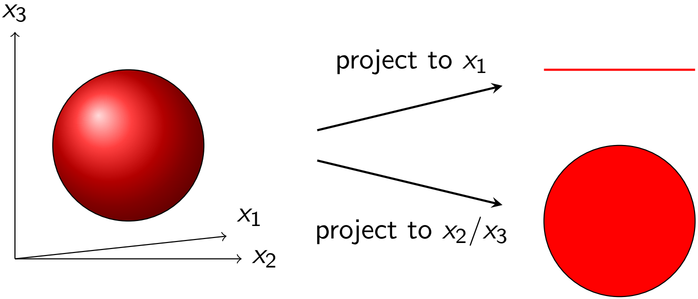
Decomposed intersection

Evaluation

Cartesian decomposition



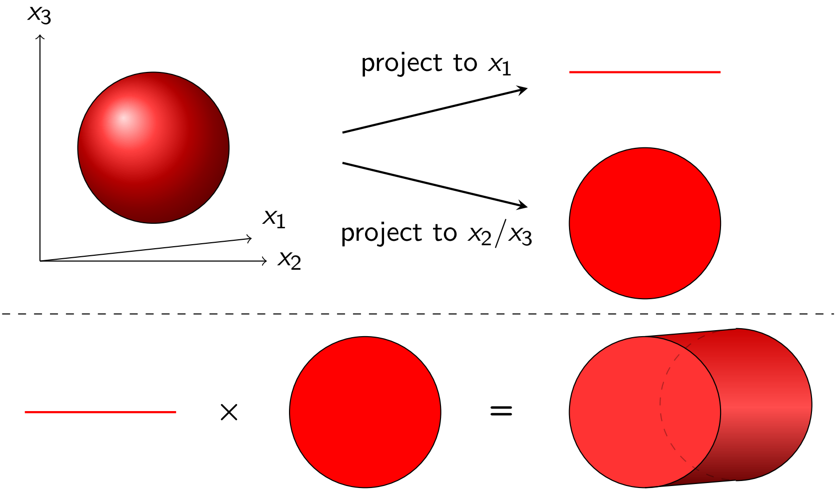
Cartesian decomposition



project to x_1

project to x_2/x_3

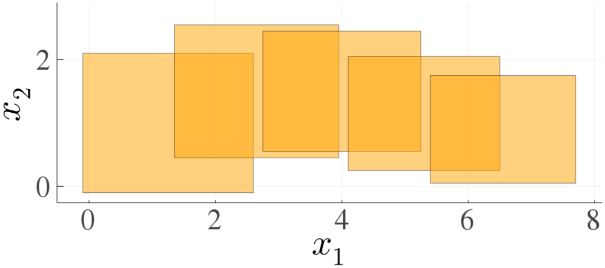
Cartesian decomposition



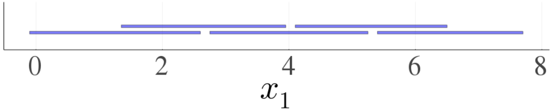
project to x_1

project to x_2/x_3

Decomposed reachability analysis for continuous systems[1]



- **Decomposition algorithm** computes **projected flowpipe**



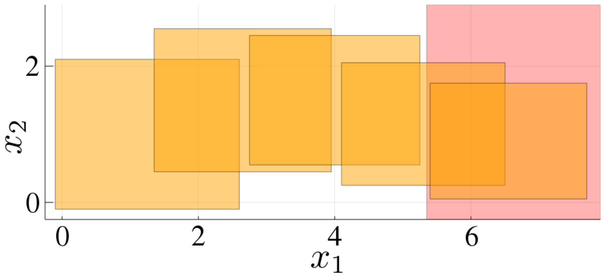
---

[1] S. Bogomolov et al. *HSCC*. 2018.

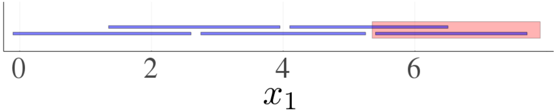
Decomposed reachability analysis for hybrid systems

- Properties of flowpipe from **decomposition algorithm**:
  - **Low dimensional** (only computed in selected dimensions)
  - **Decomposed** (Cartesian product of sets)
- Goal: use **decomposition algorithm** in **hybrid setting**
- Questions and obstacles:
  - Which **dimensions** do we need to compute?
  - Benefit from **low dimensions**
  - Benefit from **decomposed sets**
  - Decomposition algorithm needs **high-dimensional** input

Guards in decomposed setting

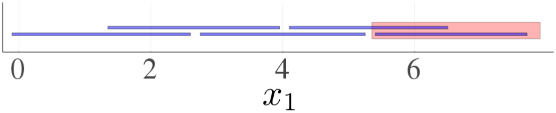


- **Decomposition algorithm** computes **projected flowpipe**

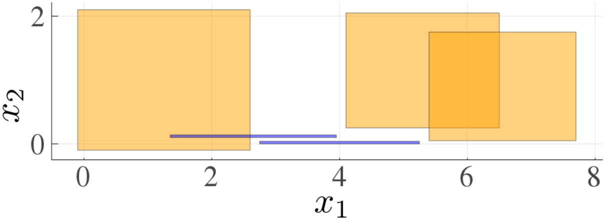


- **Guard** $\mathcal{G} := x_1 + 0x_2 \geq 5.35 \;\simeq\; \{x_1 \mid x_1 \geq 5.35\} \times \{x_2 \mid \top\}$ is **decomposed** in same blocks; projection on x_1: $x_1 \geq 5.35$

Guards in decomposed setting



- Compute flowpipe in **low dimensions**
- Detect intersection with **projected guard**
- Compute flowpipe in **high dimensions only if needed**



Decomposed reachability analysis for hybrid systems

Questions and obstacles:

✓ Which **dimensions** do we need to compute?
→ Dimensions constrained in guards (and safety property)

✓ Benefit from **low dimensions**
→ Computation in low dimensions

✓ Benefit from **decomposed sets**
→ Other operations (assignment, fixpoint check, clustering) special-cased for decomposed sets (not presented)

✓ Decomposition algorithm needs **high-dimensional** input
→ Result of guard intersection is high dimensional

Overview

Preliminaries

Decomposed reachability analysis

Decomposed intersection

Evaluation

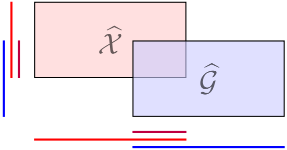
Decomposed guard intersection

- For $\widehat{\mathcal{X}} = \mathcal{X}_1 \times \mathcal{X}_2, \widehat{\mathcal{G}} = \mathcal{G}_1 \times \mathcal{G}_2$ with $\mathcal{X}_1, \mathcal{G}_1 \subseteq \mathbb{R}^n, \mathcal{X}_2, \mathcal{G}_2 \subseteq \mathbb{R}^m$ we have

$$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = (\mathcal{X}_1 \times \mathcal{X}_2) \cap (\mathcal{G}_1 \times \mathcal{G}_2) = (\mathcal{X}_1 \cap \mathcal{G}_1) \times (\mathcal{X}_2 \cap \mathcal{G}_2)$$

- Corollary:

$$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = \emptyset \iff (\mathcal{X}_1 \cap \mathcal{G}_1 = \emptyset) \vee (\mathcal{X}_2 \cap \mathcal{G}_2 = \emptyset)$$



Decomposed guard intersection

- For $\widehat{\mathcal{X}} = \mathcal{X}_1 \times \mathcal{X}_2, \widehat{\mathcal{G}} = \mathcal{G}_1 \times \mathcal{G}_2$ with $\mathcal{X}_1, \mathcal{G}_1 \subseteq \mathbb{R}^n, \mathcal{X}_2, \mathcal{G}_2 \subseteq \mathbb{R}^m$ we have

  $$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = (\mathcal{X}_1 \times \mathcal{X}_2) \cap (\mathcal{G}_1 \times \mathcal{G}_2) = (\mathcal{X}_1 \cap \mathcal{G}_1) \times (\mathcal{X}_2 \cap \mathcal{G}_2)$$

- Corollary:

  $$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = \emptyset \iff (\mathcal{X}_1 \cap \mathcal{G}_1 = \emptyset) \vee (\mathcal{X}_2 \cap \mathcal{G}_2 = \emptyset)$$

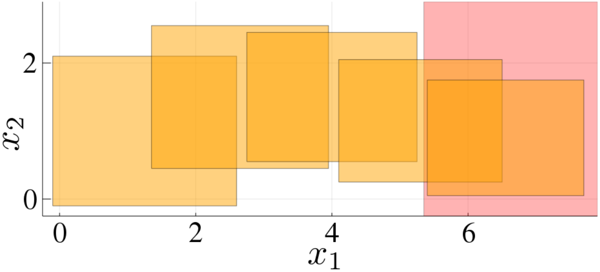
- If furthermore $\mathcal{G}_2$ is universal and $\mathcal{X}_2$ is nonempty:

  $$\mathcal{X}_2 \cap \mathcal{G}_2 = \mathcal{X}_2 \neq \emptyset$$

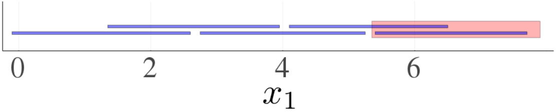
  Hence

  $$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = \emptyset \iff \mathcal{X}_1 \cap \mathcal{G}_1 = \emptyset$$

Recall: Decomposed guard intersection



- **Decomposition algorithm** computes **projected flowpipe**



Non-decomposed guard intersection

- Cases for intersection $\widehat{\mathcal{X}} \cap \mathcal{G}$ with $\widehat{\mathcal{X}} = \mathcal{X}_1 \times \mathcal{X}_2$:
  - ✓ $\widehat{\mathcal{X}}$ and $\mathcal{G}$ **decomposed** in same structure
  - $\mathcal{G}$ **not decomposed** in same structure as $\widehat{\mathcal{X}}$
  - $\mathcal{G}$ **not decomposed** at all

Non-decomposed guard intersection

- Cases for intersection $\widehat{\mathcal{X}} \cap \mathcal{G}$ with $\widehat{\mathcal{X}} = \mathcal{X}_1 \times \mathcal{X}_2$:
  - ✓ $\widehat{\mathcal{X}}$ and $\mathcal{G}$ **decomposed** in same structure
  - $\mathcal{G}$ **not decomposed** in same structure as $\widehat{\mathcal{X}}$
  - $\mathcal{G}$ **not decomposed** at all

- Let π_1 and π_2 be suitable projection matrices

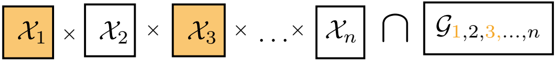
$$\widehat{\mathcal{G}} := \pi_1\mathcal{G} \times \pi_2\mathcal{G} \quad \supseteq \mathcal{G}$$

$$\widehat{\mathcal{X}} \cap \widehat{\mathcal{G}} = (\mathcal{X}_1 \cap \pi_1\mathcal{G}) \times (\mathcal{X}_2 \cap \pi_2\mathcal{G}) \quad \supseteq \widehat{\mathcal{X}} \cap \mathcal{G}$$

Decomposed intersection algorithms

Task: $\boxed{\mathcal{X}_1} \times \boxed{\mathcal{X}_2} \times \boxed{\mathcal{X}_3} \times \ldots \times \boxed{\mathcal{X}_n} \bigcap \boxed{\mathcal{G}_{1,2,3,\ldots,n}}$

Decomposed intersection algorithms

Task: 

$\boxed{\mathcal{X}_1} \times \boxed{\mathcal{X}_2} \times \boxed{\mathcal{X}_3} \times \ldots \times \boxed{\mathcal{X}_n} \bigcap \boxed{\mathcal{G}_{1,2,3,\ldots,n}}$

1) High-dimensional intersection:

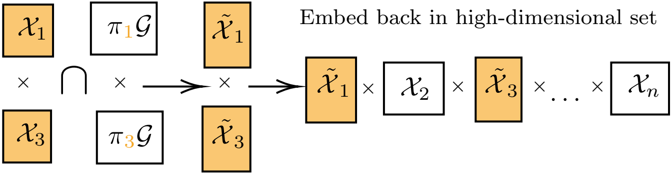
$\boxed{\mathcal{X}_1}\boxed{\mathcal{X}_2}\boxed{\mathcal{X}_3}\boxed{\cdots}\boxed{\mathcal{X}_n} \bigcap \boxed{\mathcal{G}_{1,2,3,\ldots,n}} \longrightarrow \boxed{\tilde{\mathcal{X}}_{1,2,3,..,n}}$

- Expensive
- Exact

Decomposed intersection algorithms

Task: $\boxed{\mathcal{X}_1} \times \boxed{\mathcal{X}_2} \times \boxed{\mathcal{X}_3} \times \ldots \times \boxed{\mathcal{X}_n} \bigcap \boxed{\mathcal{G}_{1,2,3,\ldots,n}}$

2) Low-dimensional intersection:

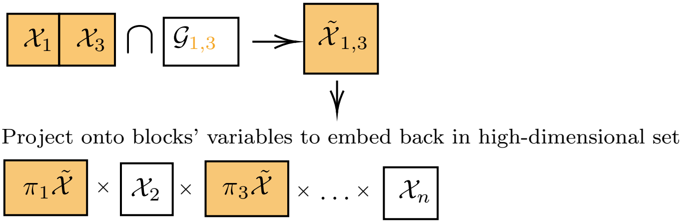


- Efficient
- Projection $\pi\mathcal{G}$ can be very coarse

Decomposed intersection algorithms

Task: $\boxed{\mathcal{X}_1} \times \boxed{\mathcal{X}_2} \times \boxed{\mathcal{X}_3} \times \dots \times \boxed{\mathcal{X}_n} \bigcap \boxed{\mathcal{G}_{1,2,3,\dots,n}}$

3) Medium-dimensional intersection:



$$\boxed{\mathcal{X}_1}\boxed{\mathcal{X}_3} \bigcap \boxed{\mathcal{G}_{1,3}} \longrightarrow \boxed{\tilde{\mathcal{X}}_{1,3}}$$

$$\downarrow$$

Project onto blocks' variables to embed back in high-dimensional set

$$\boxed{\pi_1\tilde{\mathcal{X}}} \times \boxed{\mathcal{X}_2} \times \boxed{\pi_3\tilde{\mathcal{X}}} \times \dots \times \boxed{\mathcal{X}_n}$$

- Middle ground
- First step exact
- Projection applied to a bounded set only

Overview

Preliminaries

Decomposed reachability analysis

Decomposed intersection

Evaluation

Outline

Experiments:

- Scaling in number of **constrained dimensions** k for intersection operation
- Reachability analysis
- Scaling in k for reachability analysis

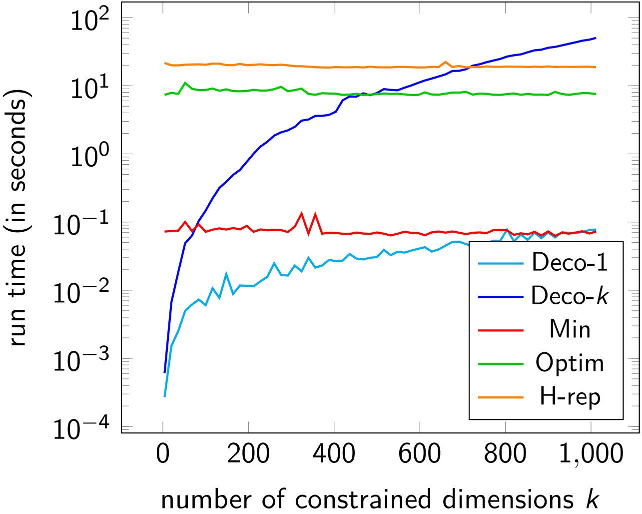
Scalability in k: Intersection operation

- Experiment: overapproximate $\mathcal{X} \cap \mathcal{G}_k$ where $\mathcal{X} \subseteq \mathbb{R}^{1024}$ is a hypercube and $\mathcal{G}_k$ is the half-space $x_1 + \cdots + x_k \leq 2$ (Note: $\mathcal{G}_k$ properly cuts $\mathcal{X}$ for $k > 0$)
- Evaluate different algorithms to approximate the intersection
    - *Deco-1*: project (here: to 1D blocks), compute exactly via half-space representation, and recombine ("low")
    - *Deco-k*: same as DecoLow but in kD ("medium")
    - *Min*: estimate support function via (coarse) heuristics $\rho_{\mathcal{X} \cap \mathcal{G}}(\ell) \leq \min(\rho_{\mathcal{X}}(\ell), \rho_{\mathcal{G}}(\ell))$
    - *Optim*: estimate support function via optimization[1]
    - *H-rep*: compute exactly via half-space representation

---

[1] G. Frehse and R. Ray. *ADHS*. 2012.

Scalability in k: Intersection operation



Reachability algorithms

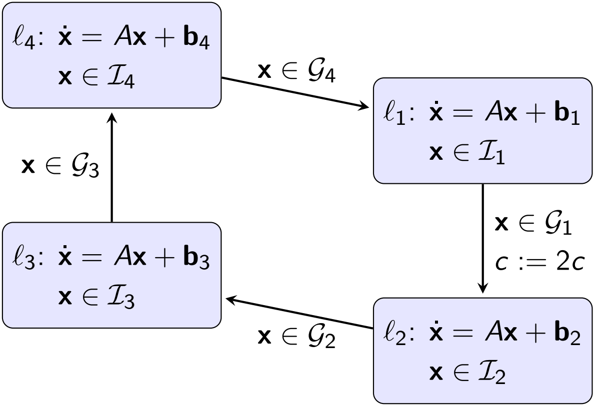
- Evaluate different algorithms (verifying the safety property)
  - *Deco*: our algorithm (decomposed algorithms for both continuous and hybrid part), using 1D blocks and medium-dimensional intersection
  - *Optim*: decomposed algorithm for continuous part but high-dimensional algorithm for hybrid part, using 1D blocks and "Optim" intersection
  - *SpaceEx LGG*: support-function algorithm[1]
  - *SpaceEx STC*: extension with automatic time step[2]

---

[1] C. L. Guernic and A. Girard. *CAV*. 2009.

[2] G. Frehse et al. *HSCC*. 2013.

Filtered oscillator



$\ell_4\colon\ \dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}_4$
$\mathbf{x} \in \mathcal{I}_4$

$\mathbf{x} \in \mathcal{G}_4$

$\ell_1\colon\ \dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}_1$
$\mathbf{x} \in \mathcal{I}_1$

$\mathbf{x} \in \mathcal{G}_3$

$\mathbf{x} \in \mathcal{G}_1$
$c := 2c$

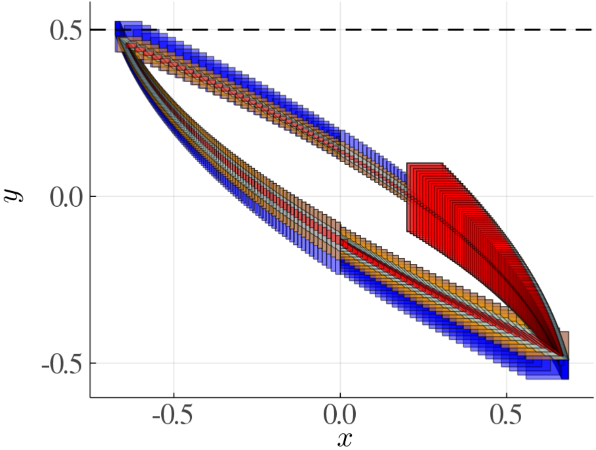
$\ell_3\colon\ \dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}_3$
$\mathbf{x} \in \mathcal{I}_3$

$\mathbf{x} \in \mathcal{G}_2$

$\ell_2\colon\ \dot{\mathbf{x}} = A\mathbf{x} + \mathbf{b}_2$
$\mathbf{x} \in \mathcal{I}_2$

- m filters, $\mathbf{x} = [x, y, x_1, \ldots, x_m]$ (plus auxiliary variable c)
- Constraints only in x and y (and c)
- Auxiliary variable c used to have just one loop iteration

Filtered oscillator



Deco with 1D blocks: dark blue (Deco-1 intersection), orange (Deco-k intersection)
Deco with 2D blocks and octagon approximation: light blue
SpaceEx LGG: red
Dashed line: threshold for safety property $y < 0.5$

Models

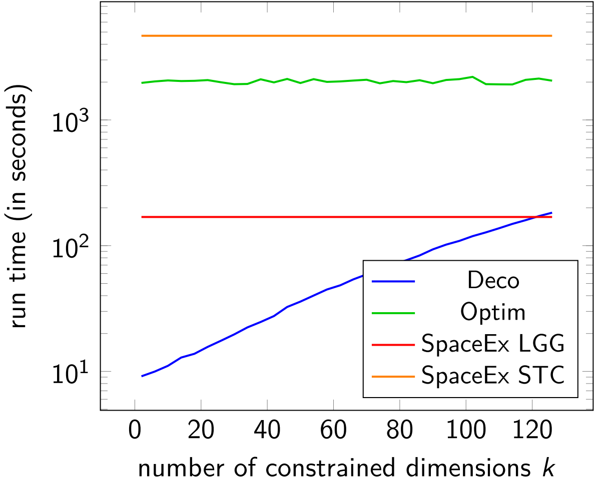
Benchmark	Dim (k)	ID
linear_switching	5 (1)	1
spacecraft_noabort	5 (4)	2a
spacecraft_120	5 (5)	2b
platoon_bounded	10 (4)	3a
platoon_unbounded	10 (4)	3b
filtered_osc64	67 (3)	4a
filtered_osc128	131 (3)	4b
filtered_osc256	259 (3)	4c
filtered_osc512	515 (3)	4d
filtered_osc1024	1027 (3)	4e

Reachability analysis on different models

ID	Dim (k)	Step	Deco	Min	Optim	SpaceEx LGG	SpaceEx STC
1	5 (1)	10^{-4}	$\mathbf{2.50 \times 10^{0}}$	1.27×10^{1}	2.81×10^{1}	2.60×10^{1}	2.30×10^{1}
2a	5 (4)	0.04	5.30×10^{0}	3.42×10^{0}	2.19×10^{2}	1.18×10^{0}	$\mathbf{3.50 \times 10^{-1}}$
2b	5 (5)	0.04	5.30×10^{0}	2.10×10^{0}	4.30×10^{1}	1.91×10^{0}	$\mathbf{8.10 \times 10^{-1}}$
3a	10 (4)	0.01	$\mathbf{1.30 \times 10^{-1}}$	1.60×10^{-1}	5.69×10^{0}	5.55×10^{0}	1.60×10^{0}
3b	10 (4)	0.03	$\mathbf{1.08 \times 10^{0}}$	1.16×10^{0}	4.96×10^{1}	3.46×10^{1}	6.50×10^{1}
4a	67 (3)	0.01	$\mathbf{2.81 \times 10^{0}}$	$(7.43 \times 10^{0})^{\dagger}$	5.63×10^{2}	2.04×10^{1}	3.25×10^{1}
4b	131 (3)	0.01	$\mathbf{7.95 \times 10^{0}}$	$(4.29 \times 10^{1})^{\dagger}$	1.79×10^{3}	1.69×10^{2}	4.67×10^{3}
4c	259 (3)	0.01	$\mathbf{2.80 \times 10^{1}}$	$(9.19 \times 10^{1})^{\dagger}$	9.99×10^{4}	8.70×10^{3}	OOM
4d	515 (3)	0.01	$\mathbf{1.13 \times 10^{2}}$	$(4.73 \times 10^{2})^{\dagger}$	TO	TO	TO
4e	1027 (3)	0.01	$\mathbf{5.09 \times 10^{2}}$	$(5.11 \times 10^{3})^{\dagger}$	TO	TO	TO

†Safety property could not be proven (overapproximation too coarse)

Scalability in k: Reachability analysis



- Add small constraints for k previously unconstrained dimensions in invariants and guards (*filtered_osc128* model)

Conclusion

- **Decomposed reachability algorithm** for linear hybrid systems
- Integration of **decomposed reachability algorithm** for affine continuous systems
- Compute **low-dimensional flowpipe**
- Detect **guard intersection** in **low dimensions**
- Compute **high-dimensional flowpipe only when necessary**
- **Highly scalable** yet **precise** under appropriate conditions (decomposed, sparsely constrained guards etc.)